

Characterizing Lossy and Lossless Compression on Emerging BlueField DPU Architectures

Yuke Li[†], Arjun Kashyap[†], Yanfei Guo[‡], and Xiaoyi Lu^{†*}

[†]University of California, Merced, Merced, CA, USA
 {yli304, akashyap5, xiaoyi.lu}@ucmerced.edu

[‡]Argonne National Laboratory, Lemont, IL, USA
 yguo@anl.gov

Abstract—The Data Processing Unit (DPU) (i.e., programmable SmartNICs with System-on-Chip or SoC cores) has emerged as a valuable supplementary resource to the host CPU. The DPU architecture has been attracting significant attention within High-Performance Computing (HPC) and data center clusters due to its advanced capabilities and accelerators, which include a hardware-based data compression engine. This positions the DPU as a prospective tool for accelerating and offloading compression workloads from the hosts, which can potentially speed up data-intensive applications. The convergence of Big Data, HPC, and Machine Learning (ML) systems has rendered large data volumes a major performance bottleneck in message communication and data storage. While compression can boost performance, recent studies reveal that compression techniques (e.g., lossy and lossless) are compute-intensive and time-consuming, particularly with larger data sizes. Consequently, this paper characterizes the performance of three lossy (SZ3) and lossless (DEFLATE and zlib) compression algorithms with seven real-world data sets on the popular NVIDIA’s BlueField DPUs to explore potential opportunities for offloading these workloads from the host. We find that compared to DPU’s SoC cores, DPU’s hardware compression engine can obtain up to 26.8x performance speedup. Furthermore, we discuss the challenges and opportunities associated with employing NVIDIA’s BlueField DPUs to accelerate lossy and lossless compression/decompression workloads. Our research discloses five important takeaways which shed light on future research directions for lossy and lossless compressions on DPUs.

Index Terms—DPU, Lossy and Lossless Compression, Performance Characterization, BlueField

I. INTRODUCTION

Due to the convergence of Big Data, High-Performance Computing (HPC), and Machine Learning (ML) technologies, the data volume in communication and storage has risen to be the major performance bottleneck for applications trying to perform efficient analysis and draw meaningful insights. For instance, prior works [1]–[4] have shown that distributed deep neural network training suffers from data movement bottlenecks. Similarly, research [5]–[7] has tried to accelerate the communication-intensive Big Data frameworks to alleviate performance bottlenecks caused by moving large volumes of data. Hence, reducing the data movement for HPC and machine learning workloads is an important problem in the research community.

Data compression is a common operation performed by many applications to help with data size reduction. It involves encoding data to decrease the original data size for storage or transmission purposes. Popular applications that employ compression techniques (lossless and lossy) [8]–[12] to enhance the performance are

machine learning [2]–[4], [13], databases [14], [15], and network communication [16], [17]. For example, the central idea of [2]–[4], [13] revolves around compressing gradients during large-scale distributed training to reduce the volume of data to be transferred between the nodes and remove data movement bottlenecks. Thus, compression plays an important role in diverse applications and is still a hot research problem in this era of Big Data.

However, recent studies [8], [9], [18] have shown that both lossy and lossless compression techniques are paramount in high-speed scientific data compression. Still, they are compute-intensive and time-consuming, especially with increasing data sizes. To expedite the performance of compression and decompression schemes, many solutions [8]–[12], [18] have been proposed in the literature with various focuses. One promising approach is exploiting software-hardware codesigns to significantly improve the performance of state-of-the-art compression schemes. To the best of our knowledge, most (if not all) of these designs usually only support common processors or platforms such as x86 CPUs, GPUs, FPGAs, Arm CPUs, etc., for a specific type of compression and decompression scheme like lossy and lossless.

Data Processing Unit (DPU) has recently become popular and adds computing power to a regular Network Interface Card (NIC) in data centers and HPC systems. Extra computing units inside a DPU make it suitable for offloading tasks from the host CPU. Major network vendors have released their programmable NICs with different architectures like NVIDIA’s BlueField [19], Fungible DPU [20], Marvell’s LiquidIO [21], Intel’s Infrastructure Processing Unit (IPU) [22], Broadcom’s Stingray [23] SmartNICs, Xilinx’s Alveo [24], and xPU in the SNIA community [25]. These emerging DPU devices provide unprecedented opportunities to offload compression and decompression tasks from busy-occupied host CPUs, benefiting many HPC and data-intensive applications. Notably, some of these DPUs (like NVIDIA BlueField) also provide hardware-based accelerators onboard to further speed up the performance of compression and decompression.

Despite the emergence of new computing devices such as DPUs in modern HPC or data center systems, our literature survey reveals a significant gap in the systematic study of performance characteristics for DPU-accelerated lossless and lossy compression and decompression schemes. Additionally, leveraging DPU with its SmartNIC capabilities enables the potential for offloading resource-intensive compression and decompression tasks from the host, leading to resource savings, reduced communication payload, and accelerated

* is the corresponding author.

applications in communication and storage scenarios. These observations hence motivate us to explore the performance characteristics of DPU-based compression and decompression schemes and workloads, intending to identify potential avenues for future research.

This paper addresses several challenges in the performance characterization of compression and decompression workloads on emerging BlueField DPUs. To comprehend the goals we introduced above, several challenges arise:

- ① How to select appropriate dimensions to comprehensively evaluate the performance of compression/decompression workloads on both SoC cores and the compression engine on the DPU;
- ② How to interpret the capabilities of the SoC cores and the compression engine in processing compression/decompression workloads through the analysis of results obtained from the characterization dimensions;
- ③ How to identify potential optimization opportunities and summarize them into guidelines for future work is a key objective, providing valuable insights for further enhancing the performance of compression and decompression workloads.

We have introduced a three-dimensional characterization methodology, encompassing hardware, dataset, and algorithm dimensions, to systematically study the performance of NVIDIA BlueField DPUs. Our key contributions are as follows.

- We perform an extensive performance study of NVIDIA BlueField DPUs for three lossless (e.g., DEFLATE [12] and zlib [26]) and lossy (e.g., SZ3 [8]) compression and decompression schemes, and identify their capabilities and limitations.
- We give deep insights into the BlueField DPU compression engine, including the performance bottlenecks and optimization opportunities. We find the compression engine on the BlueField-2 can achieve 26.8 times faster than running the compression workload on the SoC cores, while including the 90.4% significant overheads on data staging and initialization.
- We conduct a brief evaluation and comparison of BlueField-2 v.s. BlueField-3, finding the largest performance gap over 25%.
- Through the detailed characterization of DPU for compression/decompression, we give several design guidelines for researchers.

II. BACKGROUND

This section provides an overview of the essential technical background that underpins our subsequent discussions and evaluations. It begins by delving into the architectures of the NVIDIA BlueField-2 and BlueField-3 Data Processing Units (DPUs), as shown in Table I, providing insights into their distinctive features, operational modes, and hardware capabilities, particularly their roles in computation and handling compression/decompression tasks. This exploration illustrates their growing potential within High-Performance Computing (HPC) clusters and data centers. Subsequently, the focus shifts to various compression workloads, distinguishing between lossy and lossless compression techniques and their respective application areas, including their fundamental characteristics, strengths, and drawbacks.

A. NVIDIA BlueField-2/3 DPU

NVIDIA BlueField-2: DPUs or SmartNICs are programmable Network Interface Cards (NICs) that can aid servers in offloading

some of their computation. NVIDIA’s BlueField DPUs, for example, the BlueField-2 shown in Figure 1, are equipped with general-purpose System-on-Chip (SoC) cores that can run their own operating system (OS). BlueField-2 [19] also contains hardware accelerators for various operations like compression/decompression, hashing, regular expression matching, and random number generation.

NVIDIA’s DPU generally supports two modes of operation (1) Separated Host Mode: on-board computing units act as a separated and orthogonal host like any other external host, and (2) Embedded CPU Function Mode (or SmartNIC mode): onboard computing units control the NIC resources and data path. The Separated Host Mode makes DPU act like an independent server with its networking ports separate from the host. The host and ARM subsystem have their own MAC address, and the DPU ARM cores have no visibility or control over the network traffic flowing to/from the host. Thus, this mode in NVIDIA’s BlueField DPU is analogous to the functionality of off-path SmartNICs, where traffic to/from the host bypasses SmartNIC cores. On the other hand, in the SmartNIC mode in BlueField-2, all network traffic to/from the host is transmitted via a virtual switch running on ARM cores. Hence, only the SmartNIC mode of NVIDIA DPU allows custom operation on a network packet, provided the network flow rules are not offloaded to the hardware (embedded switch). In this way, the SmartNIC mode of NVIDIA DPU turns it into an on-path SmartNIC since the ARM subsystem inside the DPU can manipulate network packets during transmission/receive.

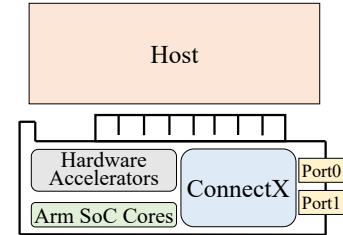


Fig. 1: BlueField-2/3 Architecture

NVIDIA BlueField-3: Recently, NVIDIA BlueField-3 [27] DPUs are becoming available in HPC clusters and datacenter environments. Besides having the same operation modes as BlueField-2, BlueField-3 differs in many other ways. Table I shows the general hardware difference between BlueField-2 and BlueField-3. In terms of networking capabilities, both BlueField-2 and BlueField-3 can support Ethernet and InfiniBand ports with BlueField-2 supporting up to 200 Gb/s, whereas BlueField-3 supports up to 400 Gb/s connectivity. The SoC cores that allow general-purpose computing are based on a 64-bit instruction set, with BlueField-2 holding up to 8 ARM Cortex-A72 cores while BlueField-3 holds up to 16 ARM Cortex-A78 cores. Regarding the availability of hardware accelerators, BlueField-3 is equipped with additional accelerators compared to BlueField-2. For example, some additional accelerators on BlueField-3 are for erasure coding, software-defined networking, GPUDirect, and all-to-all collective communication. The BlueField-2 can support up to 32GB DDR4 memory whereas the BlueField-3 supports up to 16GB DRAM but with newer DDR5 memory standard.

	SoC Cores	Interconnects	Max. BW	Network Adapter	Memory	PCIe Interface
BF-2	8× Arm Cortex-A72	Ethernet, InfiniBand	200Gb/s	NVIDIA ConnectX-6 Dx	16GB / 32GB on-board DDR4	8 or 16 lanes PCIe Gen 4.0
BF-3	16× Arm Cortex-A78AE	Ethernet, InfiniBand	400Gb/s	NVIDIA ConnectX-7	16GB on-board DDR5	32 lanes PCIe 5.0

TABLE I: Hardware Specifications of BlueField-2 and BlueField-3

B. Compression

Data reduction via compression is becoming increasingly important when large-scale HPC applications generate huge volumes of data. Broadly, the community primarily uses two compression techniques for scientific data size reduction – lossy and lossless.

(1) The lossy compression means compressing data with losing some data to achieve a higher compression ratio. Compared to lossless compression, lossy compression cannot rebuild the original data during decompression. Lossy compression is important since it offers the flexibility to trade off data quality for high compression ratios and helps reduce the size of scientific data while guaranteeing quantifiable error bounds [8]. To control the amount of distortion during lossy compression, error-bounded lossy compressors [8], [9], [28], [29] have been employed.

(2) The lossless compression technique, on the other hand, can reproduce the exact match of the compressed data during decompression. Since a lossless compression mechanism needs to preserve the integrity of the original data, it can only achieve low compression ratios. However, it is still actively being used in scenarios where scientific datasets store single- or double-precision floating-point format [30], checkpoint data, and data from which certain derived quantities will be computed [31]. Some state-of-the-art lossless compressors are GZIP [10], zstd [11], FPC [31], SPDP [30], and Blosc [32].

III. CHARACTERIZATION METHODOLOGY

This section introduces our characterization methodology, as illustrated in Figure 2. The subsequent sub-sections will provide a detailed description of each dimension. Furthermore, we will report the testbed configuration at the end of this section.

A. Three-Dimensional Characterization Methodology

1) *Compression Designs*: We selected four popular and well-used lossless and lossy compression designs, as described in Table II. These designs showcase a variety of approaches to data compression, offering an ample spectrum for our investigations.

Algorithm	Purpose	Lossless	Lossy	SoC cores	Compression engine
DEFLATE	general data compression	✓		✓	✓
zlib		✓		✓	
SZ3	scientific data compression		✓	✓	

TABLE II: Compression designs and features.

We propose a comparative analysis between the general-purpose SoC cores and the specialized compression engine by assessing the performance of compression and decompression workloads. The performance characterization is two-fold. First, we test DEFLATE, zlib, and SZ3 compression designs with seven datasets on the System-on-Chip (SoC) ARM cores of the BlueField-2 DPU. Second, we leverage the in-built Compression Accelerator of BlueField-2 for DEFLATE compression design. Notably, DEFLATE exhibits

minor variations in compression ratio with the same dataset due to differing implementation configurations on SoC and engine, such as the window size for the DEFLATE algorithm. We adjusted these configurations to align the compression ratios more closely.

For lossy compression, we choose SZ3 with error bound $1E-3$, which is the default configuration used in SZ3, because it is a well-maintained lossy compression for scientific datasets. Due to the algorithm limitation on the compression engine of BlueField-2, we only use DEFLATE via DOCA SDK on the compression engine, which is also the default choice to run on SoC cores of BF2. Furthermore, zlib compression is designed based on the DEFLATE algorithm while including additional header and tailer data to enhance the functionality of the DEFLATE compression algorithm.

2) *Datasets*: Alongside our methodology, we’ve chosen seven representative HPC datasets reported from [31], [33] with single-precision floating points and various information to study the performance of compression and decompression on BlueField-2, presented in Table III. It is important to note that we had to shrink some datasets, for example, `msg_sppm`, to 128 MB while keeping the data distribution the same to fit the maximum data chunk limitation inside a compression job for utilizing the compression engine of BlueField-2.

Dataset	Size (MB)	Unique Vals (%)	Entropy (bits)	Randomness (%)	Description of the data
obs_info	9.1	23.9	18.07	94.5	Scientific instruments
obs_error	30	18	17.8	87.2	
msg_sweep3d	60	89.8	23.41	98.6	Message sent by a node in parallel applications
msg_lu	93	99.2	24.47	99.8	
msg_bt	128	92.9	23.67	95.1	
msg_sppm	128	10.2	11.24	51.6	
num_plasma	17	0.3	13.65	99.4	Numeric simulation

TABLE III: The chosen seven HPC datasets with various statistical information in size, unique values, entropy, and randomness.

The selected datasets exhibit variability in size and other aspects, such as the percentage of unique values and randomness, as detailed in the paper [31]. This diversity in dataset characteristics is crucial for ensuring the comprehensiveness and applicability of our findings to real-world scenarios, as different datasets may exhibit distinct behaviors when subjected to compression.

3) *Hardware*: NVIDIA BlueField-2 is widely recognized as the pioneering data center infrastructure-on-a-chip solution and has gained extensive adoption among enterprises. BlueField-3, the latest DPU released by NVIDIA, brings substantial hardware improvements compared to its predecessor. BlueField DPUs offer more than just general-purpose Arm SoC cores; they also provide hardware accelerations such as the specific compression engine. Consequently, BlueField DPUs offer users more options for executing their workloads.

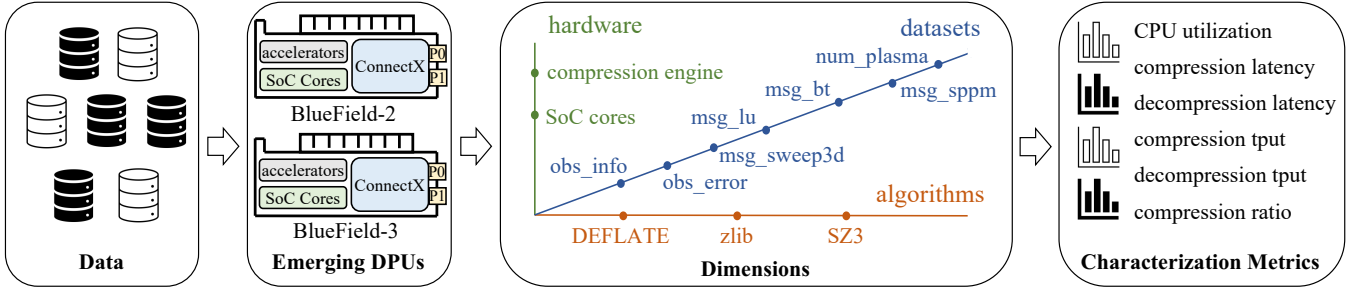


Fig. 2: Overview of Characterization Methodology.

B. Performance Metrics

To conduct a comprehensive performance characterization of BlueField DPU architectures, we executed four compression designs on seven datasets, recording metrics related to compression/decompression results and hardware utilization. This approach allows us to gather comprehensive data on both performance and hardware aspects of the BlueField DPU architectures.

We recorded the latency and throughput results regarding compression and decompression processing. Latency refers to the time a compression design takes to compress/decompress a dataset, while throughput represents the number of bytes a compression design can process per second. The compression ratio is calculated by dividing the original data size by the compressed data size.

Regarding hardware utilization, we recorded the average CPU utilization for each compression design with each dataset and expressed it as a percentage value.

Notably, to ensure fair characterization, we omitted the time required for loading/storing data to/from the disk files. The whole data was preloaded into the DPU's memory before the experiments.

C. Testbed

The evaluation for BlueField-2 was performed in our local setup which is equipped with NVIDIA BlueField-2 DPU, which consists of 8 ARM Cortex-A72 cores @2.75 GHz and 16 GB on-board DDR4 DRAM and runs Ubuntu 20.04.5 with DOCA SDK v1.5.0 [34]. The evaluation for Section IV-E was performed in HPC Advisory Council High-Performance Center's (HPCAC) [35] Thor cluster. The BlueField-2 on Thor is the same as the one on our local setup while it runs with Rocky Linux 8.6. The BlueField-3 on Thor has 16 ARM Cortex-A78 cores and 16 GB on-board DDR5 memory and runs with Rocky Linux 9.1. We run all experiments with BlueField-2/3 DPUs in the separated host mode.

IV. COMPRESSION PERFORMANCE CHARACTERIZATION

This section presents the thorough characterization of NVIDIA's BlueField-2 DPU regarding compression and decompression.

A. Characterize CPU usage on SoC Cores

Considering the relatively limited power of the SoC cores in BlueField-2 DPU when dealing with resource-intensive compression and decompression workloads, leveraging the dedicated hardware accelerator, the compression engine, presents a promising opportunity to enhance the efficiency and performance of these tasks. To validate this point, we analyze the CPU utilization

when executing compression and decompression workloads on the SoC cores and the compression engine.

Figure 3a depicts CPU utilization results from executing four distinct compression designs on seven representative datasets. When running lossless compression designs on SoC cores, it is noticeable that CPU utilization surpasses 80%, irrespective of dataset size, which ranges from 9.1M to 128M. The SZ3 lossy compression design exhibits even higher usage when executed on SoC cores, with near-full CPU utilization. However, when the compression engine is utilized with the lossless DEFLATE compression, CPU utilization drops significantly, reaching as low as 18% with the smallest 9.1M dataset and up to 53% with the biggest 128M dataset. Further observation shows that CPU utilization decreases with reduced dataset sizes when the compression engine is utilized. This suggests that using the compression accelerator reduces CPU load. The SoC cores continue to manage certain tasks, such as facilitating data movement between memory and the compression engine, and overseeing memory registration for the compression engine's usage.

Takeaway 1: Even with full compression processing executed on the compression engine, the SoC core's CPU utilization remains active at up to 53% for a dataset size of 128M. This is due to the core's involvement in tasks like data movement and memory management for the compression engine, despite offloading most of the workload.

B. Characterize Compression and Decompression Latency

Our observations indicate a decrease in CPU utilization when employing the compression engine. Our goal is to examine its performance impact on compression and decompression tasks, and quantify the advantages it provides compared to running these tasks exclusively on SoC cores.

Figure 3b and Figure 3c present the compression/decompression latency for four compression designs with seven distinct datasets.

In Figure 3b, the compression time is reported for four different compression designs using datasets of various statistical information. As we can see, utilizing the compression engine outperforms all other designs across all dataset sizes, achieving a remarkable speedup at an order of magnitude level. The speedup can be as high as 25× to 26.8× with datasets `msg_bt` and `msg_sppm` compared to the same ones executed on the SoC core. Even for the smallest 9.1M dataset `obs_info`, a compression engine can still achieve 3× of the one executed on the SoC cores. This indicates the

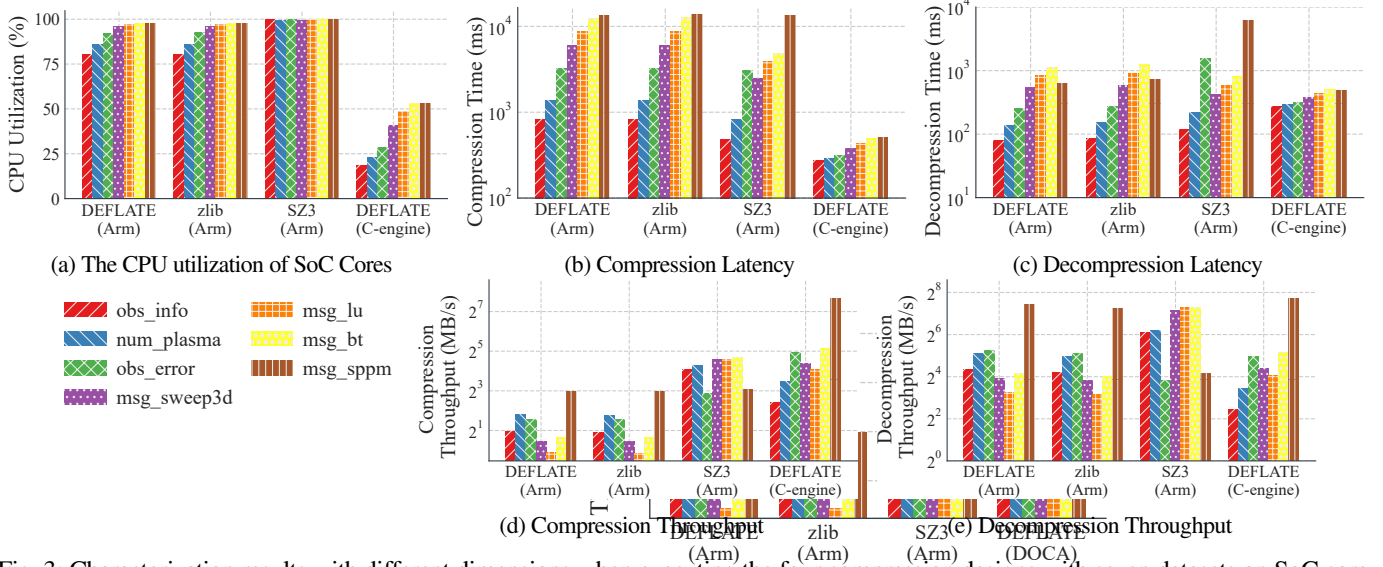


Fig. 3: Characterization results with different dimensions when executing the four compression designs with seven datasets on SoC core and compression engine of BlueField-2. The *C-engine* in the sub-figures is an abbreviation for *Compression engine*.

compression engine’s remarkable power and highlights its potential benefits for compression tasks.

However, as shown in Figure 3c, for datasets such as *obs_info*, *num_plasma*, and *obs_error* with sizes of 9.1 MB, 17 MB, and 30 MB (the first three bars in each lossless compression design), the decompression time of utilizing the compression engine degrades up to $3.5\times$ compared to decompressing the data on the SoC core. Nevertheless, we still observe the benefits of using the compression engine for datasets larger than 30 MB.

The reason for the performance degradation of utilizing a compression engine lies in the inherent overhead associated with processing data using the compression engine, specifically in two areas: 1) **data staging**, where the data must be transferred from memory to the compression engine (also referred to as the accelerator) and then back to memory, and 2) **buffer preparation**, which involves preparing the data buffer as a DOCA-specified object to utilize the accelerator. This necessitates the use of DOCA-specified mmap before any operation can be performed. Consequently, when the compression processing time is significantly larger than these overheads, the overall compression time using the accelerator can be shorter than compressions performed on the SoC cores. However, since decompression time is naturally much shorter than compression time, the reduced decompression time cannot mask the overhead of using the accelerator. Hence, the DEFLATE(DOCA)’s decompression time is greater than some others shown in Figure 3c.

Furthermore, we observed that the lossy compression algorithm SZ3 can achieve comparable compression and decompression times with the lossless compression designs for most datasets while keeping the high compression ratio, which will be discussed in Section IV-C. However, the decompression performance of SZ3 may vary significantly depending on the dataset characteristics.

Takeaway 2: Deploying compression workloads on the compression engine yields significantly shorter processing times, up to 26.8 times faster than running them on the SoC core. However, the overhead incurred by the compression engine cannot be ignored. For example, the compression engine is less performant for decompression than the SoC cores when the dataset size is relatively small (<60 M) to offset the associated overhead.

C. Characterize Compression and Decompression Throughput and Compression Ratio

Other than analyses of compression latency, it is essential to assess the efficiency of various compression designs on the SoC core and compression engine. Understanding the differences in efficiency will provide valuable insights into the performance and capabilities of these compression implementations, enabling reasonable decision-making for optimal compression strategies in the future. Figure 3d and Figure 3e present the compression and decompression throughput for four compression designs with distinct datasets.

Importantly, identical-sized datasets can display vast differences in compression throughput and ratio, even under the same algorithm, due to variations in data content and structure. Shorter latency doesn’t necessarily imply higher throughput, as it also depends on the volume and compressibility of the processed data.

We observed significant variations in compression and decompression throughput across different datasets. However, it is notable that running lossy compression designs on the SoC core or utilizing the compression engine generally leads to improved compression throughput. For instance, when considering the dataset *msg_sppm*, utilizing the compression engine with a lossless compression design achieves a throughput $25.6\times$ higher compared to executing it on the SoC core and achieves a throughput $24.1\times$ higher compared to with a lossy compression executing on the SoC core.

Due to its inherent characteristic of lossy compression, SZ3 consistently achieves a much higher compression ratio, as

demonstrated in Table IV, among all dataset sizes. That is one of the most significant features of the lossy compression algorithm – achieving a high compression ratio by sacrificing some accuracy.

However, when it comes to the decompression throughput, utilizing a compression engine, as depicted in Figure 3e, is not as efficient as the SoC-based designs for specific datasets. This is primarily attributed to the higher decompression time associated with the overheads incurred by the compression engine, as discussed in Section IV-B.

Dataset	DEFLATE (Arm)	zlib (Arm)	SZ3 (Arm)	DEFLATE (C-engine)
obs_info	1.211	1.211	9.694	1.196
num_plasma	1.387	1.387	22.785	1.24
obs_error	1.469	1.469	4.079	1.484
msg_sweep3d	1.159	1.159	83.831	1.15
msg_lu	1.095	1.095	323531	1.086
msg_bt	1.185	1.185	472321	1.161
msg_sppm	5.998	5.998	7.55	4.852

TABLE IV: Compression ratio for four compression designs with seven datasets.

Takeaway 3: While lossless compression typically yields lower compression ratios compared to lossy compression, it can still achieve high throughput for compression if the compression engine is used efficiently.

D. Performance Bottlenecks for using the compression engine

In the previous sections of performance characterizations, we observed that utilizing the compression engine is a worthwhile option for compression tasks, but not as advantageous for decompression due to the significant overheads associated with its usage. In this section, we further investigate the processing when utilizing the compression engine, aiming to identify the specific areas where these overheads arise and impact performance.

To gain deeper insights into the accelerator compression engine, Figure 4 shows an overview of utilizing the compression engine for a compression job, and Figure 5b breaks down its latency shown in Section IV-C. Notably, actual data compression only accounts for 9.4% of the total latency, with the majority of the time being spent on pre-processing tasks such as DOCA runtime initialization consumes 51.7% and DOCA buffer preparation consumes 38.6% of the total compression latency.

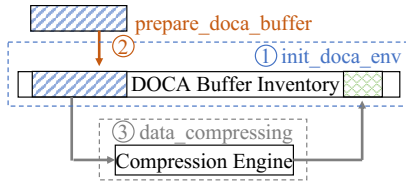
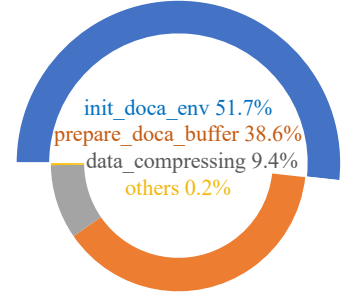


Fig. 4: Procedures of using the compression engine on DPUs.

In addition to analyzing the process of utilizing the compression engine, we also investigate its scalability, as this aspect is crucial for researchers to consider when designing systems that leverage the compression engine.

Figure 5a illustrates the limited scalability of job concurrency on the compression engine. We conducted experiments by submitting batches of 1, 2, 4, or 8 jobs to the compression engine simultaneously and recorded the compression time for each batch. From the figure, we observe that the processing time of the batch jobs increases linearly as the number of jobs in a batch increases.



(b) Time breakdown for a DOCA DEFLATE compression engine process.

Fig. 5: Performance bottlenecks characterization for execution compression with msg_bt on the compression engine of BlueField-2.

It reveals that even though the compression engine on the BlueField-2 can provide superior compression performance (as shown in previous sections), it is important for developers to be cautious when designing compression-based systems with the accelerator. For example, if multiple jobs need to be processed concurrently (whether for compression or decompression) which is common in big data scenarios, sending them to the accelerator all at once could introduce severe performance degeneration, as those concurrent processing are highly possible to be serialized.

From the compression algorithm standpoint, DOCA SDK only offers a single lossless compression algorithm for the compression engine. In the real-world scenario, different applications may have varying characteristics better suited for different compression algorithms. Therefore, addressing the provision of multiple compression algorithm choices is a potential aspect that needs to be considered.

These findings highlight the importance of 1) carefully selecting an appropriate compression job concurrency to minimize overhead, and 2) adopting a holistic design approach to efficiently reuse pre-allocated resources and batch processing with a single initialization.

Takeaway 4: Data staging and DOCA initialization constitute the most substantial overheads, consuming 90.4% of the total compression time when using the BlueField-2 compression engine. When leveraging this engine, further optimizations and consideration of the 128M data size limit per job are essential.

E. Evaluations on BlueField-2 v.s. BlueField-3

Recently, NVIDIA released the latest generation of BlueField DPU, BlueField-3, which boasts improved SoC Cores and higher

memory capability and bandwidth compared to BlueField-2. This paragraph gives a brief evaluation of the SoC Cores in both generations of BlueField DPUs.

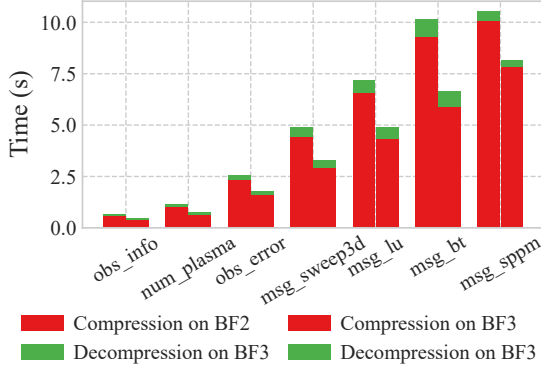


Fig. 6: BF2 v.s. BF3 Compression Time for DEFLATE compression on SoC Cores.

As illustrated in Figure 6, BlueField-3 outperforms BlueField-2 across all datasets, with the largest performance gap exceeding 25%. This difference can be attributed to the enhancements in BlueField-3, as outlined previously.

It is foreseeable that the SoC core would provide even more powerful computing resources. On the one hand, it is announced that BlueField-2 and BlueField-3 provide GPU-direct support, which will add extra heterogeneity to the system; On the other hand, emerging interconnection technologies and protocol, e.g. Compute Express Link [36], will enable more composable compute resources beyond the SoC core. All those changes will bring a DPU device, namely BlueField, with extra opportunities and challenges to developers to design a more scalable and efficient system.

We could only evaluate the BlueField-3's SoC cores for compression/decompression workloads due to the non-availability of the DOCA SDK module that provides access to the BlueField-3's compression engine on the Thor cluster. In the future, we would like to compare the compression engine of BlueField-2 and BlueField-3.

Takeaway 5: BlueField-3 reduces compression latency for DEFLATE lossless compression on SoC cores by up to 25% compared to BlueField-2. This improvement offers developers enhanced possibilities for creating scalable and efficient systems.

F. Discussions

Based on the performance characterizations mentioned above, we can offer some design guidelines for utilizing the BlueField DPU compression engine. As demonstrated, the engine is highly effective in reducing CPU utilization and outperforms SoC cores in most cases, making it an excellent choice for accelerating various applications. However, it is important to avoid overloading the compression engine as it can lead to decreased performance. To optimize the utilization of the compression engine, it is recommended to carefully balance the workload and avoid overwhelming it with excessive tasks simultaneously. Reusing allocated resources, such as the DOCA-processed buffer, and

processing as many tasks as possible within a single initialization can further enhance performance and efficiency.

Furthermore, as depicted in Figure 5b, we can clearly observe the overheads related to the utilization of BlueField's compression engine. While the compression engine shows potential for efficient data workload processing, its usage is not straightforward, and considerable efforts are required for effective programming. Consequently, it is crucial to consider this overhead when designing systems that integrate the BlueField DPU compression engine.

Additionally, it is important to note the compression engine only supports DEFLATE compression and decompression algorithms, limiting its flexibility in handling other compression schemes.

By considering these design guidelines, developers can maximize the performance of utilizing compression engines in various applications, ensuring optimal designs and minimizing potential overheads. This includes accelerating data storage systems or applying on-the-fly compression to minimize data traffic and improve bandwidth efficiency for communication libraries.

V. RELATED WORK

As discussed in Section II, there are many compression and decompression techniques proposed in the community. SZ3 [8] is an error-bounded lossy compression framework that uses a customized compression pipeline. SZ3 uses a prediction-based compression approach where the difference between original and decompressed data is lower than a user-specified error bound. cuSZ [9] is another error-bounded lossy compression framework that utilizes GPUs for compressing data. cuSZ supports fine-/coarse-grained parallelism on GPUs for Huffman Coding for the SZ compressor [28]. Each of these frameworks is supported on a specific platform and hardware architecture. For example, SZ3 is supported on CPUs, cuSZ is supported on GPUs, and WaveSZ [18] is implemented on FPGAs.

These existing schemes were all evaluated on how efficiently they store real-world scientific datasets with a specific compression type.

Also, there exists several studies [37]–[39] to characterize the performance and capabilities of SmartNICs, including both the networking and computing performance characterizations of NVIDIA BlueField and BlueField-2 DPU. These works give a general overview of the off-path DPUs performance characterizations.

Compared to these, our research specifically focuses on the synergy between a dedicated hardware accelerator and the compression and decompression workloads tailored for BlueField DPUs.

VI. CONCLUSION AND FUTURE WORK

We extensively study the capabilities of NVIDIA BlueField-2/3 DPUs on compression workloads with seven HPC datasets. We also identified the limitations and potential optimization opportunities for system designs on both SoC cores and the compression engine of BlueField DPU. We observed that the hardware compression engine of the DPU can achieve a performance speedup of up to 26.8x compared to the SoC cores for compression tasks. However, when it comes to decompression, the overhead introduced by the compression engine cannot be overlooked and may negatively impact performance. Therefore, efficiently utilizing the compression engine poses a significant challenge. Based on the characterization results, we gave multiple design guidelines for efficiently utilizing

the BlueField DPUs for compression tasks. Our study fills the gap in the literature by systematically exploring the performance characteristics of DPU-accelerated lossless and lossy compression and decompression schemes.

Based on our characterization work, our future research will further explore efficient system co-design with hardware accelerations on BlueField DPUs. For instance, leveraging the DPU's capabilities, we aim to investigate the potential of offloading communication tasks from the host. One potential area of interest is the co-design of message compression techniques to facilitate efficient message passing among nodes with the assistance of the DPU.

ACKNOWLEDGMENT

We express gratitude to our reviewers and specifically thank Weicong Chen, Liuyao Dai, and Hao Qi for enhancing our paper's figures. We gratefully acknowledge the computing resources provided on Thor, an HPC cluster operated by HPC Advisory Council, and the generous hardware donation from NVIDIA for the BlueField resources. Part of this research was funded through a subaward granted by Argonne National Laboratory. Part of this research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration, and by the U.S. Department of Energy, Office of Science, under Contract DE-AC02-06CH11357.

REFERENCES

- [1] Z. Zhang, C. Chang, H. Lin, Y. Wang, R. Arora, X. Jin, Is Network the Bottleneck of Distributed Training?, in: Proceedings of the Workshop on Network Meets AI and ML, NetAI '20, Association for Computing Machinery, 2020, p. 8–13. doi:10.1145/3405671.3405810.
- [2] H. Xu, C.-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini, P. Kalnis, GRACE: A Compressed Communication Framework for Distributed Machine Learning, in: 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), 2021, pp. 561–572. doi:10.1109/ICDCS51616.2021.00060.
- [3] Y. Lin, S. Han, H. Mao, Y. Wang, W. J. Dally, Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training (2020). arXiv:1712.01887.
- [4] K. Mishchenko, E. Gorbunov, M. Takáč, P. Richtárik, Distributed Learning with Compressed Gradient Differences (2019). arXiv:1901.09269.
- [5] X. Lu, M. W. U. Rahman, N. Islam, D. Shankar, D. K. Panda, Accelerating Spark with RDMA for Big Data Processing: Early Experiences, in: IEEE 22nd Annual Symposium on High-Performance Interconnects, 2014. doi:10.1109/HOTI.2014.15.
- [6] M. W. Rahman, N. S. Islam, X. Lu, J. Jose, H. Subramoni, H. Wang, D. K. Panda, High-Performance RDMA-based Design of Hadoop MapReduce over InfiniBand, in: The Proceedings of International Workshop on High Performance Data Intensive Computing (HPDIC), in conjunction with IEEE International Parallel and Distributed Processing Symposium (IPDPS), Boston, 2013.
- [7] N. S. Islam, X. Lu, M. W. Rahman, D. K. Panda, SOR-HDFS: A SEDA-based Approach to Maximize Overlapping in RDMA-Enhanced HDFS, in: The Proceedings of The 23rd International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC), Vancouver, Canada, 2014.
- [8] X. Liang, K. Zhao, S. Di, S. Li, R. Underwood, A. M. Gok, J. Tian, J. Deng, J. C. Calhoun, D. Tao, Z. Chen, F. Cappello, SZ3: A Modular Framework for Composing Prediction-Based Error-Bounded Lossy Compressors, IEEE Transactions on Big Data 9 (2) (2023) 485–498. doi:10.1109/TBDATA.2022.3201176.
- [9] J. Tian, S. Di, K. Zhao, C. Rivera, M. H. Fulp, R. Underwood, S. Jin, X. Liang, J. Calhoun, D. Tao, F. Cappello, cuSZ: An Efficient GPU-Based Error-Bounded Lossy Compression Framework for Scientific Data, in: Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques, Association for Computing Machinery, 2020. doi:10.1145/3410463.3414624.
- [10] L. P. Deutsch, GZIP File Format Specification Version 4.3, <https://www.rfc-editor.org/rfc/rfc1952> (1996).
- [11] Facebook, Zstandard, <https://facebook.github.io/zstd/> (2023).
- [12] NVIDIA, NVIDIA DOCA Compress Programming Guide, <https://docs.nvidia.com/doca/sdk/compress-programming-guide/index.html> (2022).
- [13] H. Lim, D. G. Andersen, M. Kaminsky, 3LC: Lightweight and Effective Traffic Compression for Distributed Machine Learning (2018). arXiv:1802.07389.
- [14] C.-F. Lee, S. W. Changchien, W.-T. Wang, J.-J. Shen, A Data Mining Approach to Database Compression, Information Systems Frontiers 8 (2006) 147–161.
- [15] S. M. Darwish, Improving Semantic Compression Specification in Large Relational Database, IET Software 10 (4) (2016) 108–115.
- [16] A Survey of Data Compression Algorithms and Their Applications, author=Hosseini, Mohammad, Network Systems Laboratory, School of Computing Science, Simon Fraser University, BC, Canada (2012).
- [17] T. Srisooksai, K. Keamarungsi, P. Lamsrichan, K. Araki, Practical Data Compression in Wireless Sensor Networks: A Survey, Journal of Network and Computer Applications 35 (1) (2012) 37–59, collaborative Computing and Applications. doi:https://doi.org/10.1016/j.jnca.2011.03.001. URL <https://www.sciencedirect.com/science/article/pii/S1084804511000555>
- [18] J. Tian, S. Di, C. Zhang, X. Liang, S. Jin, D. Cheng, D. Tao, F. Cappello, WaveSZ: A Hardware-Algorithm Co-Design of Efficient Lossy Compression for Scientific Data, in: Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '20, 2020, p. 74–88. doi:10.1145/3332466.3374525.
- [19] NVIDIA BLUEFIELD-2 DPU, <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-2-dpu.pdf> (2023).
- [20] Fungible, THE FUNGIBLE DATA PROCESSING UNIT Enabling Hyperdisaggregation of Compute and Storage Resources Across Data Center Scales, <https://www.fungible.com/product/dpu-platform/> (2022).
- [21] Marvell, Marvell LiquidIO III, <https://www.marvell.com/content/dam/marvell/en/public-collateral/embedded-processors/marvell-liquidio-III-solutions-brief.pdf> (2023).
- [22] Intel Unveils Infrastructure Processing Unit, <https://www.intel.com/content/www/us/en/newsroom/news/infrastructure-processing-unit-data-center.html>.
- [23] Broadcom, Stingray PS250, <https://docs.broadcom.com/doc/PS250-PB> (2021).
- [24] Xilinx, Alveo: Adaptable Accelerator Cards for Data Center Workloads, <https://www.xilinx.com/products/boards-and-kits/alveo.html>.
- [25] SmartNICs to xPUs: Why is the Use of Accelerators Accelerating?, <https://www.snia.org/sites/default/files/ESF/SmartNICs-to-xPUs-Why-is-the-Use-of-Accelerators-Accelerating.pdf> (2022).
- [26] J.-I. Gailly, M. Adler, zlib, <https://zlib.net/> (2022).
- [27] NVIDIA BLUEFIELD-3 DPU, <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf> (2023).
- [28] D. Tao, S. Di, Z. Chen, F. Cappello, Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization, in: 2017 IEEE International Parallel and Distributed Processing Symposium, 2017. doi:10.1109/IPDPS.2017.115.
- [29] K. Zhao, S. Di, X. Liang, S. Li, D. Tao, Z. Chen, F. Cappello, Significantly Improving Lossy Compression for HPC Datasets with Second-Order Prediction and Parameter Optimization, in: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, Association for Computing Machinery, 2020. doi:10.1145/3369583.3392688.
- [30] S. Claggett, S. Azimi, M. Burtcher, SPDP: An Automatically Synthesized Lossless Compression Algorithm for Floating-Point Data, in: 2018 Data Compression Conference, 2018, pp. 335–344. doi:10.1109/DCC.2018.00042.
- [31] M. Burtcher, P. Ratanaworabhan, FPC: A High-Speed Compressor for Double-Precision Floating-Point Data, IEEE Transactions on Computers 58 (1) (2009) 18–31. doi:10.1109/TC.2008.131.
- [32] F. Alted, Blosc, An Extremely Fast, Multi-threaded, Meta-compressor Library, <https://www.blosc.org/> (2023).
- [33] Scientific IEEE 754 32-Bit Single-Precision Floating-Point Datasets, <https://userweb.cs.txstate.edu/~burtcher/research/datasets/FPsingle/> (2023).
- [34] NVIDIA, NVIDIA DOCA Software Framework, <https://developer.nvidia.com/networking/doca> (2023).
- [35] HPC Advisory Council, <https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/overview> (2023).
- [36] Compute Express Link (CXL), <https://www.computeexpresslink.org/> (2023).
- [37] J. Liu, C. Maltzahn, C. Ulmer, M. L. Curry, Performance Characteristics of the Bluefield-2 Smartnic, arXiv preprint arXiv:2105.06619 (2021).
- [38] S. Sun, C. Huang, R. Zhang, L. Chen, Y. Huang, M. Yan, J. Wu, A Comprehensive Study on Optimizing Systems with Data Processing Units, arXiv preprint arXiv:2301.06070 (2023).
- [39] X. Wei, R. Chen, Y. Yang, R. Chen, H. Chen, A Comprehensive Study on Off-path SmartNIC, arXiv preprint arXiv:2212.07868 (2022).