《专业前沿文献阅读》

课程论文报告

学　　号：20191001171

班级序号：111192

姓　　名：滕德淋

指导老师：姜宇虹

中国地质大学（武汉）

地理与信息工程学院

2021 年 1 月

评　　语

对课程论文的评语:

平时成绩:	课程论文成绩:
总 成 绩:	评阅人签名:

注:

1、无评阅人签名成绩无效;

2、必须用钢笔或圆珠笔批阅，用铅笔阅卷无效;

3、如有平时成绩，必须在上面评分表中标出，并计算入总成绩。

Topic： Review of distributed graph computing

Text:

In this era of interconnected world, graphics analysis applications are everywhere. These applications have very low computation to byte transfer ratio and poor locality, which limits their computational efficiency on general-purpose computing systems. Traditional hardware accelerators overcome these challenges by using customized data flow and memory hierarchy [1].

This essay introduces some research on Distributed graph computing, a sub-topic in the field of graph computing. Method-1 is the Gaas-X [2]. Gaas-X is a graph analytics accelerator that inherently supports the sparse graph data representations using an in-situ compute-enabled crossbar memory architectures. Gaas-X reduces the overhead of redundant writing, sparse to dense conversion and redundant computing on invalid edges, which exists in the cross switch based PIM accelerator. GaaS-X achieves 7.7× and 2.4× performance and 22× and 5.7×, energy savings, respectively, over two state-of-the-art crossbar accelerators and offers orders of magnitude improvements over GPU and CPU solutions. Method-2 is the Depgraph [3], Method 2 is similar to method 1. Both are graph analysis accelerators, but Depgraph solves the problem of long dependency chain of graph calculation from the algorithm level, and Gaas-X solves it from the data structure level. Depgraph is coupled with the core architecture of multi-core processors, which can fundamentally alleviate the challenge of dependency on faster state propagation. Specifically, an effective dependency driven asynchronous execution method is proposed for new microarchitecture design to propagate the state faster. Depgraph expects the core vertices in real time along the dependency chain between the vertex state and the new state of the active vertex. The purpose is to effectively accelerate the propagation of the new state of the active vertex and ensure better data locality. Both method 1 and method 2 are hardware accelerators. Method 3 and method 4 solve the problem of graph calculation from the perspective of software. Method-3 is Analysis of lateral extended graphics using asynchronous block coordinate descent. BCD view provides key insights and tradeoffs for achieving high convergence speed of iterative graph algorithm. Under the algorithm design option suggested by BCD, GraphABCD [4] has the characteristics of fast convergence. GraphABCD provides algorithmic and architectural support for asynchronous execution without compromising its fast convergence characteristics. With minimal synchronization overhead, GraphABCD can be efficiently extended to heterogeneous and distributed accelerators. Method-4 is a method of training U-Net model on large-scale image data set [5]. To address the issue of training large images, a sample parallel U-Net training algorithm that exhibits near linear speedup on 1534 GPUs was presented. To overcome the memory limitations of a single GPU to train large U-Net mpdels, a pipeline-parallel training framework called TorchGPipe that implements the GPipe framework in PyTorch was evaluated for its baseline performance on a single Summit node (6 GPUs). Pipeline parallelism was shown to further boost the training speed by two to three fold while supporting deeper models with 4 larger receptive field and an order of magnitude more training parameters than the standard U-Net models reported in the literature.

In conclusion, the problem of long dependency chain in graph calculation can be solved from two aspects of software and hardware. Software accelerators can achieve high

performance of graphics processing, but they sacrifice the programmability and low entry cost provided by general-purpose processors. The hardware accelerator is designed to be integrated into the general multi-core processor, so that its core can support high-performance graphics processing by ensuring faster convergence speed and better data locality.

References:

[1] N. Jao, A. K. Ramanathan, A. Sengupta, J. Sampson, and V. Narayanan, "Programmable Non-Volatile Memory Design Featuring Reconfigurable In-Memory Operations," in 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019, pp. 1–5.

[2] Challapalle N, Rampalli S, Song L, et al. GaaS-X: Graph analytics accelerator supporting sparse data representation using crossbar architectures[C]//2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020: 433-445.

[3] Zhang Y, Liao X, Jin H, et al. DepGraph: A Dependency-Driven Accelerator for Efficient Iterative Graph Processing[C]//2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). IEEE, 2021: 371-384.

[4] Yang Y, Li Z, Deng Y, et al. GraphABCD: Scaling out graph analytics with asynchronous block coordinate descent[C]//2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2020: 419-432.

[5] Seal S K, Lim S H, Wang D, et al. Toward Large-Scale Image Segmentation on Summit[C]//49th International Conference on Parallel Processing-ICPP. 2020: 1-11.