PIM-based Accelerators for Graph Analytics

Makesh Tarun Chandran
Department of Electrical Engineering and Computer Science
Pennsylvania State University
Email: mzc88@psu.edu

I. Introduction

Graph algorithms are an interesting class of application for which a single mac mini desktop equipped with SSD outperforms a medium sized cluster! [1] The primary reason for such a counter-intuitive behavior can be attributed to the irregular data-access pattern inherent in those algorithms. They also have a low computation to communication ratio which is exacerbated in the modern memory-bandwidth limited systems. However, graph algorithms are highly iterative and there is abundance of parallelism that can be exploited. These idiosyncrasies of graph algorithm render commercial HPC systems inefficient or unsuitable for accelerating such applications. Extensive research on graph processing in the past decade have revealed new insights on the algorithmic, programming model and hardware aspects of such systems. I have tried to comprehend the relevant and significant contributions of previous work in the following section.

II. Graph Processing

Bring out the importance of Graph processing from [13]–[15] and discuss state-of-the-art systems. Take the example of SSSP application and discuss how the Implementation changes with various models.

A. Programming model

Discuss the Vertex-based [2] and Gather-Apply-Scatter [3] based programming models here. Highlight the differences and bring out the suitability of each of them for PIM operations. Also show the pseudo code for SSSP application.

B. Execution Model

Two different paradigms: Synchronous vs Asynchronous [6], Ordered vs Unordered [4]

III. In/Near-Memory Processing

A. Near-Data Processing

Discuss [16] like architectures where the computations are moved closer to the memory elements but not inside them. End the section by indicating that more bandwidth can be exploited when the processing is integrated with memory.

B. In-memory Processing

1) DRAM-PIM: Present a basic idea of the [11] paper which proposes a architecture for processing-in-memory architecture built on commercially demonstrated Hybrid Memory Cube. Also highlight the loopholes/assumptions with such design. Also list other architectures, like GraphH [17], which builds on the technology.

2) NVM-PIM: Briefly list the problems with DRAM in terms of energy and performance, and bring out NVMs as an alternative to DRAM. [12] introduces a efficient NOR operation which can be performed on the bitlines of ReRAM based memory. Show example architecture of NVM PIN and explain it working with relevant figures.

IV. Accelerators

An Ideal graph processing system is where the performance increases proportional to the size of the graphs that can be stored in the system. Unfortunately, in conventional systems, memory bandwidth remains almost constant irrespective of the memory capacity due to pin count limitation per chip. Traditional caching mechanisms fail to sustain the memory throughput requirement due to the application's poor locality.

Following is the summary of various PIM-based techniques developed to mitigate the performance bottleneck in graph processing systems.

A. Graphicionado

[5] replaces the conventional on-chip memory hierarchy (caches) with explicitly managed ScratchPad memory. The accelerator achieves upto 6x speedup while consuming less than 2% of energy of conventional systems. However, larger graph applications that doesn't fit in the on-chip memory will incur off-chip communication which is detrimental to the performance.

B. Tesseract

[6] exploits the internal bandwidth of the HMC-RAM, which is an order of magnitude higher than the off-chip bandwidth, by integrating a logic layer within the memory die. The accelerator out performs conventional systems by a factor of 10 and achieves memory-capacity-proportional bandwidth to provide scalable performance improvements.

C. GraphR

[7] is a RRAM based in-memory graph processing accelerator. They perform analog computations using RRAM crossbar structure and achieve a speedup of 4x and energy efficiency of 11x compared to Tesseract. They exploit the fact that the graph algorithms can inherently tolerate imprecision and are resilient to errors.

D. GraphP

[8] takes a data-organisation centric hardware-software design approach to minimise communication in a PIM based accelerator, to achieve a speedup of up to 1.7x and energy efficiency on of 89% compared to Tesseract.

E. GraphH

[17] GraphH takes a fresh looks at the graph analytics systems and proposes a DRAM-PIM based architecture to tackle challenges in graph processing. The proposed architecture outperforms Graphicionado by a factor of 5.12x.

All these systems have customised their programming and execution model to derive maximum performance from their hardware.

V. Design Space

A brief study of the architectures listed above reveal the following parameters in the design space of an accelerator for graph analytics.

- Core complexity - In-Order, Out-of-Order, Superscalar
- Memory type - DRAM, HMC, RRAM, Other NVMs
- Algorithms - Vertex-based [2] vs Gather-Apply-Scatter [3], Ordered vs Unordered [4]
- Power - Power Throttling in PIM based accelerators [10]
- Others - Data-Organisation [8], Accuracy [7], Synchronous vs Asynchronous Execution [6]

VI. Completed and Future Work

The architectures listed above were proposed in the last three years, with the advent of new memory technologies which made in-memory processing feasible. An in-depth study of the hardware and software stack of these PIM-based accelerators and their design choices, may expose untapped design points to improve performance. The plan was to begin by understanding various design points listed in Section V, evaluate the accelerators for strengths and limitations, and identify potential improvements in their design, leading to a survey paper or an original research paper.

I have read papers GraphR [7], GraphP [8] and other papers based on NVM-PIM [18]. Apart from that I also ramped up with Pregel and ZSim Simulation framework. The next step would be to prototype one of the architectures (possibly GraphR or GraphH) and identify bottlenecks to improve the perforamance. A survey paper on this topic makes little sense as the in-memory processing of graphs is a relatively narrow and emerging field. Moreover, there are recent resources [19], [20] that contain an exhaustive analysis of the topic.

References

[1] Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. 2012. GraphChi: large-scale graph computation on just a PC. In Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation (OSDI'12). USENIX Association, Berkeley, CA, USA, 31-46.

[2] Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data (SIGMOD '10). ACM, New York, NY, USA, 135-146. DOI: https://doi.org/10.1145/1807167.1807184

[3] Joseph E. Gonzalez, Yucheng Low, Haijie Gu, Danny Bickson, and Carlos Guestrin. 2012. PowerGraph: distributed graph-parallel computation on natural graphs. In Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation (OSDI'12). USENIX Association, Berkeley, CA, USA, 17-30.

[4] Amber Hassaan, Martin Burtscher, and Keshav Pingali. 2011. Ordered vs. unordered: a comparison of parallelism and work-efficiency in irregular algorithms. In Proceedings of the 16th ACM symposium on Principles and practice of parallel programming (PPoPP '11). ACM, New York, NY, USA, 3-12. DOI=http://dx.doi.org/10.1145/1941553.1941557

[5] T. J. Ham, L. Wu, N. Sundaram, N. Satish and M. Martonosi, "Graphicionado: A high-performance and energy-efficient accelerator for graph analytics," 2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Taipei, 2016, pp. 1-13.

[6] J. Ahn, S. Hong, S. Yoo, O. Mutlu and K. Choi, "A scalable processing-in-memory accelerator for parallel graph processing," 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), Portland, OR, 2015, pp. 105-117.

[7] L. Song, Y. Zhuo, X. Qian, H. Li and Y. Chen, "GraphR: Accelerating Graph Processing Using ReRAM," 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, 2018, pp. 531-543.

[8] M. Zhang et al., ”GraphP: Reducing Communication for PIM-Based Graph Processing with Efficient Data Partition,” 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, 2018, pp. 544-557.
[9] Muhammet Mustafa Ozdal, Serif Yesil, Taemin Kim, Andrey Ayupov, John Greth, Steven Burns, and Ozcan Ozturk. 2016. Energy efficient architecture for graph analytics accelerators. In Proceedings of the 43rd International Symposium on Computer Architecture (ISCA ’16). IEEE Press, Piscataway, NJ, USA, 166-177. DOI: https://doi.org/10.1109/ISCA.2016.24
[10] C. Zhang, T. Meng and G. Sun, ”PM3: Power Modeling and Power Management for Processing-in-Memory,” 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, 2018, pp. 558-570.
[11] R. Nair et al., ”Active Memory Cube: A processing-in-memory architecture for exascale systems,” in IBM Journal of Research and Development, vol. 59, no. 2/3, pp. 17:1-17:14, March-May 2015. doi: 10.1147/JRD.2015.2409732
[12] S. Kvatinsky et al., ”MAGICMemristor-Aided Logic,” in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 11, pp. 895-899, Nov. 2014. doi: 10.1109/TCSII.2014.2357292
[13] Muhammad Imran, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. 2015. Processing Social Media Messages in Mass Emergency: A Survey. ACM Comput. Surv. 47, 4, Article 67 (June 2015), 38 pages. DOI: https://doi.org/10.1145/2771588
[14] C. Unsalan and B. Sirmacek, ”Road Network Detection Using Probabilistic and Graph Theoretical Methods,” in IEEE Transactions on Geoscience and Remote Sensing, vol. 50, no. 11, pp. 4441-4453, Nov. 2012.
[15] Bullmore, Ed Sporns, Olaf Complex brain networks: graph theoretical analysis of structural and functional systems DOI: http://dx.doi.org/10.1038/nrn2575
[16] Boncheol Gu, Andre S. Yoon, Duck-Ho Bae, Insoon Jo, Jinyoung Lee, Jonghyun Yoon, Jeong-Uk Kang, Moonsang Kwon, Chanho Yoon, Sangyeun Cho, Jaeheon Jeong, and Duckhyun Chang. 2016. Biscuit: a framework for near-data processing of big data workloads. SIGARCH Comput. Archit. News 44, 3 (June 2016), 153-165. DOI: https://doi.org/10.1145/3007787.3001154
[17] G. Dai et al., ”GraphH: A Processing-in-Memory Architecture for Large-scale Graph Processing,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. doi: 10.1109/TCAD.2018.2821565
[18] Y. K. Rupesh, P. Behnam, G. R. Pandla, M. Miryala and M. Nazm Bojnordi, ”Accelerating k-Medians Clustering Using a Novel 4T-4R RRAM Cell,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems. doi: 10.1109/TVLSI.2018.2808468
[19] Systems for Big Graph Analytics by Yan, Da, Tian, Yuanyuan, Cheng, James
[20] Gupta, S. (2018). Processing in Memory using Emerging Memory Technologies. UC San Diego. ProQuest ID: Merritt ID: ark:/13030/m5bg7kwn. Retrieved from https://escholarship.org/uc/item/95z3z84c