2023 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT) | 979-8-3503-4254-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/PACT58117.2023.00041

SparseFT: Sparsity-aware Fault Tolerance for Reliable CNN Inference on GPUs

Gwangeun Byeon[1], Seungtae Lee[2], Seongwook Kim[1], Yongjun Kim[3], Prashant J. Nair[4], Seokin Hong[1]

Department of Electrical and Computer Engineering, Sungkyunkwan University[1]
Department of AI System Engineering, Sungkyunkwan University[2]
Samsung Electronics[3]
The University of British Columbia[4]

[1,2]{kebyun, dltmd1507, su8939, seokin}@skku.edu
[3]{yjuun.kim}@samsung.com
[4]{prashantnair}@ece.ubc.ca

***Abstract*—Graphics Processing Units (GPUs), while offering exceptional performance for CNN inference tasks, are susceptible to both transient and permanent hardware faults due to the integration of numerous processing elements and advancements in technology scaling. This paper proposes a novel and cost-effective fault mitigation technique, called *Sparsity-aware Fault Tolerance (SparseFT)*, to ensure reliable CNN inference on GPUs. SparseFT leverages inherent sparsity in the activation maps to detect and correct errors on the processing elements without hardware redundancy. By exploiting the characteristic of dot-products, where multiplications with zero operands are ineffectual, SparseFT dynamically duplicates an effectual computation (i.e., a multiplication with non-zero operands) to the processing element initially assigned to the ineffectual one. It then compares the duplicated computation results to detect errors. Experimental results demonstrate that SparseFT achieves more than 97% error detection coverage with less than 1% performance overhead for the state-of-the-art CNN models.**

***Index Terms*—Reliability, CNN, GPU, Sparsity**

I. Introduction

The rapid development of Deep Neural Networks (DNNs) has made them essential for safety-critical applications, such as self-driving cars and autonomous drones. These applications often rely on Graphics Processing Units (GPUs) for executing DNNs, as GPUs offer high-performance capabilities. In addition to performance and versatility, safety-critical applications also require a high degree of hardware resiliency [1]. Unfortunately, ensuring compute resiliency for GPUs is a complex task, often involving significant area and power overheads due to redundant computations such as time and hardware redundancy [2].

To overcome this problem, this paper proposes ***Sparsity-aware Fault Tolerance (SparseFT)***, a novel fault mitigation technique without duplicating hardware and/or instructions. To this end, SparseFT exploits the insight that the activation and weight maps are sparse (i.e., zero valued) in CNN models. The activation map includes many zero values as the activation function, such as Rectified Linear Unit (ReLU), converts negative convolution outputs into zero. The weight maps can also be sparse with pruning techniques that remove unimportant weights.

Due to large populations of zero values in activation and weight maps, some threads may become ineffectual in the warp. For example, when performing the dot-product in a CONV layer, all threads within a warp execute the FMA (Fused-Multiply and Add) instruction, whose operation is expressed as C = C + A × B. If operand A or B of some threads is zero, the result C is not updated, rendering them ineffectual. A typical GPU architecture does not allow the operation of a thread within a warp to be skipped, even if the thread is ineffectual. As a result, a warp with several ineffectual threads can cause the processing elements to be underutilized. Motivated by these observations, this paper improves GPU reliability by repurposing ineffectual threads as DMR.

Our experimental results show that the error detection coverage averages 66% across all popular CNN models when using only activation sparsity. By exploiting the sparsity in both activation and weight, the error detection coverage of SparseFT reaches around 99% for all popular CNN models.

II. Error Detection

SparseFT orchestrates a DMR-like execution dynamically by duplicating the operands of an effectual thread into one of the ineffectual threads within a warp. Since threads in a warp execute the same instruction, they will produce identical

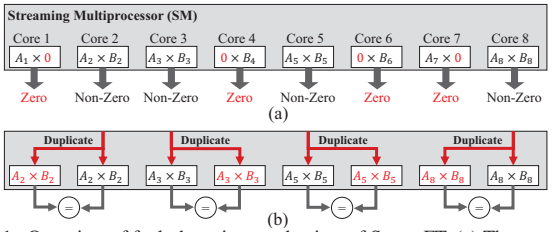

Fig. 1. Overview of fault detection mechanism of SparseFT. (a) The operation of processing elements in an SM without fault detection. (b) The operation of processing elements in an SM with fault detection.

This work was performed when Yongjun Kim was affiliated with Sungkyunkwan University

979-8-3503-4254-3/23/$31.00 ©2023 IEEE
DOI 10.1109/PACT58117.2023.00041

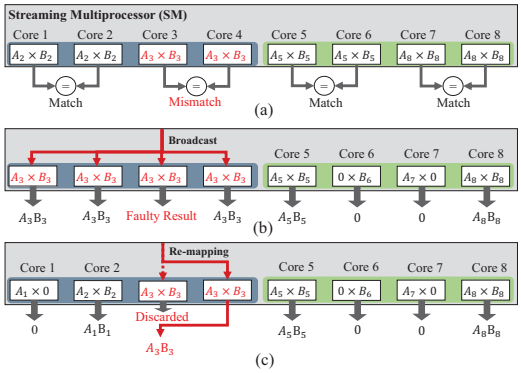

Fig. 2. Error correction and fault isolation. In this example, we assume there are two voting group in an SM, each containing four cores. (a) An Error is detected in the computation on cores 3 and 4. (b) An erroneous computation is broadcasted to all cores of a voting group to determine a faulty core. (c) A faulty core (core 3) is isolated by re-mapping an effectual thread to a non-faulty core initially mapped to an ineffectual thread.

computation results as long as they use the same operands. If an error occurs during the thread execution, the two threads may produce different results for the same operands, which enables SparseFT to detect errors.

Figure 1 shows an example of SparseFT applied to an SM with eight cores. In this example, each thread running on a core performs the multiplication operation. Among the eight threads, four of them are ineffectual because of sparsity (Figure 1-(a)). Figure 1-(b) illustrates a dynamically constructed DMR execution in which operands of effectual threads are duplicated to their adjacent ineffectual thread. For example, a thread running on core 1 uses the non-zero operands of its adjacent thread running on core 2. Then, the two threads mapped to the core 1 and 2 execute the same instruction (i.e., multiplication) with the same operands. Thus, by comparing the results of the duplicated computations on two cores, SparseFT can detect errors at run-time.

III. Error Correction and Fault Isolation

If an error is detected during the error detection phase, SparseFT re-executes the computation on a DMR core-pair in a lock-step fashion to correct an error. If the computation results do not match again, the error may be caused by a permanent fault. In this case, SparseFT broadcasts the computation into more than three cores to identify a faulty core with a voting mechanism. After identifying a faulty core, SparseFT isolates it by re-mapping the effectual computation initially mapped to the faulty core into a non-faulty one.

In Figure 2-(a), the core-pairs (1, 2), (3, 4), (5, 6), and (7, 8) perform the multiplication with a DMR-like execution for error detection. As the computation result of core-pair (3, 4) does not match, an error may occur in one of those cores. To diagnose which core is faulty, SparseFT broadcasts the erroneous computation to all cores of the voting group (Fig 2-(b)). In the example, cores 1, 2, 3, and 4 are the same voting group and execute an instruction with the same operands. If there is no faulty core, all results of the cores are the same. However, the result of core 3 differs from other cores in the example. Thus, SparseFT concludes that core 3 is faulty.

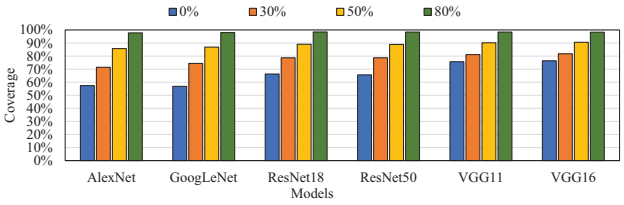

Fig. 3. Error detection coverage for CONV layers of CNN models with difference weight pruning ratio.

Figure 2-(c) shows an example of fault isolation with sparsity-aware computation re-mapping of the SparseFT. In this example, SparseFT re-mapped the faulty core(core 3) to the adjacent core(core 4) initially assigned to ineffectual computation and non-faulty to isolate the faulty core.

IV. Evaluation

To evaluate error detection coverage, we collect information about the instruction's operands using NVBit [3] for the CUDA core. We define error detection coverage as the percentage of multiplication instructions that SparseFT can protect. We compare the error detection coverage for the CNN models pruned with different target pruning rate.

Figure 3 shows the error detection coverage of SpraseFT for four different pruning rates, 0/30/50/80%. The error detection coverage increases proportionally as the target pruning rate increases. This is because, as the weight pruning rate increases, ineffectual thread in the warp also increases due to weight sparsity. Also, error detection coverage is higher in the latter layers compared to the early layers. This is because the activation sparsity increases in the later layers. As a result, the average error detection coverage of SparseFT for the three CNN models with pruning rates 0/30/50/80% are 66%, 77%, 88%, and 97%, respectively.

Acknowledgment

This work was partly supported by the National Research Foundation of Korea (NRF) grant (No. 2022R1C1C1012154, 80%), the Ministry of Science and ICT (MSIT) under the Information Technology Research Center (ITRC) support program (IITP-2021-0-02052, 10%) supervised by the Institute for Information & Communications Technology Planning & Evaluation (IITP), and Samsung Electronics Co., Ltd (IO220902-02337-01, 10%). The corresponding author of this paper is Seokin Hong.

References

[1] F. F. dos Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and increasing the reliability of convolutional neural networks on gpus," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, 2018.

[2] S. Hong and S. Kim, "A low-cost mechanism exploiting narrow-width values for tolerating hard faults in alu," *IEEE transactions on computers*, vol. 64, no. 9, pp. 2433–2446, 2014.

[3] O. Villa, M. Stephenson, D. Nellans, and S. W. Keckler, "Nvbit: A dynamic binary instrumentation framework for nvidia gpus," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2019, pp. 372–383.