Innovative: International Multi-disciplinary Journal of Applied Technology
(ISSN 2995-486X) VOLUME 04 ISSUE 04, 2026

Integrated GPU-Based Modeling of Medical Database Management Systems

Rakhimov Bakhtiyar Saidovich
Head of the Department of Biophysics and information technologies of Urgench state Medical Institute, Uzbekistan
E-mail: bahtiyar1975@mail.ru

Sobirova Sabokhat Kabulovna
Senior teachers of the Department of Biophysics and Information Technologies of Urgench State Medical Institute, Uzbekistan

Rakhimova Feroza Bakhtiyarovna
Senior teachers of the Department of Biophysics and Information Technologies of Urgench State Medical Institute, Uzbekistan
E-mail: feroza1971@gmail.com

Allayarova Asal Akabarovna
Senior teachers of the Department of Biophysics and Information Technologies of Urgench State Medical Institute, Uzbekistan

Saidov Atabek Bakhtiyarovich
Senior teachers of the Department of Biophysics and Information Technologies of Urgench State Medical Institute, Uzbekistan

Abstract:

The rapid growth of medical data volumes and the increasing complexity of analytical tasks require database management systems (DBMS) with high computational efficiency, scalability, and reliability. Modern information technologies, particularly graphics processing units (GPUs), provide powerful tools for accelerating data-intensive operations through massive parallelism. This paper presents an integrated overview of GPU architectures and their application in modeling and optimizing medical database management systems. We analyze key GPU architectural solutions (G80, GT200, and Fermi), the CUDA parallel programming model, and memory hierarchies, and demonstrate their relevance for medical data storage, processing, and signal analysis. The proposed integrated approach enables efficient processing of medical signals and images, supports large-scale databases, and reduces computational and economic costs in internet-based medical applications.

Keywords: Medical Databases, GPU Computing, CUDA, Parallel Processing, Information Technology, Medical Data Modeling.

Introduction

The digital transformation of healthcare has led to an exponential increase in medical data generated from diagnostic devices, imaging systems, laboratory equipment, and electronic health records. Efficient storage, processing, and analysis of such data are critical for decision support, diagnostics, and research. Traditional CPU-based database systems often struggle to meet performance requirements when processing large volumes of heterogeneous medical data [1].

In recent years, graphics processing units (GPUs) have evolved from specialized graphics accelerators into powerful parallel computing devices. Their architecture, based on a large number of lightweight processing cores and high memory bandwidth, makes them well suited for data-parallel workloads common in medical databases, signal processing, and computer vision. This paper integrates concepts from GPU architecture analysis and medical database modeling to present a unified framework for designing high-performance medical DBMS [2].

Materials and Methods

This study adopts an integrated analytical and simulation-based research methodology. The research combines a systematic literature review of GPU architectures and medical DBMS design with theoretical modeling of parallel computing approaches applicable to medical data workloads. Three NVIDIA GPU architectural generations — G80, GT200, and Fermi — were selected as representative platforms for comparative architectural analysis, focusing on core count, memory hierarchy, register capacity, and support for double-precision arithmetic. The CUDA parallel programming model was examined as the primary software framework, with attention to thread organization (warps, blocks, grids), memory types (shared, global, constant, texture), and kernel execution patterns [3].

Medical DBMS requirements were analyzed using a functional decomposition approach, identifying core operations including data storage, concurrent access, search, retrieval, and security enforcement. Parallel algorithm design principles were applied to map typical medical data processing pipelines — including signal acquisition, preprocessing, feature extraction, segmentation, and interpretation — onto GPU execution models. Mathematical modeling tools, including spline theory, fast spectral transforms, and matrix methods, were incorporated to evaluate signal representation and compression strategies. Simulation environments, specifically MATLAB and Simulink, were used to prototype and validate selected signal processing algorithms under GPU-accelerated conditions. Published experimental results from prior studies by the authors, including parallel algorithm benchmarks and DBMS modeling outcomes, were incorporated to support the analysis. The overall methodology follows a design science approach, combining conceptual framework development with performance-oriented evaluation of integrated GPU-DBMS architectures for medical applications [4].

1. Medical Database Management Systems

Medical database management systems are designed to store, manage, and retrieve structured and unstructured healthcare data while ensuring data integrity, security, and availability. A medical database typically contains patient records, diagnostic results, medical images, and time-series signals [5]. A database (DB) is an organized collection of interrelated data that accurately reflects the state and properties of the modeled objects. Medical databases may be logical (conceptual structure and relationships) or physical (actual storage implementation). Relational databases, composed of tables with rows and columns, remain widely used due to their consistency and flexibility [6].

Key requirements for medical DBMS include:

- Independence from data format changes;
- Efficient search and retrieval mechanisms;
- Support for concurrent multi-user access;
- High reliability and data recovery capabilities;

- Secure storage and controlled access to sensitive medical information.

Modern DBMS platforms such as MySQL, Firebird, MS Access, and other client–server systems are widely applied in healthcare. However, their performance can be significantly enhanced by integrating parallel computing technologies [7].

2. GPU Architectures for Data-Intensive Applications

2.1 Evolution of GPU Architectures

The NVIDIA G80 architecture, introduced in 2006, marked a turning point in general-purpose GPU computing. By abstracting graphics-specific components, the GPU can be viewed as a parallel computing system composed of:

- A flow control unit managed by the host CPU;
- Multiple streaming multiprocessors (SMs) operating in parallel;
- A hierarchical memory subsystem including global memory and cache levels.

Each streaming multiprocessor integrates scalar processors, special function units, shared memory, registers, and control logic. Threads are executed in groups known as warps, enabling SIMD-style parallelism with minimal synchronization overhead [8].

Subsequent architectures, such as GT200 and Fermi, further increased the number of processing cores, register capacity, and cache sizes. The Fermi architecture introduced a unified address space, enhanced double-precision performance, and configurable shared memory, making GPUs more suitable for scientific and medical computations.

2.2 CUDA Programming Model

CUDA (Compute Unified Device Architecture) provides a programming framework for developing parallel applications on GPUs. In this model, the CPU (host) executes sequential code and launches parallel kernels on the GPU (device). Kernels consist of many threads organized into blocks and grids, allowing scalable execution across available multiprocessors [9].

CUDA defines several types of memory, including registers, local, shared, global, constant, and texture memory. Effective utilization of this memory hierarchy is essential for achieving high performance in medical data processing tasks.

3. Parallel Processing in Medical Data Analysis

Many medical data processing tasks exhibit inherent data parallelism. These include image transformations, filtering operations, statistical analysis, and signal processing. Such tasks can be efficiently mapped onto GPU architectures.

Typical stages of computer vision and signal processing in medical applications include:

1. Data acquisition (images or signals);
2. Preprocessing and noise reduction;
3. Feature extraction;
4. Segmentation or detection;
5. High-level interpretation and analysis.

Most of these stages can be implemented using parallel algorithms, significantly reducing processing time. GPU-accelerated implementations are particularly effective for spectral analysis, wavelet transforms, and spline-based modeling of medical signals [10].

4. Modeling and Simulation Approaches

The modeling of medical database systems and signal processing algorithms relies on mathematical and computational methods such as functional analysis, spline theory, numerical integration, matrix methods, and parallel computing theory. Simulation environments like MATLAB and Simulink provide flexible tools for developing and validating such models [11].

Fast spectral transformations based on orthogonal bases enable efficient representation and compression of medical signals. When combined with GPU acceleration, these methods allow real-time or near-real-time processing of large datasets, which is essential for modern medical information systems.

Results

The architectural analysis of GPU generations reveals a consistent progression in computational capability directly relevant to medical DBMS workloads. The G80 architecture established the

foundation of general-purpose GPU computing by introducing streaming multiprocessors (SMs) with SIMD-style thread execution in warps, enabling efficient parallelization of data-intensive operations. The GT200 successor expanded the number of processing cores and register capacity, substantially increasing throughput for parallel database queries and image processing tasks. The Fermi architecture further advanced the platform by introducing a unified address space, enhanced double-precision floating-point performance, and configurable L1/L2 cache hierarchies, making it well suited for the numerical precision requirements of medical signal analysis and large-scale data operations [12].

The functional decomposition of medical DBMS operations confirmed that the core tasks — structured data storage, concurrent multi-user access, search and retrieval, and security enforcement — can be mapped effectively onto GPU parallel execution models. Relational database operations involving large patient record sets, time-series signal tables, and medical image repositories exhibit strong data-level parallelism, making them suitable candidates for GPU acceleration. The CUDA programming model, with its thread-block-grid hierarchy and diverse memory types, provides a flexible framework for implementing these operations efficiently across available streaming multiprocessors [13].

The five-stage medical signal processing pipeline — acquisition, preprocessing, feature extraction, segmentation, and interpretation — was found to be highly amenable to GPU parallelization at each stage. Preprocessing operations such as noise reduction and normalization, feature extraction via fast spectral transforms and wavelet decompositions, and segmentation tasks involving pixel-level or sample-level decisions all exhibit high arithmetic intensity and can be parallelized across thousands of GPU threads simultaneously [14]. Simulation results using MATLAB and Simulink prototypes demonstrated that GPU-accelerated spline-based signal reconstruction and orthogonal spectral transforms significantly reduce latency compared to sequential CPU implementations, enabling near-real-time processing of continuous medical data streams. The integration of these parallel signal processing modules with GPU-resident database operations supports a coherent, high-throughput medical information pipeline, reducing the need for repeated data transfers between CPU memory and GPU memory — a key bottleneck in heterogeneous computing systems [15].

Discussions

Integrating GPU-based parallel computing into medical DBMS offers several advantages:

- Significant reduction in computation time for data-intensive tasks;
- Improved scalability for large medical datasets;
- Enhanced support for advanced analytical methods, including computer vision and signal processing;
- Reduced operational costs for internet-based medical applications.

At the same time, careful algorithm design is required to address limitations related to memory access patterns, synchronization, and block size constraints.

Conclusion

This study presents an integrated perspective on the use of GPU architectures and parallel computing technologies in modeling medical database management systems. The evolution of GPU architectures and the CUDA programming model provide a solid foundation for accelerating medical data processing and analysis. By combining efficient database design principles with GPU-based parallelism, it is possible to build high-performance, scalable, and reliable medical information systems that meet the growing demands of modern healthcare.

Future work may focus on optimizing specific medical workloads, extending support for heterogeneous computing environments, and enhancing data security mechanisms in GPU-accelerated medical databases.

References

[1] B. S. Rakhimov, M. S. Mekhmanov, and B. G. Bekchanov, "Parallel algorithms for the creation of medical database," *J. Phys.: Conf. Ser.*, vol. 1889, no. 2, p. 022090, 2021. doi: 10.1088/1742-6596/1889/2/022090.

[2] B. S. Rakhimov, F. B. Rakhimova, and S. K. Sobirova, "Modeling database management systems in

medicine," *J. Phys.: Conf. Ser.*, vol. 1889, no. 2, p. 022028, 2021. doi: 10.1088/1742-6596/1889/2/022028.

[3] B. Rakhimov and O. Ismoilov, "Management systems for modeling medical database," *AIP Conf. Proc.*, vol. 2467, p. 060031, 2022. doi: 10.1063/5.0089711.

[4] B. S. Rakhimov, G. T. Khalikova, O. R. Allaberganov, and A. B. Saidov, "Overview of graphic processor architectures in database problems," *AIP Conf. Proc.*, vol. 2467, p. 020041, 2022. doi: 10.1063/5.0092848.

[5] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik, and O. O. Storaasli, "State-of-the-art in heterogeneous computing," *Scientific Programming*, vol. 18, no. 1, pp. 1–33, 2010.

[6] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2011.

[7] A. S. Tanenbaum, *Modern Operating Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[8] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," *Proc. IEEE*, vol. 96, no. 5, pp. 879–899, 2008. doi: 10.1109/JPROC.2008.917757.

[9] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: A unified graphics and computing architecture," *IEEE Micro*, vol. 28, no. 2, pp. 39–55, 2008. doi: 10.1109/MM.2008.31.

[10] J. Nickolls and W. J. Dally, "The GPU computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, 2010. doi: 10.1109/MM.2010.41.

[11] V. Garcia, E. Debreuve, and M. Barlaud, "Fast k nearest neighbor search using GPU," in *Proc. IEEE CVPR Workshops*, Anchorage, AK, USA, 2008, pp. 1–6. doi: 10.1109/CVPRW.2008.4563100.

[12] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded GPU using CUDA," in *Proc. 13th ACM SIGPLAN Symp. Principles and Practice of Parallel Programming (PPoPP)*, Salt Lake City, UT, USA, 2008, pp. 73–82.

[13] NVIDIA Corporation, *CUDA C Programming Guide*, version 12.0. Santa Clara, CA, USA: NVIDIA, 2023. [Online]. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide

[14] W. W. Hwu, Ed., *GPU Computing Gems Emerald Edition*. Burlington, MA, USA: Morgan Kaufmann, 2011.

[15] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA," in *Proc. 14th Int. Conf. High Performance Computing (HiPC)*, Goa, India, 2007, pp. 197–208. doi: 10.1007/978-3-540-77220-0_21.