2024 13th Non-Volatile Memory Systems and Applications Symposium (NVMSA) | 979-8-3503-8855-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/NVMSA63038.2024.10693658

Evaluation of Erasure Coding and Opportunistic Offloading Algorithms using DPU in Distributed Storage Systems

Junghyun Ryu*, Hongsu Byun*, Myungcheol Lee†, Jinchun Choi†, Youngjae Kim*‡
*Dept. of Computer Science and Engineering, Sogang University, Seoul, South Korea
†Electronics and Telecommunications Research Institute, Daejeon, South Korea

***Abstract*—In this paper, we explore accelerating Erasure Coding by applying the BlueField-3 DPU to distributed storage systems. First, we evaluated the performance of Erasure Coding when executed on the CPU and DPU, respectively. The evaluation results showed that the DPU had a execution time approximately 10.5 times slower than the CPU. This finding indicates that unconditional offloading is not beneficial; instead, offloading should be considered opportunistically when the CPU is busy and not be able to handle Erasure Coding tasks effectively. Based on this, we propose an opportunistic algorithm for offloading Erasure Coding to the DPU. Our future work is to refine the proposed offloading algorithm and apply it to distributed storage systems using DPUs to improve the performance of Erasure Coding.**

I. Introduction & Background

The Data Processing Unit (DPU) is a solution designed to optimize computing resources and accelerate data processing tasks within servers [1]. DPUs enhance overall system performance by performing data processing tasks independently from the CPU or GPU. This offers significant advantages, especially for large-scale data processing tasks such as AI, machine learning, and big data analytics [2]. DPUs can accelerate specific tasks using its hardware accelerators, such as Erasure Coding, encryption, and compression [3]. Figure 1 illustrates the overall architecture of the NVIDIA BlueField-3 DPU. The hardware Erasure Coding accelerator can be controlled via the DOCA (Data Center-on-a-Chip Architecture) interface from the host [4]. DOCA is a software framework developed by NVIDIA aimed at maximizing the performance, security, and efficiency of data center infrastructure. DOCA provides a suite of APIs, libraries, and tools to leverage the full capabilities of NVIDIA BlueField DPUs. Through DOCA, it is possible to offload, accelerate, and isolate data center services, enhancing overall system performance and security.

Distributed storage systems like Hadoop File System (HDFS) and Ceph are highly scalable and durable storage systems designed to handle large-scale data [5]. These systems provide essential infrastructure for big data analytics, cloud computing, and AI applications. The replica storage method commonly adopted to ensure reliability in distributed storage systems is inefficient for processing large volumes of data due to its additional storage consumption. Consequently, approaches have been made to implement Erasure Coding

‡Y. Kim is the corresponding author.

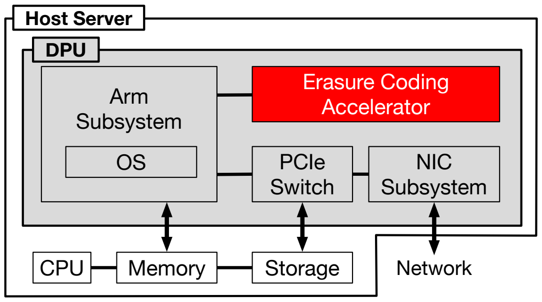

Fig. 1. An architecture of BlueField-3 DPU[3].

based on Reed-Solomon codes to achieve space efficiency and build reliable storage systems [6]. For instance, in HDFS using (6, 3) Erasure Coding can save approximately 60% of storage capacity compared to the 3× replication method [7]. However, Erasure Coding is a computation-intensive task based on matrix multiplication, which introduces additional overhead. We aim to reduce the workload on the CPU by offloading computation-intensive tasks, Erasure Coding, to the DPU to mitigate the computational overhead in distributed storage systems. There is little research ever done on Erasure Coding with DPU; this is the first work to study the offloading possibilities and utilize DPU for such purposes. We evaluated Erasure Coding performance on both the CPU and DPU. The results showed that the DPU had a execution time approximately 10.5× slower than the CPU. Based on the results, we can see that there is no benefit to simply offloading all Erasure Coding tasks to the DPU. Therefore, we propose an opportunistic Erasure Coding offloading algorithm that dynamically offloads Erasure Coding tasks based on the state of the CPU to efficiently utilize computational resources on the CPU and DPU.

II. Design & Preliminary result

Experiment Setup. To evaluate the Erasure Coding performance of the CPU and DPU, we measured the Erasure Coding execution time based on different file sizes. The specifications of the host server and the Bluefield-3 DPU are shown in Table I. The parameters for the Erasure Coding were set to (128, 32), where 128 represents the number of data chunks and 32 represents the number of parity chunks. We assessed Erasure Coding on two systems:

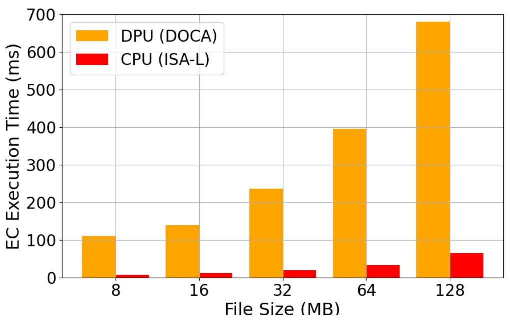

Fig. 2. Erasure Coding execution time comparison: Host CPU vs. DPU.

- **CPU:** The host CPU performs Erasure Coding on data loaded into the host memory and then stores the results in the host's SSD. The Erasure Coding library used was Intel's ISA-L [8]. Erasure Coding task was executed by binding it to a single CPU core.
- **DPU:** DPU performs Erasure Coding using the Erasure Coding accelerator of the BlueField-3 on data loaded into the host memory and then stores the results in the host's SSD. DOCA was used for the execution of Erasure Coding.

Results. Figure 2 shows the differences in Erasure Coding execution times between the CPU and the DPU. Encoding was performed on files of 8MB, 16MB, 32MB, 64MB, and 128MB. The DPU was $13.3\times$ slower than the CPU for an 8MB file, and $10.5\times$ slower for a 128MB file. This finding indicates that unconditional offloading is not beneficial; instead, offloading should be considered opportunistically when the CPU is busy and not be able to handle Erasure Coding tasks effectively. There is a trade-off between freeing host resources and Erasure Coding execution time. Therefore, we suggest two approaches: 1) Given the inefficiency of simply offloading to the DPU Erasure Coding accelerator, Erasure Coding should be offloaded opportunistically when the host CPU is busy; 2) In addition to utilizing the DPU Erasure Coding accelerator, it is possible to fully leverage the ARM cores of the DPU for a hybrid hardware/software approach, offloading opportunistically as in the first approach. In this paper we explore and propose only the first approach.

Offloading Algorithm using DPU. Algorithm 1 illustrates the preliminary design of offloading algorithm for determining the execution location(CPU or DPU) of Erasure Coding. The $ECPollingThread$ function polls the queue where Erasure Coding requests arrive and decides the execution location based on the situation. If the CPU is busy, the Erasure Coding execution is offloaded to the DPU. Otherwise, it operates in one of three modes based on the admin-configurable status. ❶ In latency-sensitive scenarios, Erasure Coding is executed on the CPU to achieve the fastest response. ❷ If not, the DPU is asynchronously requested to perform Erasure Coding, allowing the Host Server to focus on other tasks. ❸ Lastly, hybrid execution can be requested, where α and β represent the number of Erasure Coding tasks executed by the CPU and DPU, respectively, with their values dependent on system performance.

TABLE I
TESTBED SPECIFICATIONS.

Type	Host Server	BlueField-3
CPU	Intel Xeon Silver 4410Y 48-Core, 3.9GHz	Armv8.2 + A78 Hercules 16-Core, 2.0GHz
Memory	DDR4, 16 GB * 4 (=64 GB)	DDR5, 32 GB (On-board)
OS	Ubuntu 22.04.4	Ubuntu 22.04.3

Algorithm 1 Resource-aware Erasure Coding Offloading

```
1:  $Queue_{ECRequest} \leftarrow$ queue where EC requests arrive
2:  function ECPollingThread
3:      while True do
4:          if isCpuBusy then
5:              // EC offloaded to DPU
6:              $Queue_{ECRequest}.dequeue() \rightarrow DPU$
7:          else
8:              switch $status$ do
9:                  case $isLatenySensitive$
10:                     $Queue_{ECRequest}.dequeue() \rightarrow CPU$
11:                 case $isAsyncECAllowed$
12:                     $Queue_{ECRequest}.dequeue() \rightarrow DPU$
13:                 case $isHybrid$
14:                     // $\alpha$ and $\beta$ represent the number of EC tasks
15:                     $Queue_{ECRequest}.dequeue(\alpha) \rightarrow CPU$
16:                     $Queue_{ECRequest}.dequeue(\beta) \rightarrow DPU$
```

III. Conclusion

To explore the utility of DPUs in distributed storage systems, we investigated offloading Erasure Coding tasks to the DPU. Experimental results showed that executing Erasure Coding on the DPU through DOCA increased the execution time by approximately $10.5\times$ compared to the host server. Therefore, we propose an opportunistic offloading algorithm in the form of pseudo-code to efficiently utilize all computational resources.

Acknowledgment

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2021-0-00136, Development of Big Blockchain Data Highly Scalable Distributed Storage Technology for Increased Applications in Various Industries).

References

[1] "What is a dpu data processing unit." https://blogs.nvidia.com/blog/whats-a-dpu-data-processing-unit/, 2020.
[2] X. Wei, R. Cheng, Y. Yang, R. Chen, and H. Chen, "Characterizing off-path smartnic for accelerating distributed systems," in *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, pp. 987–1004, 2023.
[3] "NVIDIA DOCA Overview." https://docs.nvidia.com/doca/sdk/nvidia+doca+overview/index.html, 2024.
[4] "DOCA erasure coding." https://docs.nvidia.com/doca/sdk/doca+erasure+coding/index.html, 2024.
[5] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
[6] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.
[7] "Evaluation erasure coding hadoop 3." https://db-blog.web.cern.ch/blog/emil-kleszcz/2019-10-evaluation-erasure-coding-hadoop-3, 2019.
[8] Intel Corporation, "ISA-L: Intelligent Storage Acceleration Library." https://github.com/intel/isa-l, 2015. Accessed: 2024-07-01.