Journal of Combinatorial Mathematics
and Combinatorial Computing
www.combinatorialpress.com/jcmcc

Combinatorial Press

Research on optimization method of multithreaded computing architecture for large-scale digital media data processing

Tao Liu[1,✉], Meiling Yang[2]

[1] *Department of Journalism and Communication, Anhui Vocational College of Press and Publishing, Hefei, Anhui, 230601, China*

[2] *Hefei Transportation Comprehensive Administrative Law Enforcement Detachment, Hefei Transportation Bureau, Hefei, Anhui, 230601, China*

ABSTRACT

In recent years, with the rapid development of information technology, the traditional single-threaded processing method can no longer meet the rapid growth of digital media data volume. In this paper, based on the digital media data processing system based on BS structure, the GPGPU parallel processing architecture is used for optimization. The access efficiency of massive parallel multithreading is ensured by executing a multilevel storage architecture composed of behavior decision unit, branch merge unit and branch recovery stack. The study designs the computational resource pool as well as the storage resource pool to form an infrastructure solution to the data processing problem. The query performance of the digital media data processing system using the GPGPU microarchitecture with multithreaded parallel processing is improved by about 81% and 69% or so compared to the Ocelot and prototype systems, respectively. And the average execution time for performing dynamic data allocation is 5.17s less than that of the original system. It shows that the optimized digital media data processing system has better data processing efficiency.

Keywords: Data processing system, GPGPU microarchitecture, multithreading, digital media

1. Introduction

With the rapid development of technology, digital media has an increasing impact on our lives. For example, globalized political participation, control of violence, educational content creation, and cognitive change [1-4]. Digital media is a form of media based on digital information technology for content production and dissemination [5]. It includes audio, video, image, text and other forms, and covers a variety of channels such as the Internet, cell phone mobile platforms, digital television, digital

✉ Corresponding author.
E-mail address: liutao5268@163.com (T. Liu).
Received 25 January 2024; Revised 15 March 2024; Accepted 05 December 2024; Published Online 16 April 2025.
DOI: 10.61091/jcmcc127b-518

radio and other channels in terms of dissemination methods [6-8]. It can be seen that there is a lot of data, and the processing of these data has become a research hotspot. Based on this series of data processing process, a powerful programming must be needed to process these data to ensure the good functioning of the system, and multi-threaded programming has this ability.

Multithreading is the process of allowing multiple streams of instructions to be executed concurrently in a program, each of which is called a thread. Threads are also known as lightweight processes which have independent execution control like processes and are scheduled by the operating system [9]. The difference between the two is that threads do not have an independent storage space, but share a storage space with other threads in the process to which they belong, which makes the communication between threads much simpler than processes. The execution of multiple threads is concurrent, that is, logically simultaneous, regardless of whether they are physically simultaneous or not [10-11]. If the system has only one central processing unit (CPU), then true simultaneity is impossible, but since the CPU is so fast that the user does not feel the difference, we don't have to care about it, we just have to envision that the individual threads are executing at the same time. The biggest difference between multithreading and traditional single-threading in program design is that, because the control flow of each thread is independent of each other, so that the code between each thread is executed in a chaotic order, which leads to thread scheduling, synchronization and other problems [12]. In contrast, multithreaded computing architecture is the simultaneous task processing or operations performed by multiple threads in a given program based on sharing the program's resources and memory space [13-15]. It is also because in multithreaded processing, each thread can perform tasks independently without interfering or blocking each other, and can share the program's resources such as memory, files, and network connections, thus improving the concurrency and responsiveness of the program. It occupies big advantages in handling huge data in digital media, such as improving system data processing capability and performance optimization, improving user experience, and simplifying code structure [16-19]. However, it also has some problems that need to be optimized by programmers. For example, multi-threaded computing architecture is difficult to program, resource consumption, and data synchronization problems [20]. Therefore, in this paper, we will optimize the multithreaded computing architecture based on large-scale digital media data processing for the purpose of research.

In this paper, we design a better performance GPGPU branch merge micro-architecture to optimize the digital media data processing system based on BS structure. The architecture consists of three sub-modules: the execution behavior decision unit, the branch merge unit and the branch recovery stack. For branch separation-intensive programs, the design supports the current optimal branch merging strategy to fully improve the computational resource utilization in a single cycle. At the same time, the design utilizes the completely uncorrelated execution paths of different branch separations to improve the execution performance of GPGPU. And the design of computational and storage resource pools further forms an infrastructure solution to the data processing problem.

2. Digital media data processing methods

With the rapid development of information technology, the traditional preservation of digital media resources can no longer meet the storage requirements of existing digital media resources. Large-scale oriented digital media data processing is mainly realized by digital media data processing system. The digital media data processing system based on BS structure mainly includes the functions of system management, data uploading, data downloading, data categorization and data retrieval [21]. After the user logs into the system, the system assigns different functional interfaces according to the user's rights, and the user can retrieve, download and upload data according to the specific needs, and the system administrator can manage the system in real time, data archiving, storage migration and user management.

2.1. Overall data flow analysis

The overall data flow of this system is shown in Figure 1. Users categorize the data to be uploaded through data classification, and the system stores them in different servers according to the type of uploaded data. When the user searches for the data, he/she searches through the data attribute table, and downloads and previews the data according to the found data.

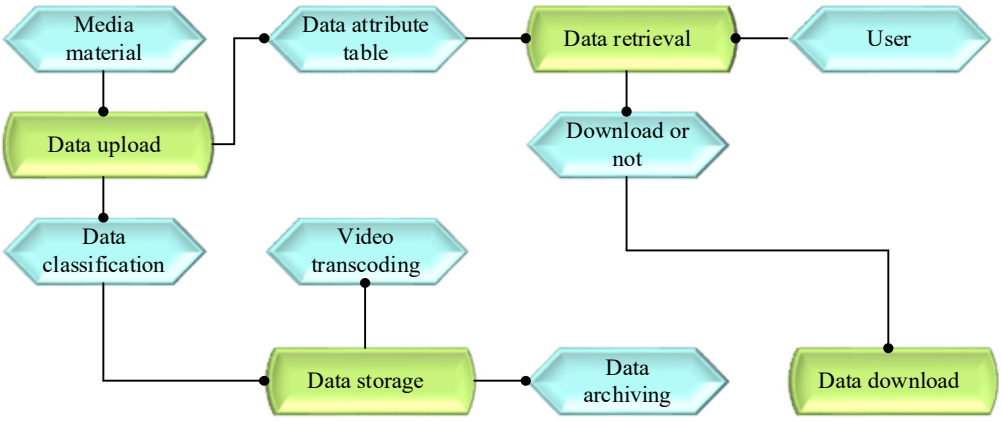

Fig. 1. The overall data flow of the system

2.2. Uploading data process analysis

The system's information upload data flow is shown in Figure 2. The underlying storage of the system includes document server, video server and database server. The system categorizes the data upload of the information and stores the different information in different servers, and stores the database through the categorization rules, so that when the system is retrieved and called, the location of the information storage is retrieved precisely through the search of the database.

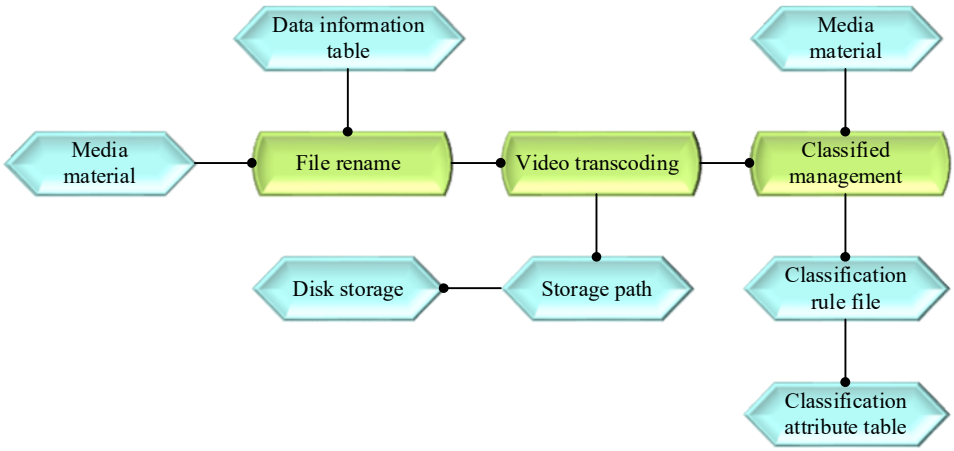

Fig. 2. The data of the system is uploaded to the data process

2.3. Download data flow analysis

The process of retrieving and downloading data is shown in Figure 3. The system can effectively solve the problems of downloading, long-distance transmission and sharing of multimedia data through the data retrieval function and data downloading function, so as to realize the effective use of digital multimedia resources.

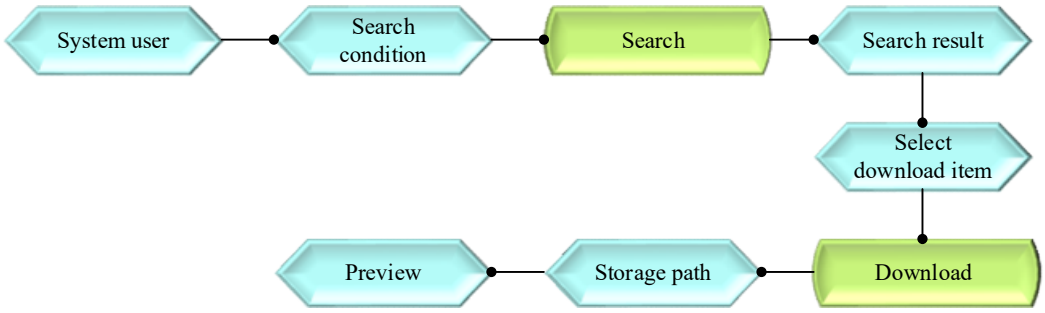

Fig. 3. Retrieve the download data process

3. Computational architecture optimization based on multithreaded parallelism

3.1. Microarchitecture design

In this paper, we optimize the previously studied BS architecture based digital media data processing system and propose to use the GPGPU parallel processing architecture to manage the execution of threaded tasks using a single instruction multithreaded execution model [22].

The overall design of the GPGPU micro-architecture consists of three sub-modules, namely, the execution behavior decision unit, the branch merge unit, and the branch recovery stack, and requires minor modifications to the warp issue stage of the warp scheduling unit, and the SIMT stack. The functions of each sub-module in the micro-architecture design are detailed below.

3.1.1. Implementing behavioral decision logic. The execution behavior decision logic consists of two TB-wide bitmasks, and in this paper, the execution behavior decision logic is used to mark the warps that can be alternatively scheduled under different instruction paths. The information of the bitmasks is updated by the branch recovery stack under specific conditions and is accessed by the warp scheduling unit in the transmitter stream at each cycle. In this paper, we use warp_mask_t to mark the warps that can be scheduled for execution under the main path, and a bit with a value of 1 in warp_mask_t indicates that the warp corresponding to this bit belongs to the main path, where the main path refers to the instruction path that has the highest priority for execution under the current cycle. In addition, this paper uses warp_assist_mask_t to mark the warps that can be scheduled for execution under all the remaining instruction paths except the main path, which is the biggest difference between this design and the previous branch-and-merge microarchitecture design. Each cycle, the WARP scheduler in the WARP ISSUE stream accesses the WARP_MASK_T information that executes the behavioral decision logic and sequentially determines whether each WARP marked in the WARP_MASK_T satisfies the conditions for launching execution. There are two kinds of judgment results: if there is a warp in warp_mask_t that satisfies the launch execution condition, the scheduling unit will take out the instruction in the instruction buffer corresponding to the warp, judge the type of the instruction, and launch the warp into the corresponding execution stream. If all the warps marked in warp_mask_t are in blocking state, the warp scheduling unit further accesses the information of warp_assist_mask_t to get the warps under the remaining instruction paths and further makes the judgment of launching the warp for execution. With the above design, the delay caused by the blocking of all warps under the main path can be masked by the scheduling execution of warps under the remaining instruction paths, and the parallel scheduling execution of sub-warps under multiple instruction paths is realized.

3.1.2. Branching and consolidation module. The branch merge unit caches information under different instruction paths when each warp executes a branch instruction, and performs branch merge when all warps in a TB are synchronized at a branch instruction, and finally uses the merged information to update the execution behavior decision logic and the branch recovery stack. The branch

merge unit consists of a two-entry lookup table and a warp merge unit (WCU), in which each entry of the lookup table contains four valid functional fields: the PC field denotes the address of the first instruction in the instruction path cached in this entry, and the RPC denotes the PC value of the recovery point of the instruction path, which can completely mark the beginning and end of an instruction path by using the PC field and the RPC field. The Active_mask function field is a TB-wide bitmask, which is used to mark the threads in the entire TB that are diverted to the current instruction path, and a bit with a value of 1 indicates that the corresponding thread belongs to the current control flow.The Mask_in_path function field is a mark for all warps in the TB. This function field is still a bitmask, and a bit with a value of 1 indicates that there are threads in the original warp corresponding to this instruction path. When the synchronization of all warps in the TB is completed, the lookup table transmits the information under the two instruction paths stored in its internal memory to the warp merge unit and performs branch merging. To ensure the merging efficiency, this paper also adopts the thread mapping strategy of channel replacement in the initialization phase of each warp. Finally, when both instruction paths have completed the branch merge operation, the branch merge unit will update the execution behavior decision logic to mark the subsequent warps that can be scheduled for execution under each instruction path, and control the branch recovery stack to press a new entry, which contains the information of the two instruction paths after the branch merge at the same time.

3.1.3. Branch Recovery Stack. The branch recovery stack obtains the PC and warp mask information of both instruction paths from the branch merge unit, constructs a new stack entry and presses it into the top of the stack, and each entry in the stack corresponds to one branch merge. In addition, when the execution of all warp tasks under one instruction path is completed, the branch recovery stack decides the subsequent warps that can be executed in parallel under the main path and the remaining paths, and updates the bitmask in the execution behavior decision logic. Unlike the previous branch merge strategies, the branch recovery stack in this design stores both instruction path information generated by the branch instruction in a single entry, and by accessing the top of the branch recovery stack it is possible to update both warp_mask_t and warp_assist_mask_t at the same time. The pri_mask and lat_mask functional domains are two warp-wide bitmasks that are used to mark the warps that can be scheduled for execution under the two paths after the branch merge. The Status function field is a 1-bit bit that is used to mark the instruction path that is currently prioritized for execution. A value of 0 indicates that the current priority execution is the main instruction path after the branch merge, and vice versa indicates that the current priority execution path is the sub-instruction path resulting from that branch merge. When all warps under an instruction path are executed, the branch recovery stack first determines the top of stack STATUS function field, and if the value is 0, it indicates that all thread tasks under the main instruction path have been executed. Next, the branch recovery stack first overwrites the pri_mask function field using the lat_mask function field of the top-of-stack entry, indicating that the current upcoming execution of the warp under the subinstruction path is prioritized. Immediately after that, this design updates the warp_mask_t and warp_assist_mask_t in the execution behavior decision logic based on the information under the sub-instruction path. Finally, this design sets the STATUS function field of the branch recovery top-of-stack entry to 1, indicating that the thread task under the main instruction path has been executed. If the value of status itself is 1, it means that both instruction paths of the stack entry pair have been executed. At this point, this design first pops the top of the branch recovery stack, and then uses the new top-of-stack information to directly update warp_mask_t and warp_assist_mask_t.

3.2. Parallel Processing Architecture Optimization Design and Implementation

3.2.1. Computing resource pool. The bottleneck of the digital media data processing system is mainly reflected in the data processing link, specifically for data loading and processing, so for the analysis and design of the computing resource pool of the system, on the one hand, using the CPU, memory, IO

channel virtualization resources to design and build a number of virtual servers, to provide a number of databases involved in the system's data collection platform, as well as the back two of these systems, the data processing, operation and management, and the deployment of the three types of applications for external services, on the other hand, the dynamic allocation of resources is achieved. Service three types of applications deployed, on the other hand, to achieve the dynamic allocation of resources, the key technology is virtualization technology and automated management tools [23].

The specific implementation is also divided into three levels, summarized as follows:

1) Physical resources. Digital media data processing system in the infrastructure platform optimization will use multiple high-end servers with virtualization special. The IBM minicomputer Power 780, which uses the Power7 chip, is a mainstream high-end server designed for the physical resource layer. Each configuration of 32 CPU, 128G memory, with the help of the server itself has the ability of hyper-threading, but also allows the deployment of the system on its recognition to more than 2 times the number of actual configuration of the CPU, in addition to the system is also configured with disk arrays, as well as other basic network, security equipment, constituting the system's physical resource layer.

2) Resource virtualization. The optimization plan uses CPU virtualization and IO virtualization technology to build a multi-processor and IO shared pool. Multiple virtual servers are divided according to system needs, and database and application software are deployed respectively. The virtual server's and network card, Fibre Channel card) is a logical resource allocation built on top of the server virtualization technology system. each virtual server does not exclusively enjoy fixed and resources like the physical partitioning used by HP servers used in production environments, but rather uses the CPU and IO resources in the resource pool on demand according to the server. the IBM minicomputer Power 780 is virtualized through the IBM PowerVM small machine virtualization technology provides a shared CPU pool, virtual I / O server, dynamic logical partitioning and online partition migration and other technologies, you can maximize the use of Power P780 physical server resources, dynamic deployment of business system workloads required CPU, memory, I / O capacity, and reduce the time of the system rock machine, enhance system flexibility.

3) Virtual Resource Pool Management. Virtual resource pool management (cloud computing management) is realized by dedicated management software, which can adjust the number of resources expected to be used by the virtual server online, migrate the virtual server to another physical server to run online, flexibly plan the scope and quota of the virtual resource pool, quickly deploy or clone the system by mirroring, and all operations can be conveniently carried out through the graphical interface.

3.2.2. Storage resource pool. After analyzing the optimization and integration of server resources, the storage layer, which provides basic capacity and access performance services to the servers, also requires exhaustive consideration and design.

First of all, the storage layer provides suitable storage space for the server layer. With the storage space, the server resources can effectively combine software, computation, and data to realize good business support. In terms of providing space, for the design requirements of each business, as well as adjustments and changes in the future at different stages of development, for example, a certain business module in the development and testing, on-line preparations, formal operation, update or replacement, withdrawal of services and other parts of the life cycle, which involves the allocation of different storage capacity, recycling, and the rational allocation and scheduling of resources, is the storage infrastructure design needs to be clear. Secondly, the allocated data storage space needs to have corresponding performance and data protection mechanisms, which directly affects the server's computing speed and service delivery quality. In the environment of multiple storage coexistence, and there are differences in demand, only through the initial planning to determine the distribution of storage performance, is not conducive to long-term development and change, especially when the need

to quickly respond to market demand, quickly adjust the allocation of storage resources, it will be very obvious that the time will be stretched. Third, the increase in storage space and performance should be flexible and strategic with standards. A centralized management platform is required for monitoring and management. In response to the system's requirements for storage, the virtualization of the storage system is realized and a unified storage resource pool planning is established. Through the establishment of storage resource pool, it solves the problem of hot spots in some disks that existed before, and at the same time, it closely matches the computing resource pool to realize the advanced, stable, flexible and scalable infrastructure, which provides a prerequisite guarantee for the standardization of the establishment of resources. The pooling of hardware (server, storage) resources is realized through virtualization technology, which is a step forward for the whole data center to cloud technology data center, and the pool of resources provides highly available protection for the last application.

4. Analysis of the optimization effect of computing architecture based on multi-threaded parallelism

4.1. Data sets

This paper uses the SSB test benchmark to validate the query performance of a digital media data processing system optimized with a multi-threaded parallel computing architecture.The SSB dataset contains five tables in a star schema: one fact table and four dimension tables (customer, part, date, supplier) and 13 standard SQL query test statements.

In this paper, the RS dataset is used to test the data allocation performance. The dataset contains two tables, Table R and Table S. The primary key of Table R encompasses 16×106 tuples, each with a unique value. Table S generates five different data sizes with 4×106,8×106,16×106,32×106,64×106 primary key tuples for each table.The dataset for each size of table S satisfies two distribution cases, uniform and high data skewed distribution. In the uniform distribution, each tuple has an equal probability of appearing in table S. In the skewed distribution, the tuple with the highest frequency of occurrence has a probability of occurrence of 22%, and the rest of the tuples have decreasing probabilities of occurrence, and the sum of the frequencies of the tuples with the top ten frequencies of occurrence is 55%.

4.2. System Performance Analysis

In order to verify the performance of the optimized multi-threaded parallel computing architecture designed in this paper for large-scale digital media data processing, four sets of experiments are designed and completed in this paper: (1) Comparing the execution time of selection operations of the digital media data processing system with the optimized GPGPU microarchitecture and the prototype system through experiments, to verify the effectiveness of the optimized GPGPU microarchitecture. (2) The multi-threaded operation performance of the digital media data processing system with optimized GPGPU microarchitecture and the prototype system under different expansion factors and different complex queries are compared through experiments to verify the efficiency of multi-threaded parallel operation. (3) Experimentally comparing the connection operation performance of the digital media data processing system with the optimized GPGPU microarchitecture and the prototype system under different scaling factors and different complex queries, to verify the effectiveness of the connection operation of the digital media data processing system with the optimized GPGPU microarchitecture. (4) Comparison of the performance of the prototype system with the optimized GPGPU micro-architecture digital media data processing system using parallel processing and the GPU query engine of Ocelot for different complex queries, to validate the impact of multi-threaded parallel processing on query performance.

Experiment 1: Comparing the performance of selection operation with different expansion factors

The experiments are conducted by setting different expansion factors for SSB dataset so as to get different data volumes. Then select operations in query statements are extracted for the experiment. The execution time comparison of the selection operations of the prototype system and the digital media data processing system using the optimized GPGPU micro-architecture under different expansion factors is shown in Fig. 4, with the horizontal axis representing the expansion factor SF (the same as below). The experimental results show that the performance of selection operation using the optimized GPGPU microarchitecture can be improved by 12.32% compared with the prototype system.

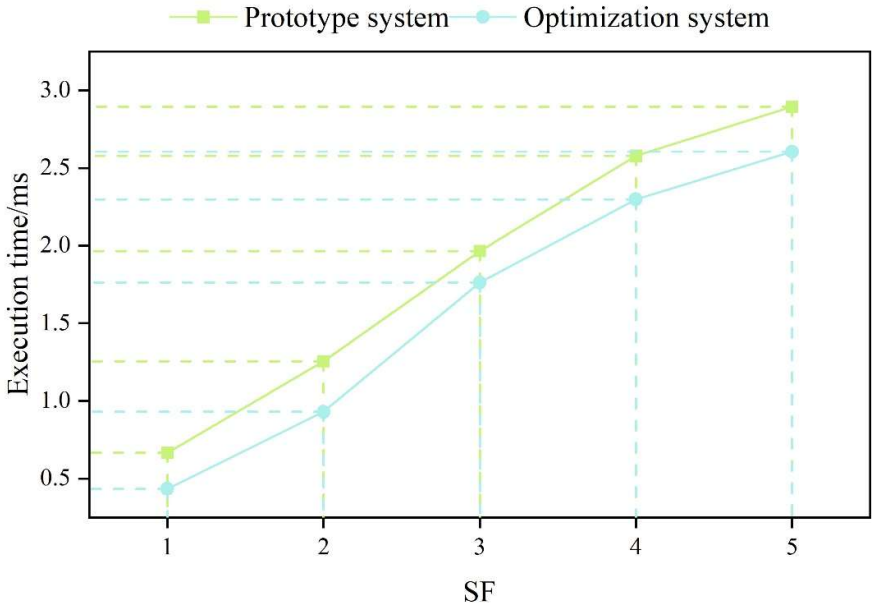

Fig. 4. Choice interoperability can be compared

Experiment 2: Comparing the performance of multi-threaded parallel operations with different expansion factors and different queries

The experiments are conducted by setting different expansion factors for the SSB dataset so as to obtain different data volumes. Then multi-threaded operations in query statements are extracted for experimental comparison. The performance comparison of multithreaded parallel operation between the prototype system and the digital media data processing system with optimized GPGPU microarchitecture under different expansion factors is shown in Fig. 5. Due to the division of data into individual processing units, grouped multithreaded processing requires only local processing and then global synchronization. The digital media data processing system with the optimized GPGPU microarchitecture outperforms the prototype system by 13.25% of the execution time for multithreaded parallel operation.

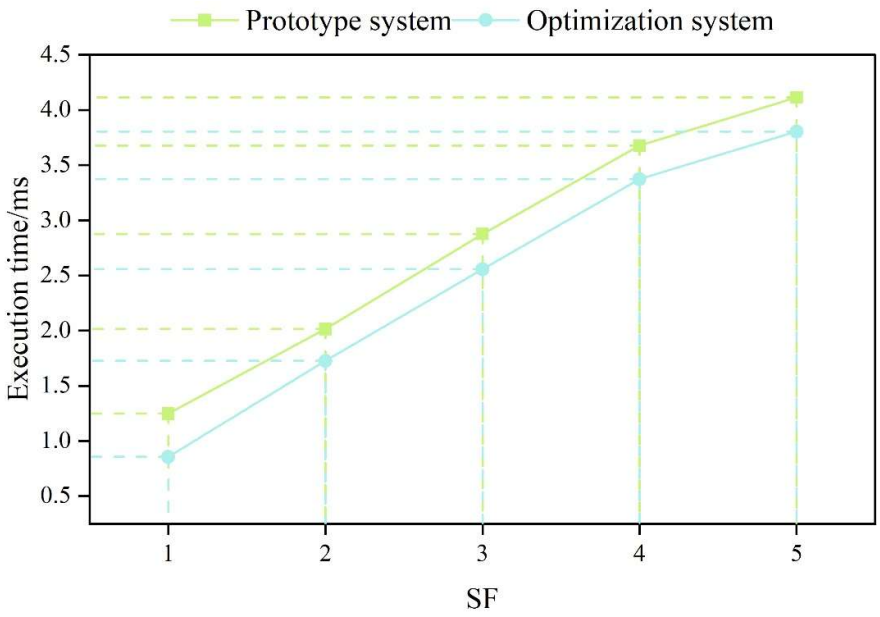

Fig. 5. Performance comparison of parallel operations

Experiment 3: Comparing the performance of join queries with different expansion factors and different queries

The experiments are conducted by setting different expansion factors for the SSB dataset so as to obtain different data volumes. Then the connection operations in query statements are extracted for experimental comparison. The performance comparison between the prototype system and the digital media data processing system with optimized GPGPU microarchitecture for parallel join operations under different scaling factors is shown in Figure 6. The data is distributed in each processing unit according to the connection characteristics, and the execution time is improved by 11.02% by distributing the connection-related data in the same processing unit to avoid cross-access between processing units. Therefore, the connection performance of the digital media data processing system using the optimized GPGPU microarchitecture is better than that of the prototype system.

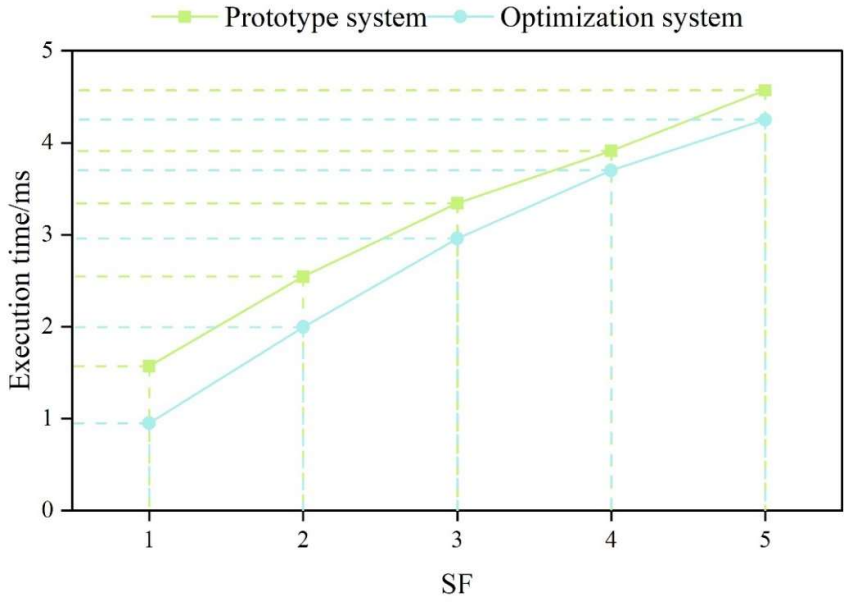

Fig. 6. Performance comparison of parallel connection operations

Experiment 4: Comparison of query performance between the prototype system and the GPGPU micro-architecture digital media data processing system based on multi-threaded parallel processing and the GPU query engine Ocelot.

The query execution times of the query statements Q1.1, Q1.2, and Q1.3 in the SSB test benchmarks are compared on the prototype system, the GPGPU micro-architecture-based digital media data processing system using multi-threaded parallel processing, and the GPU query engine Ocelot-based system, to validate the performance advantages of each system under different complex queries. The experimental results are shown in Fig. 7, the digital media data processing system with GPGPU microarchitecture using multithreaded parallel processing obtains some performance improvement under different complex queries, about 81% compared to Ocelot query performance, and about 69% compared to the prototype system query performance.

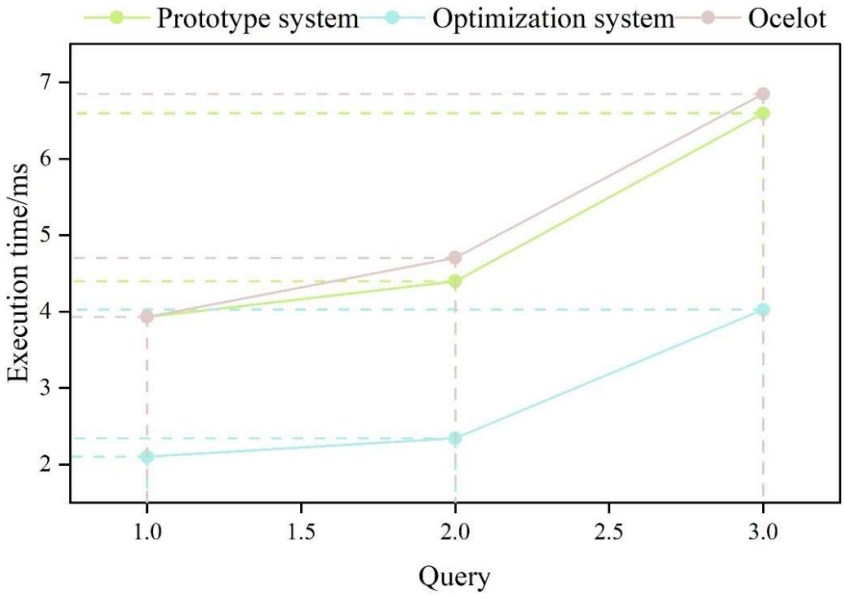

Fig. 7. Query performance contrast

4.3. Environmentally sensitive dynamic data distribution

In this subsection, the fine-grained step S1 of the grouping section is selected as the experimental object, a fixed data allocation ratio of 37% is selected, and the number of sub-processors allocated to R1 is fixed to 4 when sub-processors need to be re-allocated.The CPU occupancy of the database process is changed by employing the opening of additional processes to occupy the CPU, and comparing the execution time of the digital media data processing system using the optimization with the execution time of the prototype system The difference in execution time between the optimized digital media data processing system and the prototype system is compared. The execution time comparison is shown in Fig. 8, where the horizontal coordinates indicate the CPU occupancy of the extra processes. The average execution time of the optimized digital media data processing system and the prototype system are 45.52s and 50.69s respectively, which indicates that the optimized digital media data processing system is more efficient in execution.

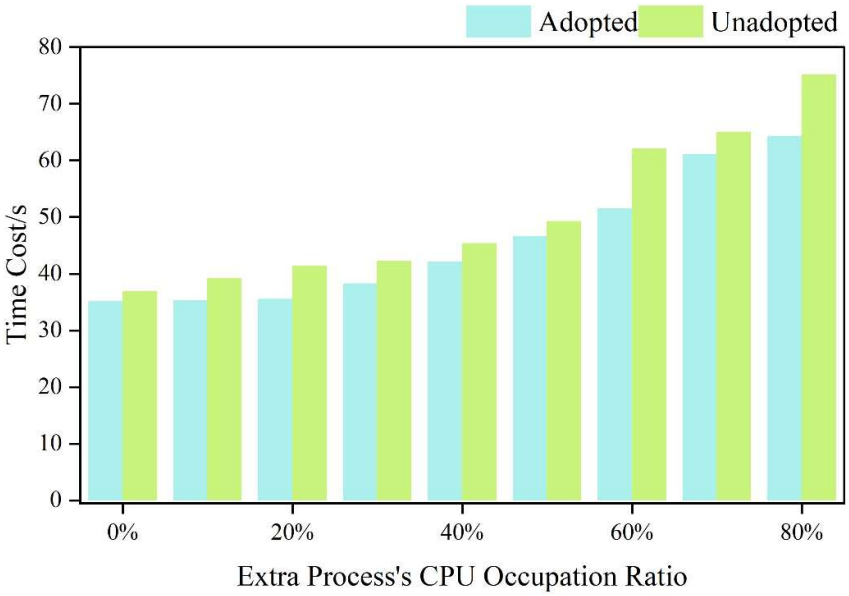

Fig. 8. Execution time contrast

The execution acceleration ratio is shown in Fig. 9. It can be seen that the average acceleration ratio using the optimized digital media data processing system is 1.07 times and the maximum acceleration ratio can reach 1.21 times. It shows that the performance advantage of the optimized digital media data processing system is extremely valuable.

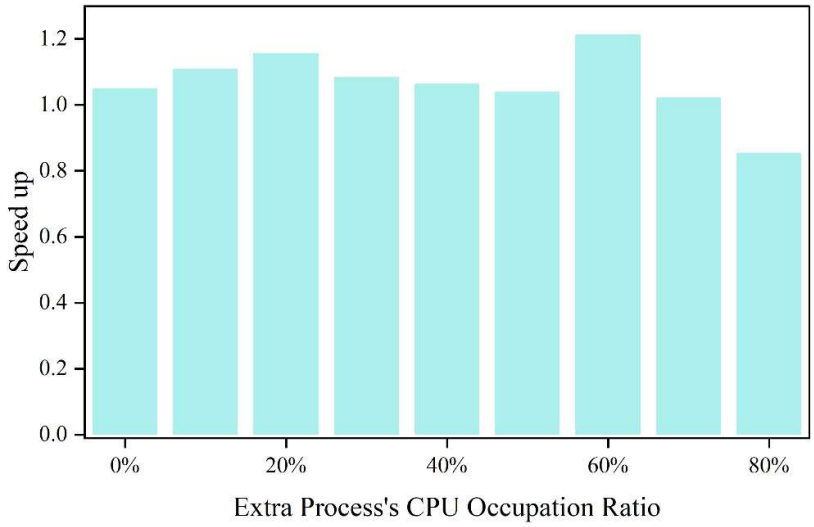

Fig. 9. Execution acceleration ratio

5. Conclusion

The study optimizes a digital media data processing system based on BS architecture and designs a digital media data processing system using GPGPU microarchitecture with multi-threaded parallel processing.

The performance of the optimized system is verified using SSB dataset. Compared to Ocelot and the prototype system, the optimized digital media data processing system designed in this paper has the best query performance, which is about 81% and 69% improvement compared to Ocelot and the prototype system, respectively.

Data allocation performance is tested using RS dataset. The optimized system is more efficient in

data allocation and the average execution time is 5.17s shorter than the original system.

References

[1] Lorenz-Spreen, P., Oswald, L., Lewandowsky, S., & Hertwig, R. (2023). A systematic review of worldwide causal and correlational evidence on digital media and democracy. Nature Human Behaviour.

[2] Dragiewicz, M., Burgess, J., Matamoros-Fernandez, A., Salter, M., Suzor, N. P., Woodlock, D., & Harris, B. (2018). Technology facilitated coercive control: domestic violence and the competing roles of digital media platforms. Feminist Media Studies, 609-625.

[3] Paskevicius, M. (2021). Educators as Content Creators in a Diverse Digital Media Landscape. Journal of Interactive Media in Education, 2021(1), NA-NA.

[4] Tsai, J. Y., Phua, J., Pan, S., & Yang, C. C. (2020). Intergroup Contact, COVID-19 News Consumption, and the Moderating Role of Digital Media Trust on Prejudice Toward Asians in the United States: Cross-Sectional Study. J Med Internet Res, 22(9), e22767.

[5] Webster, J. G. (2017). Three myths of digital media. Convergence, 23(4), 352-361.

[6] Twenge, J. M., Martin, G. N., & Spitzberg, B. H. (2019). Trends in US Adolescents' media use, 1976–2016: The rise of digital media, the decline of TV, and the (near) demise of print. Psychology of Popular Media Culture, 8(4), 329-345.

[7] Milton, J., & Cobelo, S. (2023). Translation, adaptation and digital media. Taylor & Francis.

[8] Carah, N., & Brodmerkel, S. (2021). Alcohol Marketing in the Era of Digital Media Platforms. Journal of Studies on Alcohol and Drugs, 82(1), 18-27.

[9] Yosifovich, P., Ionescu, A., Russinovich, M. E., & Solomon, D. A. (2017). Windows Internals: System architecture, processes, threads, memory management, and more, Part 1. Microsoft Press.

[10] Giebas, D., & Wojszczyk, R. (2021). Detection of Concurrency Errors in Multithreaded Applications Based on Static Source Code Analysis. IEEE Access, 9, 61298-61323.

[11] Asyabi, E., Sharafzadeh, E., SanaeeKohroudi, S., & Sharifi, M. (2019). CTS: An operating system CPU scheduler to mitigate tail latency for latency-sensitive multi-threaded applications. Journal of Parallel and Distributed Computing, 133, 232-243.

[12] Jacobs, B., Bosnacki, D., & Kuiper, R. (2018). Modular termination verification of single-threaded and multithreaded programs. ACM Transactions on Programming Languages and Systems, 40(3), A12.

[13] Nemirovsky, M., & Tullsen, D. (2022). Multithreading architecture. Springer Nature.

[14] Agrawal, S. R., Idicula, S., Raghavan, A., Vlachos, E., Govindaraju, V., Varadarajan, V., ... & Sedlar, E. (2017, October). A Many-core Architecture for In-Memory Data Processing. In 2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO) (pp. 245-258). IEEE.

[15] Yaroshko, S. (2019). Multithreaded evolutionary computing. In 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering, UKRCON 2019-Proceedings (pp. 1041-1045).

[16] Aljawarneh, S., Yassein, M. B., & Talafha, W. A. A. (2018). A multithreaded programming approach for multimedia big data: encryption system. Multimedia Tools & Applications, 77(9).

[17] Que, Z., Nakahara, H., Fan, H., Meng, J., Tsoi, K. H., Niu, X., ... & Luk, W. (2020, December). A Reconfigurable Multithreaded Accelerator for Recurrent Neural Networks. In 2020 International Conference on Field-Programmable Technology, ICFPT 2020 (pp. 20-28). Institute of Electrical and Electronics Engineers Inc..

[18] Jin, X., Zhou, Y., Huang, B., Yu, Z., Zhan, X., Wang, H., ... & Bao, Y. (2019, June). Qosmt: supporting precise performance control for simultaneous multithreading architecture. In Proceedings of the ACM

International Conference on Supercomputing (pp. 206-216).

[19] Wong, A., Bucci, P., Beschastnikh, I., & Fedorova, A. (2024). Making Sense of Multi-threaded Application Performance at Scale with NonSequitur. Proceedings of the ACM on Programming Languages, 8(OOPSLA2), 2325-2354.

[20] Dosanjh, M. G., Grant, R. E., Schonbein, W., & Bridges, P. G. (2019). Tail queues: A multi-threaded matching architecture. Concurrency and Computation. Practice and Experience, 32(SAND-2019-1466J).

[21] Intelligence And Neuroscience Computational. (2023). Retracted: An Intelligent Digital Media Asset Management Model Based on Business Ecosystem. Computational intelligence and neuroscience9867948-9867948.

[22] Toan Nguyen Mau & Yasushi Inoguchi. (2018). Audio fingerprint hierarchy searching strategies on GPGPU massively parallel computer. Journal of Information and Telecommunication(3),265-290.

[23] Wenyan Yang & Yang Wenyan. (2020). Exploration of Cloud Computing Resource Pool Virtualization Technology and Implementation Methods. Journal of Physics: Conference Series(1),012001-.