Compression Analysis for BlueField-2/-3 Data Processing Units: Lossy and Lossless Perspectives

Yuke Li and Arjun Kashyap, *University of California, Merced, Merced, CA, 95343, USA*
Yanfei Guo, *Argonne National Laboratory, Lemont, IL, 60439, USA*
Xiaoyi Lu, *University of California, Merced, Merced, CA, 95343, USA*

A data processing unit (DPU) with programmable smart network interface card containing system-on-chip (SoC) cores is now a valuable addition to the host CPU, finding use in high-performance computing (HPC) and data center clusters for its advanced features, notably, a hardware-based data compression engine (C-engine). With the convergence of big data, HPC, and machine learning, data volumes burden communication and storage, making efficient compression vital. This positions DPUs as tools to accelerate compression workloads and enhance data-intensive applications. This article characterizes lossy (e.g., SZ3) and lossless (e.g., DEFLATE, lz4, and zlib) compression algorithms using seven real-world datasets on Nvidia BlueField-2/-3 DPUs. We explore the potential opportunities for offloading these compression workloads from the host. Our findings demonstrate that the C-engine within the DPU can achieve up to 26.8x speedup compared to its SoC core. We also provide insights on harnessing BlueField for compression, presenting seven crucial takeaways to steer future compression research with DPUs.

The convergence of big data, high-performance computing (HPC), and machine learning (ML) technologies has exacerbated data volume challenges within communication and storage systems, making it a salient application bottleneck. Past research[1,2] has accentuated the data movement challenges in distributed deep neural networks. This has ignited investigations[3] to boost communication-intensive frameworks for big data applications. The quest for efficient data reduction techniques has become paramount within the HPC and ML paradigms.

Data compression, a salient data reduction method, has been widely adopted by many applications, including ML, databases, and network communication. In ML, methods like those in Xu et al.[2] compress gradients during distributed training to curtail data movement bottlenecks.

0272-1732 © 2023 IEEE
Digital Object Identifier 10.1109/MM.2023.3343636
Date of publication 18 December 2023; date of current version 9 April 2024.

Recent findings[4] emphasize the significance of lossy and lossless compression techniques in scientific data processing. These techniques, however, are computationally intensive. Consequently, software–hardware co-designed solutions have emerged, mostly tailored to traditional platforms like x86 CPUs, GPUs, field-programmable gate arrays, and ARM CPUs.

Simultaneously, the data processing unit (DPU) evolution offers a promising frontier. DPUs like Nvidia's BlueField[5] provide capabilities beyond traditional network interface cards (NICs) and are suitable for offloading tasks from the CPU. They are equipped with hardware accelerators, enhancing compression operations.

Yet, there is a research gap in thoroughly understanding DPU-accelerated compression/decompression techniques. Using DPUs can conserve resources, reduce communication payload, and bolster application performance. We aim to probe the performance aspects of compression and decompression on DPUs, shedding light on future research trajectories.

However, as we strive to achieve this overarching goal, a set of significant challenges emerges, which we

encapsulate into the following three fundamental questions:

1) How do we determine the appropriate dimensions for a comprehensive assessment of performance for compression/decompression workloads across both system-on-chip (SoC) cores and the compression engine (C-engine) within the DPU?
2) How do we analyze the results derived from characterization dimensions to decipher the processing capabilities of the SoC cores and the C-engine when handling compression/decompression workloads?
3) How do we unearth potential optimization prospects and consolidate them into guidelines for future endeavors, thereby furnishing valuable insights to further elevate the performance of compression/decompression workloads?

To answer these questions, in this article, we introduce a 3-D characterization methodology encompassing hardware, datasets, and algorithm dimensions to study the performance of Nvidia BlueField-2/-3 DPUs systematically. To the best of our knowledge, this is the first evaluation and analysis of the compression and decompression capabilities of Nvidia BlueField-2/-3 DPUs. Our key contributions are as follows:

- Our study encompasses a comprehensive performance analysis of Nvidia BlueField DPUs, specifically focusing on one lossy (SZ3[4]) and three lossless (DEFLATE,[5] lz4,[6] and zlib[7]) compression/decompression schemes. This examination allows us to discern their capabilities and limitations effectively.
- Our investigation delves deeply into the C-engine of BlueField-2/-3 DPUs, providing insights into their performance bottlenecks and optimization potentials. Our findings reveal that the C-engine on BlueField-2/-3 DPUs is capable of achieving a 26.8× (4.4×) speedup compared to running compression (decompression) workloads on the SoC cores. However, this improvement is accompanied by a significant 90.4% overhead attributed to data staging and initialization.
- After conducting comprehensive evaluations and comparisons, we have observed a performance improvement with the BlueField-3 DPU, exceeding 25% compared to the BlueField-2 DPU.
- The meticulous characterization of the DPU's compression and decompression functions leads us to propose a set of design guidelines that researchers can follow for optimizing their approaches.

BACKGROUND

This section offers insights into Nvidia DPUs and discusses the significance of compression techniques.

Nvidia BlueField-2/-3 DPUs

BlueField-2

DPUs, or SmartNICs, are enhanced NICs that aid servers in computation offloading. Nvidia's BlueField-2 DPUs,[5] shown in Figure 1, have general-purpose SoC cores and house hardware accelerators for functions like compression and decompression. The C-engine in a BlueField-2 DPU supports the DEFLATE lossless compression algorithm. BlueField DPUs operate in two primary modes:

1) *Separated host mode*: The DPU acts independently, keeping its network ports distinct from the host.
2) *SmartNIC mode*: All network traffic goes through a virtual switch on ARM cores, giving it the ability to control network packets.

BlueField-3

The BlueField-3,[5] an evolution of BlueField-2, brings additional hardware capabilities. Table 1 outlines the differences. BlueField-3 supports faster networking, more powerful ARM cores, and extra hardware accelerators. Memory capabilities are also upgraded with BlueField-3 adopting the Double Data Rate 5 (DDR5) standard. Apart from the DEFLATE decompression scheme, the BlueField-3 C-engine introduces support for an additional decompression algorithm—lz4, which is unavailable in BlueField-2.

To program with BlueField-2/-3's hardware accelerators, the Data Center Infrastructure on a Chip Architecture (DOCA) software development kit (SDK[5]) can be used.

Compression

Data reduction via compression is becoming increasingly important when large-scale HPC and data center

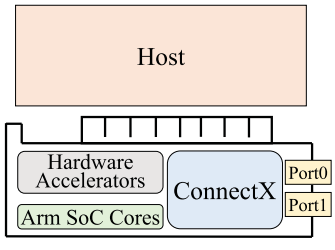

FIGURE 1. BlueField-2/-3 architecture.

TABLE 1. Hardware specifications of BlueField-2 and BlueField-3.

	SoC cores	Interconnects	Max BW	Network adapter	Memory	PCIe Interface
BF-2	8× ARM Cortex-A72	Ethernet, InfiniBand	200 Gb/s	Nvidia ConnectX-6 Dx	16-GB/32-GB onboard DDR4	Eight- or 16-lane PCIe Gen 4.0
BF-3	16× ARM Cortex-A78AE	Ethernet, InfiniBand	400 Gb/s	Nvidia ConnectX-7	16-GB onboard DDR5	32-lane PCIe 5.0

max BW: maximum bandwidth; DDR5: Double Data Rate 5.

applications generate huge volumes of data. Broadly, the community primarily uses two compression techniques for scientific data size reduction—lossy and lossless:

1) *Lossy compression*: This achieves high compression ratios by sacrificing some information. It cannot rebuild the exact original data. Lossy compression methods (e.g., SZ3[4]) are crucial for data reduction while maintaining certain error bounds.
2) *Lossless compression*: This reproduces the exact original data upon decompression, preserving data integrity and quality. Although it offers lower compression ratios, it is vital for specific datasets. Popular examples of lossless compression include DEFLATE,[5] lz4,[6] and zlib.[7]

CHARACTERIZATION METHODOLOGY

This section details our characterization methodology, depicted in Figure 2, with detailed descriptions for each dimension in subsections. We also outline our testbed setup.

3-D Characterization Methodology

Compression Designs

We test four key compression algorithms (lossless and lossy) on the SoC core and C-engine, as illustrated in Table 2, across various BlueField DPUs.

TABLE 2. Compression designs and features.

Algorithm	Purpose	Lossless	Lossy
DEFLATE	General data compression	✓	
zlib	—	✓	
lz4	—	✓	
SZ3	Scientific data compression		✓

We focus on the algorithms supported by BlueField-2 and BlueField-3, as highlighted in Table 3. Our three-phase approach includes 1) testing DEFLATE, zlib, and SZ3 compression using seven datasets on the BlueField-2 DPU's SoC ARM cores; 2) utilizing the C-engine of BlueField-2 to assess DEFLATE compression/decompression; and 3) investigating lz4 and DEFLATE decompression performance on the newly released BlueField-3 DPU's C-engine.

For lossless algorithms (e.g., DEFLATE, zlib, and lz4), we maintain default settings. DEFLATE's compression ratios are slightly tweaked due to variations caused by distinct SoC and engine configurations.

Regarding lossy compression, we chose SZ3 with an error bound of `1E-3`, its default setting. On BlueField-2's C-engine, only DEFLATE is used via the DOCA SDK due to algorithmic limitations. Additionally, zlib is an enhanced version of DEFLATE, with added header and trailer data.[7]

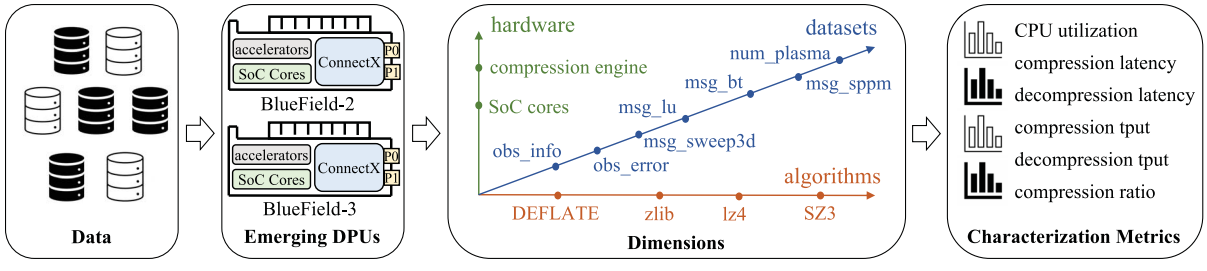

FIGURE 2. Overview of the characterization methodology.

TABLE 3. Compression algorithms supported by different hardware.

Algorithm	SoC core	C-engine	
		Compression	Decompression
DEFLATE	BF2, BF3	BF2	BF2, BF3
zlib	BF2, BF3	—	—
lz4	BF2, BF3	—	BF3
SZ3	BF2, BF3	—	—

Datasets

We select seven HPC datasets with single-precision floating points from Burtscher et al.[8] to evaluate compression performance on BlueField DPUs, as shown in Table 4. Notably, some datasets, like `msg_sppm`, are resized to 128 MB to adhere to BlueField DPU's C-engine data chunk limits while preserving data distribution.

The chosen datasets, detailed in the article,[8] vary in size, unique value percentages, and randomness. Such diversity ensures our findings' breadth and relevance as datasets can behave differently under compression.

Hardware

BlueField-2 is renowned as a groundbreaking data center chip solution, with its successor, BlueField-3, introducing significant hardware enhancements. Beyond general-purpose SoC cores, these DPUs incorporate features like dedicated C-engines, broadening workload execution options for users.

Performance Metrics

We evaluate BlueField DPU architectures using four compression designs across seven datasets. Our tests measure both compression outcomes and hardware utilization.

The metrics include latency (time taken for compression/decompression) and throughput (bytes processed per second). The compression ratio was determined by the original data size divided by its compressed size.

For hardware usage, we log average CPU utilization for each design-dataset pair, shown as a percentage. To ensure unbiased results, data loading/storing times are excluded; all data were preloaded into DPU memory before testing.

Testbed

Our BlueField-2 local setup features eight ARM Cortex-A72 cores, 16-GB DDR4 dynamic random access, and is operated on Ubuntu 20.04.5 with DOCA SDK v1.5.0.[5] Evaluations for the "Comparative Evaluation: BlueField-2 Versus BlueField-3 SoC Core," "Evaluations on BlueField-2 Versus BlueField-3—C-Engine," and "Compression on BlueField-3" sections are conducted on the HPC Advisory Council High-Performance Center's[9] Thor cluster. Thor's BlueField-2 mirrors our local setup but uses Rocky

TABLE 4. The chosen seven HPC datasets with various statistical information in size, unique values, entropy, and randomness.

Dataset	Size (MB)	Unique values (%)	Entropy (bits)	Randomness (%)	Data description
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

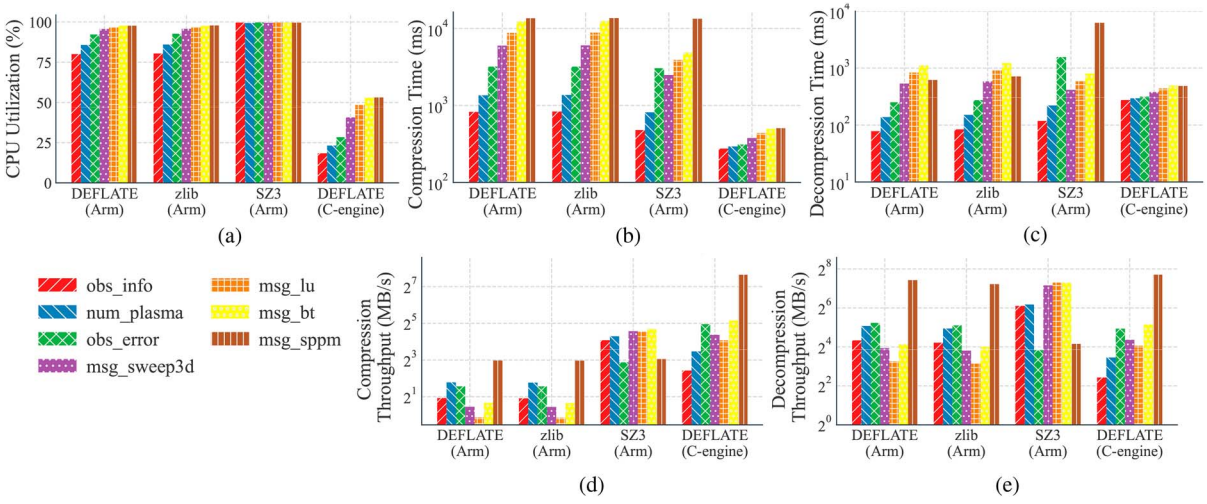

FIGURE 3. Characterization results with different dimensions when executing the four compression designs with seven datasets on the SoC core and the C-engine of BlueField-2. (a) CPU usage the of SoC core. (b) Compression latency. (c) Decompression latency. (d) Compression throughput. (e) Decompression throughput.

Linux 8.6. The BlueField-3 DPU on Thor features 16 ARM Cortex-A78 cores, 16-GB DDR5 memory, and runs Rocky Linux 9.1. Both Thor's BlueField-2/-3 utilize DOCA SDK v2.0. All experiments are run in separated host mode for BlueField-2/-3 DPUs.

COMPRESSION PERFORMANCE CHARACTERIZATION

This section presents a thorough characterization of Nvidia's BlueField-2/-3 DPUs regarding compression/decompression.

Characterize CPU Usage on the SoC Core

Given BlueField-2's SoC core's limited capacity for compression/decompression tasks, the dedicated C-engine offers a path to improved efficiency and performance. We study CPU usage for these tasks on both the SoC core and C-engine.

Figure 3(a) displays CPU usage for four compression designs across seven datasets. Running lossless compression on the SoC core, CPU usage exceeds 80%, regardless of the dataset size. SZ3 compression shows near-maximal CPU usage. By contrast, leveraging the C-engine with DEFLATE compression, CPU usage drops dramatically, ranging from 18% for the 9.1-MB dataset to 53% for the 128-MB dataset (see "Takeaway 1"). It is evident that smaller datasets on the C-engine further decrease CPU load. Still, the SoC core manages tasks like data movement coordination and memory registration for the engine.

Takeaway 1

Despite offloading most of the tasks to the compression engine, the SoC core's CPU usage can reach 53% for a 128-MB dataset, which is attributed to its role in data movement and memory management for the engine.

Characterize Compression and Decompression Latency

We note a reduction in CPU usage when leveraging the C-engine. This section aims to assess its impact on compression/decompression performance and to delineate the benefits of relying solely on the SoC core.

Figure 3(a) and (b) depicts the latency for compression/decompression across various compression designs and datasets. From Figure 3(b), it is evident that the design utilizing the C-engine consistently surpasses other designs in performance, achieving speedups of an order of magnitude. Specifically, datasets `msg_bt` and `msg_sppm` witness speedups ranging from 25× to 26.8× when compared to tasks run on the SoC core. Even with the smallest dataset, `obs_info`, the C-engine manages a speedup of 3× over the SoC core, underscoring its efficiency and potential for compression tasks.

However, as shown in Figure 3(c), for datasets such as `obs_info`, `num_plasma`, and `obs_error` with sizes of 9.1, 17, and 30 MB (the first three bars in each lossless compression design), respectively, the decompression

time of utilizing the C-engine degrades up to 3.5× compared to decompressing the data on the SoC core. Nevertheless, we still observe the benefits of using the C-engine for datasets larger than 30 MB.

The performance decline in using a C-engine stems from the inherent overheads of 1) data staging, where the data must be transferred from memory to the C-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 core. However, as decompression time is naturally much shorter than compression time, the reduced decompression time cannot mask the overhead of using the accelerator (see "Takeaway 2"). Hence, the DEFLATE (C-engine) decompression time is greater than others shown in Figure 3(c).

Furthermore, we observe that 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 is discussed in the "Characterize Compression and Decompression Throughput and the Compression Ratio" section. However, the decompression performance of SZ3 may vary significantly depending on the dataset characteristics.

Takeaway 2

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

Characterize Compression and Decompression Throughput and the Compression Ratio

It is also essential to assess the efficiency of various compression designs on the SoC core and the C-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.

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. A shorter latency does not necessarily imply higher throughput as it also depends on the volume and compressibility of the processed data.

In Figure 3(d) and (e), we observe significant variations in compression/decompression throughput across different datasets. However, it is notable that running lossy compression (e.g., SZ3) on the SoC core achieves higher throughput than running lossless compression (e.g., DEFLATE and zlib) on the SoC core. For instance, for the `msg_lu` dataset, running lossy compression SZ3 on the SoC core achieves a throughput 260× higher compared to running lossless compression DEFLATE on the SoC core.

In addition, running lossless compression (e.g., DEFLATE) on the C-engine leads to improved compression throughput compared with running lossless compression on the SoC core. For instance, when considering the dataset `msg_sppm`, utilizing the C-engine with DEFLATE achieves a throughput 25.6× higher compared to executing DEFLATE on the SoC core.

Due to its inherent characteristic of lossy compression, SZ3 consistently achieves a much higher compression ratio (see "Takeaway 3"), as demonstrated in Table 5, among all dataset sizes; that is one of the most significant features of lossy compression algorithms.

However, when it comes to decompression throughput, utilizing a C-engine, as depicted in Figure 3(e), 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 C-engine, as discussed in the "Characterize Compression and Decompression Latency" section.

Takeaway 3

Although 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.

TABLE 5. Compression ratio for four compression designs with seven datasets.

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

Performance Bottlenecks for Using the C-Engine

In the previous performance analysis, we observed that although the C-engine is beneficial for compression tasks, it is less so for decompression due to substantial overheads. In this section, we aim to pinpoint the sources of these overheads.

To gain deeper insights into the accelerator C-engine, Figure 4(a) and (b) provides an overview of the C-engine's operation and its associated latency. Notably, actual data compression accounts for only 9.4% of the total latency, with preprocessing tasks, particularly DOCA runtime initialization (51.7%) and buffer preparation (38.6%), occupying the majority.

We further investigate the C-engine's scalability, vital for system designs that utilize it. Figure 4(c) reveals its constrained job concurrency scalability. In experiments with batches of one, two, four, and eight jobs submitted simultaneously, we find that processing time increased linearly with batch size. This suggests that the engine processes jobs sequentially, introducing execution serialization challenges for batched jobs.

Although the BlueField-2 DPU's C-engine exhibits high performance, developers should exercise caution when designing compression systems, especially in big data scenarios with concurrent job processing. Overloading the accelerator may degrade performance due to potential serialization.

The DOCA SDK's limitation to a single lossless compression algorithm on BlueField-2 also poses challenges as different applications may require diverse algorithms for optimal performance (see "Takeaway 4"). 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 preallocated 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 DPU compression engine. When leveraging this engine, further optimizations and consideration of the 128-MB data-size limit per job are essential.

Comparative Evaluation: BlueField-2 Versus BlueField-3 SoC Core

BlueField-3 exhibits superior SoC cores and commands a higher memory capability and bandwidth. This section presents a concise evaluation comparing the SoC cores of both the BlueField-2 and BlueField-3 generations.

The comparative analysis depicted in Figure 5(a) underscores the enhanced performance of the BlueField-3 SoC core, Armv8.2+ A78, compared to the BlueField-2's Armv8 A72 core. Across all examined datasets, BlueField-3 DPU consistently outperforms BlueField-2 DPU, showcasing performance discrepancies surpassing 25% (see "Takeaway 5").

Takeaway 5

BlueField-3 reduces compression latency for DEFLATE lossless compression on the SoC core by up to 25% compared to BlueField-2. This improvement offers developers enhanced possibilities for creating scalable and efficient data-intensive systems.

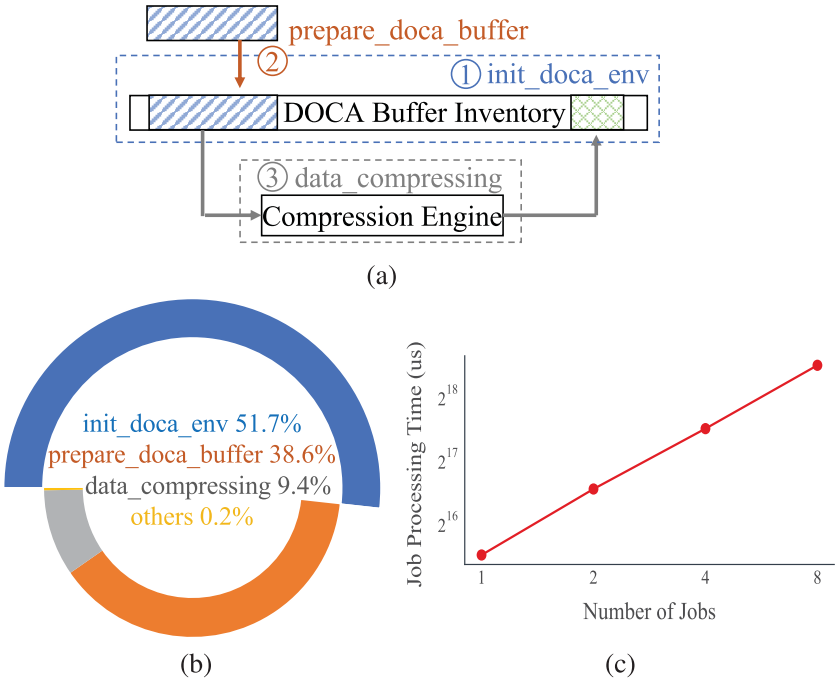

FIGURE 4. Performance characterization for execution compression with msg_bt on the C-engine of BlueField-2. (a) Steps for using the DPU's C-engine. (b) Time breakdown for a DOCA DEFLATE C-engine process. (c) Scalability evaluation of the C-engine. The *x*-axis is the number of jobs submitted to the C-engine at a time.

Evaluations on BlueField-2 Versus BlueField-3—C-Engine

Expanding upon the SoC performance comparison between BlueField-2 and BlueField-3, this section delves into the advancements in the C-engine. Specifically, we investigate whether the progress seen in BlueField-3's C-engine aligns with broader enhancements in its capabilities and performance. This analysis aims to provide a comprehensive understanding of the technological evolution spanning these two generations of BlueField DPUs.

Figure 5(b) presents a comparative assessment of decompression times involving the C-engines of BlueField-2 and BlueField-3, encompassing datasets ranging from 9.1 to 128 MB. It is important to note that the displayed times exclude any overhead from DOCA initialization and DOCA buffer preparation on the SoC core. The focus remains solely on the raw performance of decompression operations carried out by the C-engine.

An intriguing observation arises from this comparison: despite employing the same compression algorithm and dataset, the two BlueField generations exhibit distinct decompression timings. Upon a side-by-side examination, the C-engine of BlueField-3 consistently surpasses its BlueField-2 predecessor, with peak performance improvements reaching a noteworthy 58% (see "Takeaway 6"). This notable boost in speed is clear evidence that the newer BlueField-3's C-engine has undergone enhancements compared to the earlier BlueField-2 iteration.

Takeaway 6

BlueField-3's compression engine demonstrated up to a 58% improvement over BlueField-2, underscoring its enhanced performance.

Compression on BlueField-3

This section is dedicated to characterizing the performance of the lz4 compression algorithm on BlueField-3, spanning both the SoC core and C-engine. The approach involves initially characterizing the lz4 compression operation on the SoC core, followed by a comprehensive analysis and comparison of the lz4 decompression operation across both the SoC core and C-engine.

Figure 6(a) illustrates the performance characterization of the lz4 compression algorithm on the SoC

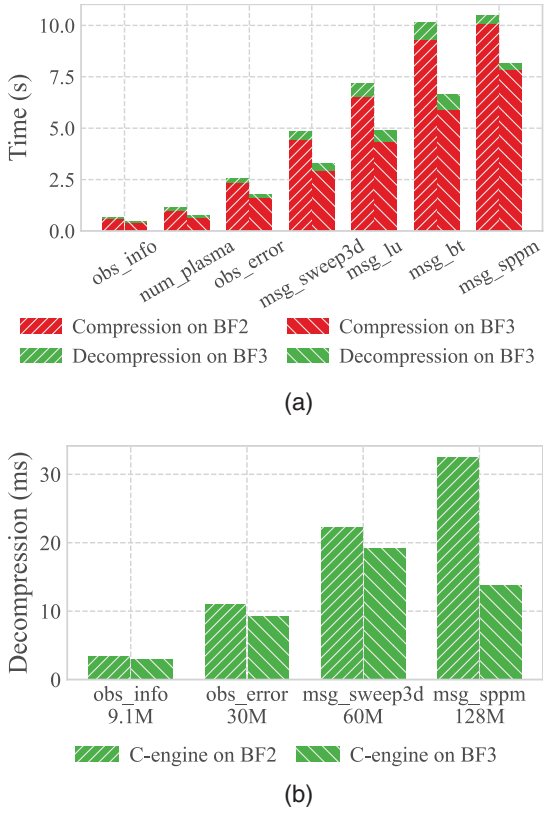

FIGURE 5. Compression time and decompression time comparisons for the DEFLATE algorithm. (a) Compression time comparison: the DEFLATE algorithm on the SoC core of BF2 versus BF3. (b) Decompression time comparison: the DEFLATE algorithm on the C-engine, BF2 versus BF3.

core of the BlueField-3 DPU. This analysis spans datasets ranging in size from 9.1 MB to 128 MB. A direct comparison with the DEFLATE algorithm [Figure 5(a)] reveals that lz4 completes compression tasks in a shorter timeframe. However, this advantage comes with a tradeoff. The compression ratio achieved by lz4, as outlined in Table 6, is marginally lower than DEFLATE when applied to the same dataset. This variance can be attributed to inherent differences in the design philosophies of the two algorithms. Consequent1y, in cases where lz4's compression ratio falls short of DEFLATE's for a given dataset, its compression throughput also lags. This pattern is visually evident in Figure 6(b).

Figure 6(c) showcases the performance metrics of the lz4 decompression operation executed on the C-engine of BlueField-3. Notably, the depicted timings exclude any overheads associated with initialization and buffer preparation on the SoC core, ensuring a pure evaluation of decompression performance. A distinct trend can be observed from the data: the C-engine notably outperforms the SoC core, particularly when managing datasets featuring higher compression ratios. This performance contrast becomes even more conspicuous when we delve into the throughput comparison depicted in Figure 6(d). To contextualize this, for the 128-MB dataset, boasting the highest compression ratio, the C-engine exhibits a decompression capability that is remarkably 4.4× faster than the SoC core (see "Takeaway 7").

Takeaway 7

The BlueField-3 compression engine introduces lz4 support, which was not available in BlueField-2. Notably, the engine outperforms the SoC core in decompression tasks, especially for high compression ratios, achieving speeds of up to 4.4× faster for specific datasets.

Discussion

Based on the outlined performance characterizations, several design guidelines emerge for effectively utilizing the BlueField DPU's C-engine. As evidenced, this engine showcases exceptional proficiency in reducing CPU usage and consistently outperforms the SoC core across numerous scenarios, making it an excellent choice for accelerating a spectrum of applications. For achieving optimal performance with the C-engine, it is important to design strategies like reusing resources, such as the DOCA-processed buffer, and maximizing task execution within a single initialization.

Furthermore, the overheads associated with employing BlueField's C-engine, as depicted in Figure 4(b), underscore the importance of judicious usage. While the engine exhibits promise for efficient data workload processing, its integration requires meticulous programming efforts. Hence, acknowledging this overhead becomes pivotal when crafting systems that incorporate the BlueField DPU's C-engine.

Moreover, the SoC core and the C-engine on the BlueField-2 DPU offer synergistic benefits for compression tasks. The data compressed with the SoC core using DEFLATE on BlueField-2 can be decompressed by its C-engine and vice versa, which is also compatible with BlueField-3, although BlueField-3's C-engine currently supports only decompression operations. Additionally, lz4, tailored for decompression on BlueField-3, further enables this synergy. Despite configurable settings

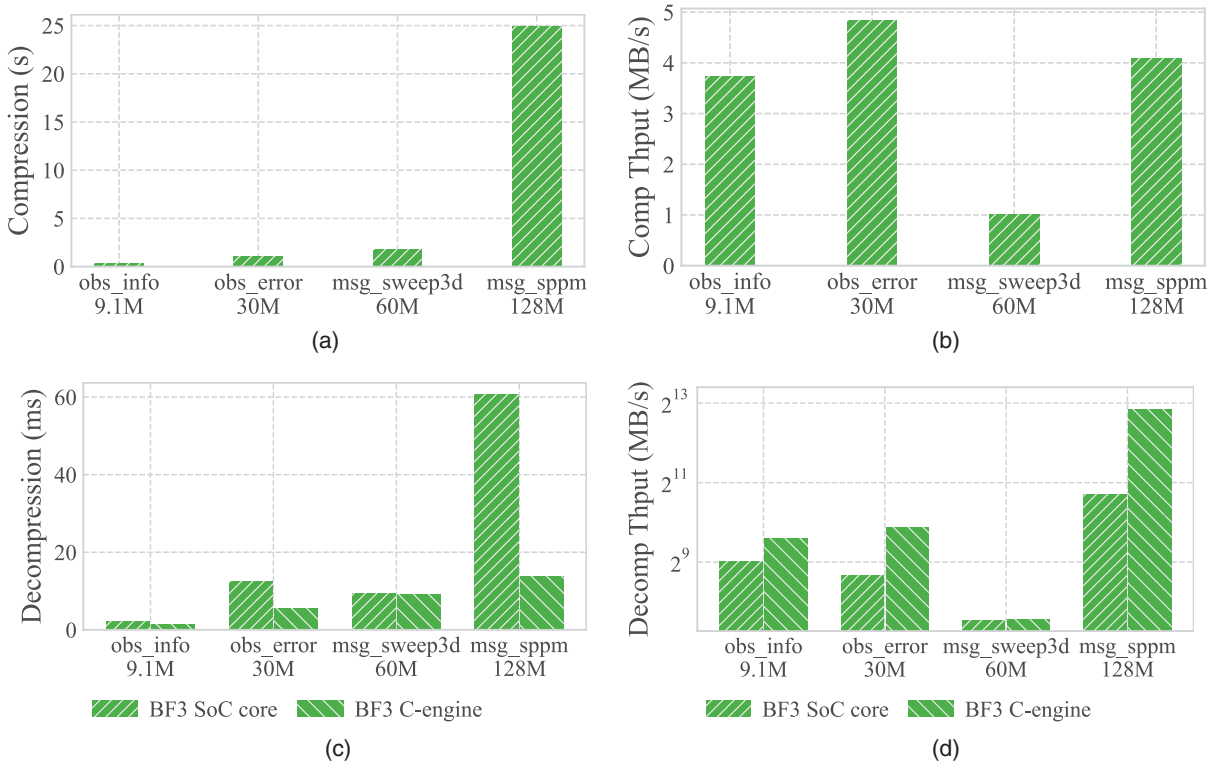

FIGURE 6. Characterization results with the lz4 algorithm on the BlueField-3 SoC core and the C-engine (BlueField-3 does not support compression operations on the C-engine). (a) Compression latency of lz4 on the BlueField-3 SoC core. (b) Compression throughput of lz4 on the BlueField-3 SoC core. (c) Decompression latency of lz4 on BlueField-3. (d) Decompression throughput of lz4 on BlueField-3.

like compression level for DEFLATE and lz4, effective cooperation can be achieved without requiring manual data format adjustments, which enables a versatile data processing pipeline.

By adhering to these design guidelines, developers can maximize the performance of C-engines in diverse applications, ensuring optimal designs while minimizing potential overheads. This includes enhancing data-intensive systems, libraries, and applications.

TABLE 6. Compression ratio comparison of lz4 and DEFLATE (on an SoC core) algorithm.

Dataset	CR-lz4	CR-DEFLATE
obs_info	1.135	1.211
obs_error	1.204	1.469
msg_sweep3d	1.03	1.159
msg_sppm	4.825	5.998

RELATED WORK

As highlighted in the "Background" section, various compression techniques exist. SZ3[4] is a prediction-based, error-bounded lossy compression for CPUs that maintains data accuracy within a specified error margin. Many of these compression methods focus on storage efficiency for real-world scientific datasets.

There are also studies[10,11,12] that analyze the performance attributes of SmartNICs, especially Nvidia BlueField DPUs. Distinctively, our research ventures into the intricate relationship between hardware accelerators and compression tasks, centering its focus on BlueField DPUs.

Relative to our original paper,[10] this article unveils fresh content. This includes comprehensive evaluations of the C-engine across BlueField-2 and BlueField-3 DPUs and a study for the recently incorporated lz4 compression design on BlueField-3. These additions are enriched with detailed performance analyses and the latest experimental insights.

CONCLUSION AND FUTURE WORK

This article extensively studied 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 C-engine of BlueField DPUs. We observe that the hardware C-engine of a DPU can achieve a performance speedup of up to 26.8× compared to the SoC cores for compression tasks. However, when it comes to decompression, the overhead introduced by a C-engine cannot be overlooked and may negatively impact performance. Therefore, efficiently utilizing the C-engine poses a significant challenge. Based on the characterization results, we give multiple design guidelines for efficiently utilizing 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-designs with C-engines on BlueField DPUs.

ACKNOWLEDGMENTS

We are grateful to our reviewers for their invaluable feedback on the article. We thank our lab mates, with special appreciation to Weicong Chen, Liuyao Dai, and Hao Qi for their valuable input. 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 supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy's (DOE's) Office of Science and the National Nuclear Security Administration, and by the DOE's Office of Science, under Contract DE-AC02-06CH11357. This work was supported in part by National Science Foundation research Grant Advanced Cyberinfrastructure 2321123 and Grant 2340982, a subaward granted by Argonne National Laboratory, and DOE research Grant DE-SC0024207.

REFERENCES

1. Z. Zhang, C. Chang, H. Lin, Y. Wang, R. Arora, and X. Jin, "Is network the bottleneck of distributed training?" in *Proc. Workshop Netw. Meets AI ML (NetAI)*, New York, NY, USA: Association for Computing Machinery, 2020, pp. 8–13, doi: 10.1145/3405671.3405810.
2. H. Xu et al., "GRACE: A compressed communication framework for distributed machine learning," in *Proc. IEEE 41st Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2021, pp. 561–572, doi: 10.1109/ICDCS51616.2021.00060.
3. X. Lu, M. W. U. Rahman, N. Islam, D. Shankar, and D. K. Panda, "Accelerating spark with RDMA for big data processing: Early experiences," in *Proc. IEEE 22nd Annu. Symp. High-Perform. Interconnects*, 2014, pp. 9–6, doi: 10.1109/HOTI.2014.15.
4. X. Liang et al., "SZ3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Trans. Big Data*, vol. 9, no. 2, pp. 485–498, Apr. 2023, doi: 10.1109/TBDATA.2022.3201176.
5. "NVIDIA BlueField data processing units." NVIDIA. Accessed: 2023. [Online]. Available: https://www.nvidia.com/en-us/networking/products/data-processing-unit/
6. "LZ4 - Extremely fast compression." LZ4. Accessed: 2023. [Online]. Available: https://lz4.org/
7. J-l Gailly and M. Adler, *zlib*, 2022. Accessed: 2023. [Online]. Available: https://zlib.net/
8. M. Burtscher, and P. Ratanaworabhan, "FPC: A high-speed compressor for double-precision floating-point data," *IEEE Trans. Comput.*, vol. 58, no. 1, pp. 18–31, Jan. 2009, doi: 10.1109/TC.2008.131.
9. HPC Advisory Council. Accessed: 2023. [Online]. Available: https://hpcadvisorycouncil.atlassian.net/wiki/spaces/HPCWORKS/overview
10. Y. Li, A. Kashyap, Y. Guo, and X. Lu, "Characterizing lossy and lossless compression on emerging BlueField DPU architectures," in *Proc. IEEE Symp. High-Perform. Interconnects (HOTI)*, 2023, pp. 33–40, doi: 10.1109/HOTI59126.2023.00019.
11. J. Liu, C. Maltzahn, C. D. Ulmer, and M. L. Curry, "*Performance characteristics of the BlueField-2 SmartNIC,*" Office of Scientific and Technical Information, Oak Ridge, TN, USA, 2021. [Online]. Available: https://www.osti.gov/biblio/1783736
12. X. Wei, R. Cheng, Y. Yang, R. Chen, and H. Chen, "Characterizing off-path SmartNIC for accelerating distributed systems," in *Proc. 17th USENIX Symp. Oper. Syst. Des. Implementation (OSDI 23)*, Boston, MA, USA: USENIX Association, 2023, pp. 987–1004. [Online]. Available: https://www.usenix.org/conference/osdi23/presentation/wei-smartnic

YUKE LI is a Ph.D. student in the Parallel and Distributed Systems Laboratory, Department of Computer Science and Engineering, University of California, Merced, Merced, CA, USA. Her research interests include high-performance computing, message passing interface, remote direct memory access, and data processing units. Li received her M.S. degree in computer science from the University of Edinburgh. She is

a student member of the Association for Computing Machinery. Contact her at yli304@ucmerced.edu.

ARJUN KASHYAP is a Ph.D. student in the Parallel and Distributed Systems Laboratory, Department of Computer Science and Engineering, University of California, Merced, Merced, CA, USA. His research interests include designing storage systems for high-performance computing and cloud computing. Kashyap received his master's degree from the University of Wisconsin–Madison. Contact him at akashyap5@ucmerced.edu.

YANFEI GUO holds an appointment as a computer scientist at Argonne National Laboratory, Lemont, IL, 60439, USA, where he leads the exascale computing message passing interface project and is a member of the MPICH team. His research interests include parallel programming models and runtime systems in extreme-scale supercomputing systems, data-intensive computing, and cloud computing systems. Guo received his Ph.D. degree in computer science from the University of Colorado, Colorado Springs. He is a Member of IEEE and the Association for Computing Machinery. Contact him at yguo@anl.gov.

XIAOYI LU is an assistant professor in the Department of Computer Science and Engineering, University of California, Merced, Merced, CA, USA, where he leads the Parallel and Distributed Systems Laboratory. His research interests include parallel and distributed computing, high-performance interconnects, advanced input–output technologies, big data analytics, and deep learning systems software. Lu received his Ph.D. degree in computer science from University of Chinese Academy of Sciences. He is a Member of IEEE and the Association for Computing Machinery. He is the corresponding author of this article. Contact him at xiaoyi.lu@ucmerced.edu.