Literature Review

Cheng Liu

2016-10-06

1 xDGP

With the observation that graph partition has deep influence on the performance of graph traversing and optimal partition for a dynamic graph changes with time, thus it is critial to present a dynamic graph partition mechanism. This work [5] adopts vertex migration to achive the dynamic partition. The vertex migration strategy is drived from label propagation algorithm in data mining. Basically a vertex decides the migration based on its neighbors. As the algorthm uses only local information, it ensures the performance as well as the scalability.

The basic idea of this work is simple and it is developed over google pregel framework. The authors spent most of the pages on experiments.

2 Accelerator Design for Graph Analytics

This paper [3] is essentially a graph accelerator implementation framework of Gather-Apply-Scatter model as in GraphLab [2]. It particularly focuses on iterative graph-parallel applications with asynchronous execution and asymmetric convergence. In order to support domain of graph processing, it has a template of hardware for common operations including memory access, synchronization and communication. In order to provide application specific optimization, a design space exploration is also supported like typical domain specific accelerators.

2.1 Highlights

Here are the list of highlights of this work.

- It targets graph applications with asynchronous execution and asymmetric convergence which doesn't work well on GPUs. This is also one of the major reasons that contributes to the the high power efficiency.
- It provides a hardware version of GraphLab [2] and maintains sequential consistency model with a synchronous unit (SYU) which essentially follows the edge consistency.
- memory access optimization: It has special cache, load, store units for each data type such as vertex information (VI) and edge information (EI). The

cache structure is also configurable to meet the requirements of as VI and EI which have different locality characteristic.

- Graph partition: The framework has each accelerator optimized for fine grained operation level parallelism. And it also replicates the accelerator unit to explore high-level parallelism based on a static graph partition.

2.2 Questions

Is it necessary to maintain sequential consistency, will it be possible to loose the consistency model for more parallelism and higher performance?

According to the Graphicionado [1], cache may not be a good memory hierarchy for graph processing as the graph problems typically has poor locality. Will a scratch pad memory work better for this design? This work utilize vertex and edge as the basic cache granularity instead of general data type may probably alleviate the problem.

The graph partition is not detailed, how does the partition affect the overall system performance?

3 Graphicioando

This paper utilize GraphMat [4] as the graph processing framework. With the observation that graph processing has ineffective usage of both on-chip memory and bandwidth, this work particularly optimizes the on chip memory usage over the baseline hardware accelerator obtained from GraphMat.

According to the pipeline of the baseline accelerator, the on chip memory access characteristics of the different pipeline stages are analyzed and a few optimizations are applied to remove the bottleneck of the accelerator pipeline. Here are the list of the major optimizations.

- It uses on-chip eDRAM as scratchpad memory to alleviate the random destination vertex and edge ID access. For the rest of the sequential memory access, a prefetch scheme is used to hide the memory access latency,

- Instead of replicating the accelerator directly, the authors divide the processing phase into source oriented portion and destination oriented portion.

With this strategy, the hardware can be easily partitioned into parts without overlaps. Basically, the scratchpad memory is shared among the vertex processing streams.

- The number of edges are typically much larger than that of the vertex, so the edge access is usually a bottleneck of the accelerator design. This work uses an array of input queues and output queues connected with a crossbar to access memory and feed data to the downstream processing.
- In order to cope with graphs with larger scratchpad memory requirements, the graph is sliced, though the slicing is a simple one.

4 Database query with hardware/software co-design

This work is developed to handle OLTP, but it doesn't show any special design optimization for OLTP system. According to my understanding, the FPGA acceleration for query itself is kind of OLTP support.

Here are the highlights of this paper.

- The design has a query control block (QCB) included to support configuration for different queries. Basically, it bridges the gap between the database query and the accelerator.
- This work details how the database query can be transformed to the accelerator configuration which I seldom see in other papers (page 9). The authors argue that SQL statement doesn't include enough information for the accelerator and they transform the query operations based on the output of DBMS transformation instead of AST derived from SQL directly.
- This paper is an extension of previous work. It particularly presents the sorting (Tournament tree sorting) and predicate evaluation implementation.
- When the query can't be mapped to the FPGA accelerator, the query operation can be decomposed to sub-operations and intermediate results will be stored as temporary files. Then it relies the software to merge the temporary files for the result.

References

[1] Tae Jun Ham, Lisa Wu, Narayanan Sundaram, Nadathur Satish, and Margaret Martonosi. Graphicionado: A high-performance and energy-efficient accelerator for graph analytics.

[2] Eriko Nurvitadhi, Gabriel Weisz, Yu Wang, Skand Hurkat, Marie Nguyen, James C Hoe, José F Martínez, and Carlos Guestrin. Graphgen: An fpga framework for vertex-centric graph computation. In *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, pages 25–28. IEEE, 2014.

[3] Muhammet Mustafa Ozdal, Serif Yesil, Taemin Kim, Andrey Ayupov, John Greth, Steven Burns, and Ozcan Ozturk. Energy efficient architecture for graph analytics accelerators. In *Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on*, pages 166–177. IEEE, 2016.

[4] Narayanan Sundaram, Nadathur Satish, Md Mostofa Ali Patwary, Subramanya R Dulloor, Michael J Anderson, Satya Gautam Vadlamudi, Dipankar Das, and Pradeep Dubey. Graphmat: High performance graph analytics made productive. *Proceedings of the VLDB Endowment*, 8(11):1214–1225, 2015.

[5] Luis Vaquero, Félix Cuadrado, Dionysios Logothetis, and Claudio Martella. xdgp: A dynamic graph processing system with adaptive partitioning. *arXiv preprint arXiv:1309.1049*, 2013.