Evaluation of Energy Consumption in Matrix Computations Using Compressed Data

Abdelmouhaimen Sarhane
University of Perpignan - E.N.A.C, FR

July 21, 2023

Abstract

This paper presents an evaluation of energy consumption in matrix computations using compressed data. The objective is to compare the energy efficiency of blaz, a lossy compression method that allows direct computation on compressed data, with zfp, another compression method, and no compression. The study utilizes the PowerJoular tool for energy consumption measurement and develops C codes implementing elementary matrix computations. The methodology includes analyzing the energy gains achieved by blaz and comparing them with zfp and no compression. The findings demonstrate the potential of blaz to significantly reduce energy consumption in matrix computations, offering promising prospects for sustainable scientific computing practices.

Keywords : Green computing; energy consumption; Floating-point arrays; compressed data; sustainable computing; blaz; zfp.

I Introduction

Scientific computing plays a crucial role in conducting extensive numerical simulations in various fields of science and industry. However, these simulations often require substantial computational power and handle vast amounts of data, measured in exaflops and petabytes.

One of the key challenges in dealing with large data sets is the need for efficient storage, communication, and processing. Traditional storage methods often fall short in terms of scalability and energy efficiency when handling the immense volumes of scientific data. Additionally, the transmission and processing of such data require substantial energy resources. As a result, the development of sustainable science based on frugal computing solutions has become a critical area of research, addressing the need for energy-efficient techniques that can handle these large data sets.

In recent years, various compression techniques have been proposed to alleviate the storage burden associated with scientific data. These techniques focus on reducing the memory requirements of floating-point number arrays used in scientific computations [3, 1]. Although these methods have proven effective in saving memory, they introduce additional computational overhead due to the compression and decompression processes required before data processing. This trade-off between memory gains and increased computational resources poses a significant challenge in achieving energy-efficient scientific computing.

To address this challenge, a new lossy compressor called blaz has been recently proposed for 2D floating-point number matrices [4]. Blaz allows for direct computation on compressed data without the need for decompression, enabling significant memory and computational time savings compared to traditional compression techniques. Basic linear algebra operations, such as dot product, addition, and multiplication by a matrix or a constant, can be performed directly on compressed matrices using blaz.

The objective of this research is, using the PowerJoular tool [5], to evaluate the energy consumption of different methods for performing elementary matrix computations, specifically comparing blaz with another compressor called zfp [3], as well as the scenario of no compression. By assessing the energy consumption of these methods, we aim to determine the energy efficiency gains achieved by blaz and compare them to alternative approaches.

Moreover, we apply the compressors to a practical sports field scenario, specifically football match analysis. Analyzing player movement heatmaps is a crucial tool used by coaches, analysts, and sports scientists to understand player positions, patterns, and tactical behavior during matches. We compute the average player movement heatmap using blaz and zfp compressors and evaluate their energy consumption, demonstrating the applicability of these methods beyond scientific computing.

II Tools

II.I Zfp

ZFP is a fixed-rate compression scheme for floating-point data that aims to provide near-lossless compression while allowing random access to the compressed data. Inspired by fixed-rate texture compression methods used in graphics hardware, ZFP maps small blocks of 4^d values in d dimensions to a fixed, user-specified number of bits per block. This enables read and write random access to compressed floating-point data at the granularity of blocks.

The compression process in ZFP involves (1) aligning the values in a block to a common exponent and (2) converting them to a fixed-point representation;(3) an orthogonal block transform is then applied to decorrelate the values, and (4) the resulting coefficients are ordered by expected magnitude. (5) Finally, the coefficients are encoded one "bit plane" at a time [3]. ZFP supports truncating the bit stream at any point, allowing for flexible bit rate selection using a single compression scheme.

ZFP uses a software write-back cache of uncompressed blocks to enhance random access capabilities. This cache reduces the frequency of compression and decompression operations, thereby minimizing the need for compression or decompression with every data access. However, for the purpose of comparing compressors, we will disable this feature.

ZFP offers several advantages, including efficient compression and decompression, support for random access reads and writes, and the ability to specify the precise number of bits allocated to each array. The design of ZFP prioritizes computational simplicity and speed, making it suitable for potential hardware implementations. Its performance and applicability have been demonstrated in various domains, such as visualization, quantitative data analysis, and numerical simulation.[3]

II.II Blaz

Blaz is a lossy compression technique for 2D arrays of floating-point numbers that enables direct computation on compressed data without the need for decompression. It aims to reduce storage requirements and computational resources while maintaining acceptable accuracy. The compression scheme used by Blaz involves (1) block splitting, (2) block normalization, (3) prediction, (4) block transform, and (5) quantization. (Figure 1)

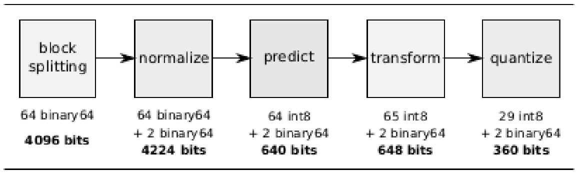


Figure 1: Overview of blaz compression scheme [4]

Block splitting is performed to divide the original matrix into smaller blocks, typically 8x8 in size. Block normalization is carried out to reduce the range of values within each block by computing the differences between consecutive elements. Prediction is then applied to approximate the slopes of the normalized block using a set of pre-defined slopes as references. The block is further transformed using a Type II two-dimensional discrete cosine transform (DCT) to aggregate large coefficients. Finally, quantization is performed by discarding small coefficients and keeping only the elements of the first two lines and columns of the transformed matrix. The compressed data structure of Blaz consists of 29 8-bit integers and 2 binary64 floating-point numbers per block, resulting in a compression rate of 11.37.

The compression scheme enables direct computation on compressed matrices using basic linear algebra algorithms. These algorithms facilitate operations such as addition, multiplication by a constant, dot product, and matrix multiplication to be performed directly on the compressed matrices.

Experimental results have shown that Blaz offers significant speedups in execution time compared to standard uncompressed matrices and other compressors, such as zfp. While Blaz may introduce a slightly higher loss of accuracy compared to zfp, the trade-off is deemed acceptable in many contexts. [4]

II.III PowerJoular

PowerJoular is a software power monitoring tool designed to help software developers, system administrators, and automated tools in understanding and analyzing the power consumption of their programs and devices.[5] The tool offers support for multiple platforms, including PCs, servers, and single-board computers like Raspberry Pi. For servers, it utilizes the Intel RAPL interface through Linux to monitor power consumption. Additionally, PowerJoular is written in Ada, a programming language renowned for its energy efficiency. This choice of language ensures that PowerJoular is a low-impact tool that actively contributes to energy efficiency efforts.[5]

PowerJoular is designed to automatically detect the hardware configuration and supported modules of a system, allowing it to provide accurate power data. It utilizes the Intel RAPL power data through the Linux powercap interface to monitor the CPU power consumption. By aggregating power readings from various components such as CPU cores, integrated graphics, memory controller, and last level caches, PowerJoular calculates the overall power consumption of the system. If supported, it can also monitor GPU power consumption using NVIDIA's System Management Interface.[5]

PowerJoular offers a command-line interface for flexibility, allowing users to display power monitoring data on the terminal or write it to CSV files. The tool provides continuous and automated monitoring capabilities through its systemd service, enabling the collection of runtime power consumption data for servers and devices. It also allows integration with external frameworks or dashboards, facilitating the visualization and analysis of power data from multiple servers or Raspberry Pi devices.[5]

Furthermore, PowerJoular can monitor the power consumption of individual processes by specifying their process ID (PID) at runtime. This feature enables users to track power data for specific applications or software components.[5]

In conclusion, PowerJoular is a versatile power monitoring tool that offers multi-platform support, accurate power estimation models, and flexible data collection options. It aims to assist software developers and system administrators in understanding and optimizing the power consumption of their programs and devices, ultimately contributing to the development of energy-efficient software.

III Energy Measurments

III.I Methodology

C codes will be developed to implement various elementary matrix computations, including dot product, addition, multiplication by a matrix, and multiplication by a constant. These computations will be performed on matrices represented in blaz format, compressed using zfp, and in the uncompressed state for comparison.

In addition to elementary matrix computations, we will compare the energy consumption when combining multiple operations before re-compressing the data. This reflects real-world scenarios where matrices undergo sequential computations. We will focus on combining multiple additions or combining addition and multiplication by a matrix, as these operations are frequently encountered in matrix computations and can significantly affect energy consumption.

The specific methods employed to measure energy consumption, as well as a detailed explanation of what is being measured, are outlined in Section III.II.

A fixed-rate of 11.37 for blaz and adjusting zfp to match the same rate and a range of [16×16-8192×8192] matrix sizes will be considered to capture diverse computational scenarios. The experiments will be conducted multiple times to account for variations and ensure reliable results.

Energy Consumption Measurement: The PowerJoular tool [5] will be utilized to measure the energy consumption of each computation scenario. PowerJoular provides energy consumption measurements, enabling evaluation of the energy consumed during the execution of matrix computations.

We performed our experiments on a PC equipped with an Intel Xeon CPU E5-2603 v3 and 16GB of RAM. The PC was running Ubuntu 22.04.2 LTS with kernel version 5.19.0-46 and GCC version 11.3. In order to focus solely on measuring the energy consumption of the CPU, we disabled the GPU to prevent parallel execution during the experiment. By doing so, we ensured that the proccess in running in the CPU only and the energy measurements were specifically attributed to the CPU's power usage.

III.II Methods

III.II.1 Elementary Operations

Algorithm 1 Blaz Operations

```
Require: Two or One Compressed Matrix of size $N \times N$
  We Start PowerJoular simultaneously to measure energy
  for $l \leftarrow 0$ to $N_{loop}$ do
    perform operations on Matrices
  end for
```

Algorithm 2 Zfp Operations

```
Require: Two or One Compressed Matrix of size $N \times N$
  We Start PowerJoular simultaneously to measure energy
  for $l \leftarrow 0$ to $N_{loop}$ do
    Decompress Matrices
    perform operations on Matrices
    Compress the result matrix
  end for
```

Algorithm 3 No Compression Operations

```
Require: Two or One Uncompressed Matrix of size $N \times N$
  We Start PowerJoular simultaneously to measure energy
  for $l \leftarrow 0$ to $N_{loop}$ do
     perform operations on Matrices
  end for
```

III.II.2 Sequence of Operations

Algorithm 4 Blaz Operations

```
Require: A number $N_{operations} + 1$ of compressed Matrices of size $512 \times 512$
  We Start PowerJoular simultaneously to measure energy
  for $l \leftarrow 0$ to $N_{loop}$ do
     for $i \leftarrow 0$ to $N_{operations}$ do
        perform operation between current compressed Matrix and compressed result
  Matrix
     end for
  end for
```

Algorithm 5 Zfp Operations

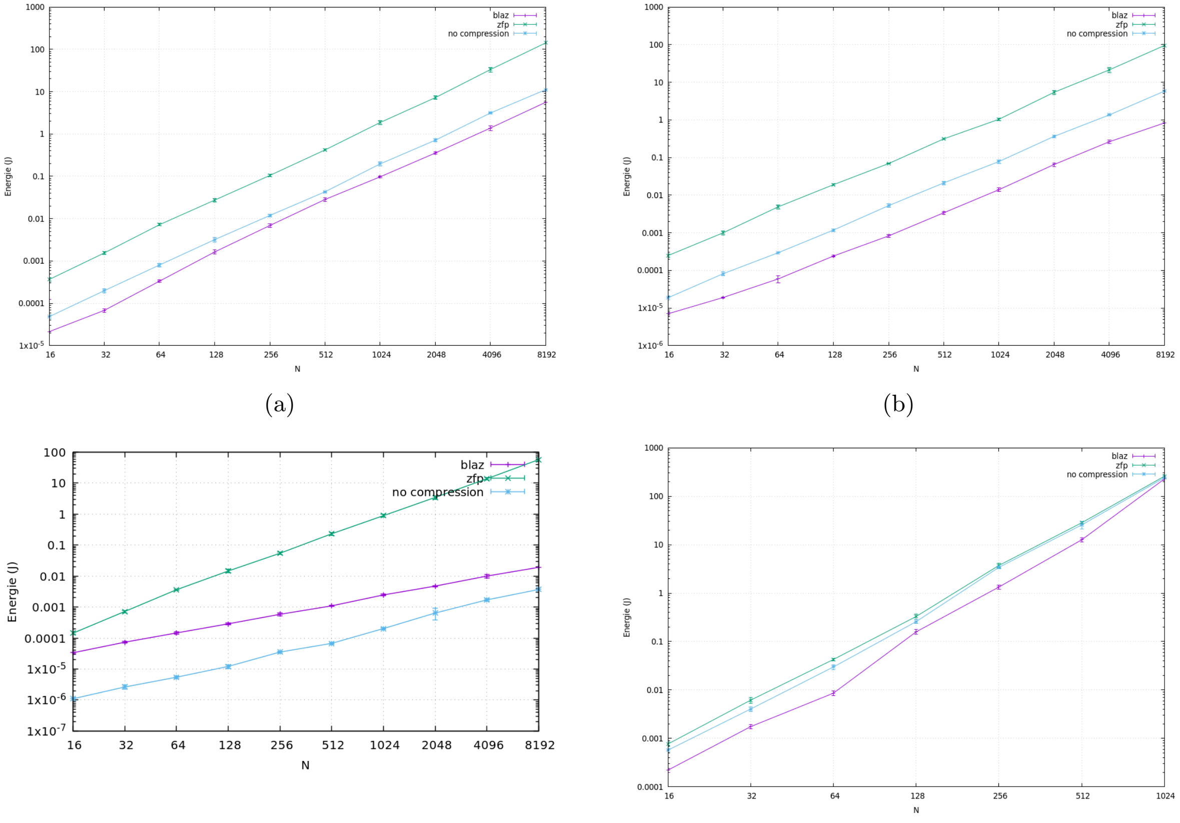
```
Require: A number $N_{operations} + 1$ of compressed Matrices of size $512 \times 512$
  We Start PowerJoular simultaneously to measure energy
  for $l \leftarrow 0$ to $N_{loop}$ do
     for $i \leftarrow 0$ to $N_{operations}$ do
        Decompress current matrix
        perform operation between current Matrix and result Matrix
     end for
     Compress result matrix
  end for
```

Algorithm 6 No Compression Operations

```
Require: A number $N_{operations} + 1$ of uncompressed Matrices of size $512 \times 512$
  We Start PowerJoular simultaneously to measure energy
  for $l \leftarrow 0$ to $N_{loop}$ do
      for $i \leftarrow 0$ to $N_{operations}$ do
          perform operation between current Matrix and result Matrix
      end for
  end for
```

III.III Results

The obtained energy consumption measurements of each operation for a $N \times N$ Matrix of double numbers are shown in figure 2 .



(a)

(b)

(c)

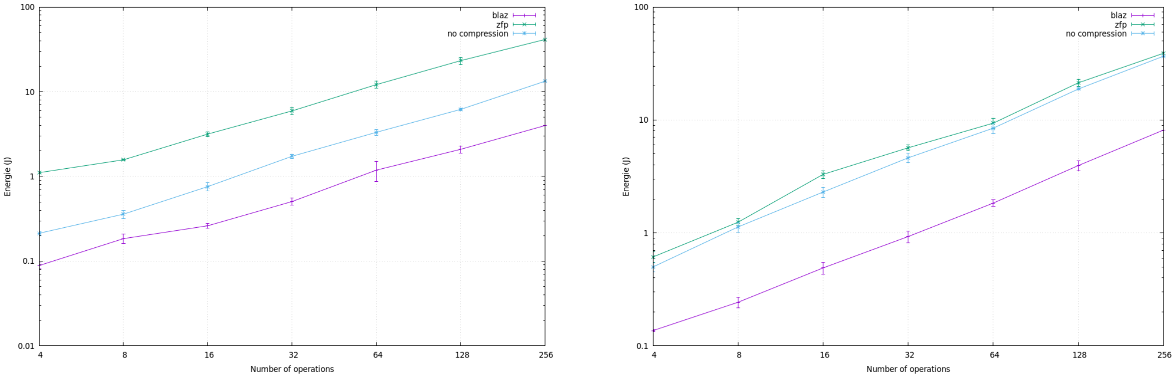
(d)

Figure 2: Energy consumption of operations in function of the size of the matrices (energy given in 10-logarithmic scale and N in 2-logarithmic scale). Top left: Addition. Top right: Multiplication by a constant. Bottom left: Dot product. Bottom right: Matrix multiplication.

The results of the experiments demonstrate that blaz compression can result in energy savings in matrix computations compared to both zfp compression and the uncompressed state, with the exception of the dot product operation. In addition and multiplication by a constant operations, blaz compression shows a more significant reduction in energy consumption. However, in the case of matrix multiplication, especially for large matrices, the advantage of blaz compression diminishes slightly, particularly for very large matrices.

As shown in Figure 2c blaz compression does not outperform the uncompressed state in terms of energy consumption for the dot product operation This can be attributed to the additional operations involved in the dot product algorithm when using blaz compression.

Figure 3 presents the results of the energy measurement for a sequence of operations. The experiment was conducted using a matrix size of 512×512 for the additions-only sequence and a matrix size of 128×128 for the combined addition and multiplication sequence.



(a) (b)

Figure 3: Energy consumption of a sequence operations (energy given in 10-logarithmic scale and Number of operations in 2-logarithmic scale). (a): Addition operations Only. (b): Addition and Multiplication sequence of operations

Blaz consistently demonstrates the lowest energy consumption across the various operations, indicating its superior energy efficiency compared to zfp and the uncompressed approach. This suggests that blaz's ability to directly compute on compressed data without the need for decompression and recompression significantly reduces the energy requirements for matrix computations.

On the other hand, zfp shows higher energy consumption compared to blaz, due to the additional computations involved in compressing and decompressing the data. The compression and decompression processes in zfp introduce overhead, resulting in increased energy consumption.

These findings support the potential of blaz as an energy-efficient compression technique for matrix computations.

IV Application to Sport Field: Football Match Analysis

In the context of sports analysis, especially in football, gaining insights into player movements and positioning is crucial for enhancing team performance, devising effective strategies, and making data-driven decisions. Analyzing player movement heatmaps is a valuable tool used by coaches, analysts, and sports scientists to understand player positions, patterns, and tactical behavior during matches.

A typical player movement heatmap consists of a 2D array of floating-point numbers, where each value represents the player's presence or activity level in a specific region of the football pitch. For instance, we can create a heatmap where each value corresponds to the average number of times a player has been present within a 10 cm^2 region of the pitch across the last 10 matches. Considering a standard football pitch size of 105 meters x 68 meters, the resulting heatmap array would approximately have dimensions of 1048 x 680.

In this study, to evaluate the energy efficiency of the compressors in computing the average player movement heatmap, we will generate random heatmaps simulating player positions for comparison. These random heatmaps will serve as a benchmark to assess the energy consumption of blaz and zfp compressors when processing player movement data from the last 10 matches.

The energy consumption measurements obtained from computing the average of ten floating-point arrays of size 1050 x 680 are shown in Table 1 .

	Blaz	Zfp	No Compression
Energy (mJ)	528 ± 53	5700 ± 530	2360 ± 162
Time (ms)	37 ± 0.08	420 ± 4	175 ± 1

Table 1: Energy consumption measurements to calculate the average of ten heatmaps.

Since blaz enables direct computations on the compressed data, it reduces the computational overhead and storage requirements, thereby leading to potential energy savings.

Assume that there are approximately 100,000 football matches played worldwide every year, each involving at least 10 players whose movement needs to be analyzed using player movement heatmaps. For simplicity, let's assume that the average energy consumption difference between blaz and zfp is 5 Joules, as shown in Table 1.

This means that by using blaz in football match analysis for all matches worldwide, the football community can potentially save approximately 5 megajoules of energy annually. To put this into perspective:
5 MJ is About 0.16% of the average US household energy consumption in a month It is also roughly equivalent to the energy generated by burning around 1.2 liter of gasoline.[2]

These numbers demonstrate the impact that adopting energy-efficient techniques like blaz can have, not only in football match analysis but in various other data-intensive fields as well.

V Conclusion

In conclusion, this research evaluated the energy consumption of matrix computations using compressed data, comparing blaz, zfp, and no compression scenarios. The results consistently showed blaz's superior energy efficiency, reducing energy consumption in addition, multiplication by a constant, dot product, and matrix multiplication operations compared to zfp and the uncompressed state. This demonstrates blaz's potential for sustainable computing practices.

The application of blaz and zfp in football match analysis also highlighted blaz's energy-saving advantage. By computing average player movement heatmaps, blaz outperformed zfp in terms of energy efficiency when processing data from the last 10 matches.

These findings are significant for promoting green computing and reducing the environmental impact of scientific data processing. Future research can explore the energy efficiency gains for different computations and larger matrices, optimizing compression techniques for diverse computational scenarios.

In summary, this study emphasizes the importance of energy-efficient approaches in scientific computing, providing valuable insights for developing sustainable and frugal computing solutions in the future.

References

[1] Sheng Di and Franck Cappello. “Fast error-bounded lossy HPC data compression with SZ”. In: *2016 ieee international parallel and distributed processing symposium (ipdps)*. IEEE. 2016, pp. 730–739.

[2] US EIA. *How much electricity does an American home use.* 2021.

[3] Peter Lindstrom. “Fixed-rate compressed floating-point arrays”. In: *IEEE transactions on visualization and computer graphics* 20.12 (2014), pp. 2674–2683.

[4] Matthieu Martel. “Compressed Matrix Computations”. In: *2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*. IEEE. 2022, pp. 68–76.

[5] Adel Noureddine. “PowerJoular and JoularJX: Multi-Platform Software Power Monitoring Tools”. In: *18th International Conference on Intelligent Environments (IE2022)*. Biarritz, France, June 2022.