ISSN: 2053-3578 I.F. 12.34

USE OF GRAPHICS PROCESSING UNITS IN DATA COMPRESSION ALGORITHMS

Xalilova Shahnoza Uchkunovna
Lecturer at Navoiy Digital Technology Technical School
Field: Computer Engineering ("Computer Systems Design")

Abstract. This article examines the application of graphics processing units in data compression algorithms, analyzing current approaches to parallelizing compression processes and evaluating their effectiveness compared to traditional central processor-based methods. The analysis reveals that graphics processing units demonstrate significant performance advantages in specific compression scenarios, particularly when processing large data volumes, though implementation complexity and hardware requirements present notable considerations.

Keywords: graphics processing units, data compression, parallel computing, GPU acceleration, lossless compression, lossy compression, CUDA

Аннотация. В данной статье рассматривается применение графических процессоров в алгоритмах сжатия данных, анализируются современные подходы к распараллеливанию процессов сжатия и оценивается их эффективность по сравнению с традиционными методами на основе центральных процессоров. Анализ показывает, что графические процессоры демонстрируют значительные преимущества в производительности в определённых сценариях сжатия, особенно при обработке больших объёмов данных, хотя сложность реализации и требования к аппаратному обеспечению представляют собой существенные факторы для рассмотрения.

Ключевые слова: графические процессоры, сжатие данных, параллельные вычисления, ускорение на GPU, сжатие без потерь, сжатие с потерями, CUDA, вычислительная эффективность.

Annotatsiya. Ushbu maqolada ma'lumotlarni siqish algoritmlarida grafik protsessorlardan foydalanish masalasi o'rganilgan, siqish jarayonlarini parallellashtirish bo'yicha zamonaviy yondashuvlar tahlil qilingan hamda ularning samaradorligi an'anaviy markaziy protsessorlarga asoslangan usullar bilan solishtirilgan. Tahlil shuni ko'rsatadiki, grafik protsessorlar ma'lum siqish stsenariylarida, ayniqsa katta hajmdagi ma'lumotlarni qayta ishlashda sezilarli ishlash afzalliklarini namoyish etadi, garchi amalga oshirish murakkabligi va apparat talablari muhim omillar hisoblanadi.

ISSN: 2053-3578 I.F. 12.34

Kalit so'zlar: grafik protsessorlar, ma'lumotlarni siqish, parallel hisoblash, GPU tezlashtirish, yo'qotishsiz siqish, yo'qotishli siqish, CUDA, hisoblash samaradorligi.

INTRODUCTION

The exponential growth of digital data generation across all sectors of modern society has created unprecedented demands for efficient storage and transmission solutions. Data compression algorithms serve as fundamental tools for addressing these challenges, reducing storage requirements and bandwidth consumption while preserving information integrity or acceptable quality levels. Traditional compression implementations relying on central processing units face inherent limitations when processing massive datasets, as sequential execution architectures struggle to meet real-time processing requirements in contemporary applications [1].

Graphics processing units, originally designed for rendering visual content, have emerged as powerful platforms for general-purpose computing tasks due to their massively parallel architecture. Modern GPUs contain thousands of processing cores capable of executing simultaneous operations, presenting opportunities for substantial acceleration of computationally intensive algorithms [2]. The application of GPU computing to data compression represents a convergence of algorithmic optimization and hardware advancement that merits systematic investigation.

METHODOLOGY AND LITERATURE ANALYSIS

The research methodology employed in this study follows a systematic literature review approach, examining peer-reviewed publications, conference proceedings, and technical documentation from domestic and international sources. Central processors optimize for sequential task execution with sophisticated branch prediction and cache hierarchies, while graphics processors prioritize throughput through massive parallelism with simpler individual cores [3]. Compression algorithms exhibit varying degrees of parallelization potential based on their inherent computational structures. Lossy compression techniques, particularly those employed in image and video processing, often demonstrate favorable parallelization characteristics due to block-based processing approaches that enable independent computation across data segments [4].

Research by Ozsoy and Swany demonstrates that dictionary-based methods such as LZ77 require careful algorithmic restructuring to exploit GPU parallelism effectively, as traditional implementations rely on sequential scanning and pattern matching that poorly suit massively parallel architectures [5]. The CUDA programming platform developed by NVIDIA has

emerged as the predominant framework for GPU-accelerated compression research, providing programming abstractions and optimization tools that facilitate algorithm implementation [6]. Karimov and Yusupov examined CUDA-based implementations of various compression algorithms, finding that memory bandwidth frequently constitutes the primary performance bottleneck rather than computational capacity, necessitating careful attention to data transfer optimization between host and device memory [7].

Comparative analyses in existing literature reveal that compression ratio achievements remain largely consistent between CPU and GPU implementations of identical algorithms, with GPU acceleration primarily affecting processing speed rather than compression effectiveness [8]. International research has extensively examined specific algorithm implementations, with entropy coding receiving particular attention due to its ubiquity across compression standards. Adaptive arithmetic coding and Huffman coding variants have been successfully parallelized through table-based approaches and parallel prefix operations, though achieving optimal performance requires balancing parallelism granularity against synchronization overhead [9].

RESULTS AND DISCUSSION

The synthesis of analyzed literature reveals consistent patterns regarding GPU-accelerated compression performance across diverse implementation contexts. Quantitative findings from multiple studies enable comparative evaluation of speedup factors and efficiency metrics, while qualitative analysis illuminates implementation considerations and deployment factors.

Table 1. Performance comparison of GPU-accelerated compression algorithms

Compression Algorithm	Average GPU Speedup Factor	Optimal Data Size Threshold	Primary Application Domain
JPEG encoding	15-25x	>2 megapixels	Image processing
H.264 video encoding	8-12x	>720p resolution	Video streaming
LZ77 variants	3-6x	>10 megabytes	General lossless
Huffman coding	5-8x	>1 megabyte	Entropy coding

Deflate algorithm	4-7x	>5 megabytes	Archive compression

The data presented in Table 1 synthesizes performance findings reported across analyzed sources, revealing substantial variation in acceleration factors across algorithm categories. Image and video compression algorithms demonstrate the highest speedup factors, reflecting their inherent parallelization suitability through block-based processing structures. The discrete cosine transform operations fundamental to JPEG encoding distribute naturally across GPU cores, enabling near-linear scaling with available computational resources under optimal conditions [4]. Video encoding benefits similarly, with motion estimation and compensation stages offering additional parallelization opportunities that compound acceleration effects.

Lossless compression algorithms exhibit more modest speedup factors, consistent with theoretical expectations regarding sequential dependencies inherent to dictionary-based methods. The 3-6x acceleration range for LZ77 variants represents significant improvement over sequential execution while acknowledging fundamental algorithmic constraints that limit parallelization potential. Implementation complexity for these algorithms substantially exceeds that required for embarrassingly parallel image processing operations, requiring sophisticated approaches to dependency management and workload distribution.

Table 2. Implementation factors affecting GPU compression efficiency

Factor Category	Impact Level	Mitigation Complexity	Research Consensus
Memory transfer overhead	High	Medium	Strong agreement
Thread divergence	Medium	High	Moderate agreement
Hardware availability	High	Low	Strong agreement
Algorithm adaptation effort	Medium	High	Strong agreement
Power consumption	Medium	Low	Limited research

Table 2 presents analysis of implementation factors influencing practical GPU compression deployment, synthesized from challenges and considerations discussed across reviewed literature. Memory transfer overhead emerges as the most consistently identified performance limitation, as data movement between host system memory and GPU device memory introduces latency that can substantially diminish net acceleration benefits for smaller datasets [7]. This factor explains the data size thresholds identified in Table 1, below which CPU-based compression may achieve superior overall performance despite slower per-operation execution.

Thread divergence, occurring when parallel threads follow different execution paths due to conditional branching, presents medium impact with high mitigation complexity. Compression algorithms frequently employ conditional logic for pattern matching and encoding decisions, creating divergence scenarios that reduce effective parallelism. Researchers have developed branch-free algorithm variants and predication techniques to address this challenge, though such modifications increase implementation complexity and may alter computational characteristics [10].

Hardware availability considerations encompass both physical GPU presence and capability matching between algorithm requirements and available resources. While GPU computing has achieved broad adoption in data centers and high-performance computing environments, edge computing and embedded systems may lack suitable hardware, constraining deployment contexts. Power consumption emerges as an underexplored factor in reviewed literature, with limited research examining energy efficiency comparisons between CPU and GPU compression implementations despite relevance to mobile and resource-constrained environments.

CONCLUSION

This systematic analysis of graphics processing unit application in data compression algorithms reveals substantial acceleration potential alongside meaningful implementation considerations. GPU-accelerated compression demonstrates speedup factors ranging from 3x to 25x depending on algorithm characteristics, with image and video compression exhibiting superior parallelization suitability compared to lossless dictionary-based methods. Memory transfer overhead emerges as the primary practical limitation, establishing minimum data size thresholds below which GPU acceleration provides diminished benefits.

The findings support strategic GPU deployment for high-throughput compression applications processing large data volumes, while acknowledging that traditional CPU-based

implementations remain appropriate for smaller-scale or resource-constrained contexts. Future research directions should address energy efficiency comparisons, hybrid CPU-GPU optimization strategies, and emerging hardware architectures that may alter current performance relationships. As data generation continues accelerating across all sectors, GPU-accelerated compression will likely assume increasing importance in meeting processing demands while managing storage and transmission resource constraints.

REFERENCES

1. Salomon, D. Data Compression: The Complete Reference / D. Salomon. – London: Springer-Verlag, 2007. – 1092 p.
2. Kirk, D.B. Programming Massively Parallel Processors: A Hands-on Approach / D.B. Kirk, W.W. Hwu. – San Francisco: Morgan Kaufmann, 2016. – 576 p.
3. Владимиров, А.А. Параллельные вычисления на графических процессорах / А.А. Владимиров, С.С. Петров // Вестник информационных технологий. – 2019. – № 4. – С. 45-52.
4. Nvidia Corporation. CUDA C++ Programming Guide. – Santa Clara: Nvidia, 2023. – 428 p.
5. Ozsoy, A. CULZSS: LZSS Lossless Data Compression on CUDA / A. Ozsoy, M. Swany // IEEE International Conference on Cluster Computing. – 2011. – P. 403-411.
6. Sanders, J. CUDA by Example: An Introduction to General-Purpose GPU Programming / J. Sanders, E. Kandrot. – Boston: Addison-Wesley, 2010. – 312 p.
7. Каримов, Ф.И. Оптимизация алгоритмов сжатия данных на основе CUDA / Ф.И. Каримов, Б.Т. Юсупов // Информатика ва энергетика муаммолари. – 2020. – № 2. – С. 78-85.
8. Balevic, A. Parallel Variable-Length Encoding on GPGPUs / A. Balevic // Euro-Par 2009 Parallel Processing Workshops. – Berlin: Springer, 2010. – P. 26-35.
9. Расулов, М.Х. Маълумотларни сиқиш усулларининг замонавий ёндашувлари / М.Х. Расулов // Ахборот технологиялари журнали. – 2021. – № 3. – С. 112-119.
10. Patel, R.A. Parallel Lossless Data Compression on the GPU / R.A. Patel, Y. Zhang, J. Mak // Innovative Parallel Computing Conference. – 2012. – P. 1-9.