Optimizing Acoustic Field Rendering Through Heterogeneous Computing

Zhang Sheng*[a], Huiling Zhu[a]
[a]Hefei University of Technology
*dcszhang@foxmail.com

ABSTRACT

The objective of this research is to enhance sound field rendering efficiency using a heterogeneous computing framework. By integrating GPU and CPU resources, we address performance challenges in complex environments and large-scale ray tasks. Evaluations of the NVIDIA RTX A6000 GPU and AMD 9754 CPU led to an ray task allocation mechanism, optimizing resource use and maximizing computational efficiency. Experimental results show that this strategy significantly accelerates sound field rendering, achieving nearly 400 times faster performance than traditional single-core CPU computations.

Keywords: sound field rendering, heterogeneous computing, parallel computing

1. INTRODUCTION

Noise significantly affects human health, causing hearing loss, sleep disturbances, cardiovascular diseases, cognitive impairments, and psychological issues[1]. Effective noise management requires accurate noise prediction to identify sources and guide urban planning and control measures.Traditional wave-based noise prediction methods, though accurate, are computationally expensive, particularly for high-frequency sounds in complex environments[2,3]. The Gaussian Beam Tracing (GBT) method addresses these issues by simulating sound as Gaussian beams, enhancing both accuracy and efficiency[4].Heterogeneous computing, combining CPUs and GPUs, accelerates tasks like sound field rendering. GPUs excel in parallel processing, while CPUs handle complex control logic. Optimal task distribution between CPUs and GPUs achieves load balancing and maximizes resource utilization.This study presents a high-performance computing architecture for the GBT algorithm, integrating Message Passing Interface (MPI) and Compute Unified Device Architecture (CUDA) technologies. GPU dynamic parallel acceleration in the Gaussian Beam Summation (GBS) process further enhances computational efficiency[5].

2. ARCHITECTURE

2.1 Gaussian Beam Tracing

Gaussian Beam Tracing (GBT) is an advanced geometric acoustics simulation technique used to predict and analyze the propagation characteristics of sound waves in complex environments. Compared to traditional geometric acoustics methods such as ray tracing, the GBT method introduces the concept of Gaussian beams, thereby more accurately simulating the complex interactions of sound waves with obstacles, including phenomena such as reflection, refraction, and diffraction.

Ray Tracing Module (RT): In GBT, the initial step is to compute sound wave propagation paths using straight-line acoustical ray tracing, assuming constant sound speed within each medium layer for approximately straight paths. As sound waves move between layers, their propagation direction shifts according to Snell's law due to refraction across varying sound speeds. Following basic path determination, GBT employs the DRT technique for accurate simulation of sound wave physical properties, including sound pressure, speed, and intensity at critical points, ensuring simulations accurately reflect sound wave interactions with the environment beyond simple geometric paths.

Gaussian Beam Summation (GBS): The final step involves reconstructing the entire sound field through Gaussian beam summation. The GBS method synthesizes the contributions of multiple Gaussian beams along different paths, calculates the sound pressure contribution of each Gaussian beam at specific observation points, and integrates the effects of all relevant Gaussian beams in integral form, ultimately yielding an accurate sound field distribution[6].

Through this series of steps, the GBT method can accurately simulate the propagation behavior of sound waves in complex environments, including phenomena such as refraction of sound waves in multilayer media, reflection, and diffraction between different obstacles. The application of this method has improved the efficiency and accuracy of acoustic simulations, particularly suitable for complex acoustic environments and sound field designs.

2.2 Preprocessing Stage

The program starts by loading input data and initializing the MPI environment, creating multiple MPI threads for parallel execution. The master thread (ID 0) manages task scheduling and result collection, coordinating the entire computational process, including GPU operations. Tasks are distributed to MPI threads based on the performance of GPUs and CPUs, ensuring efficient load balancing and preventing resource bottlenecks.The logical framework of the whole procedure is shown in Figure 1.

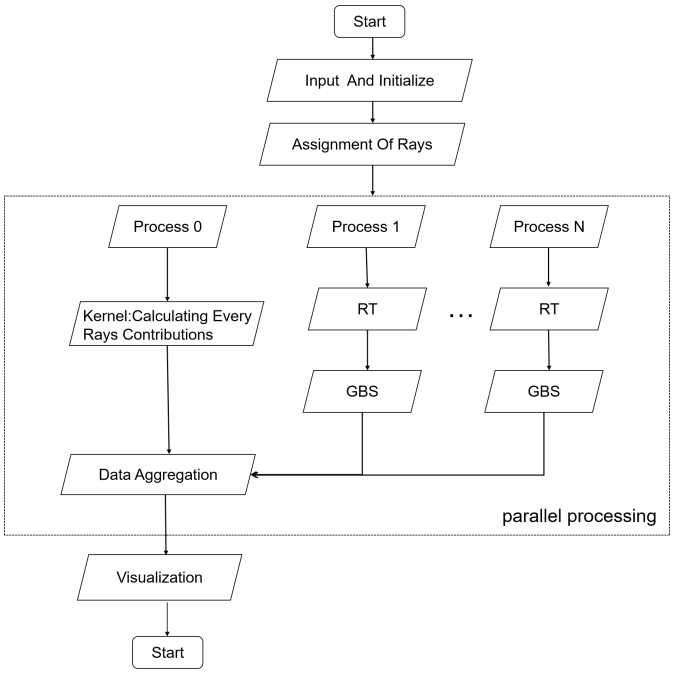


Figure 1. General architecture of heterogeneous computing for acoustic rendering programs.

2.3 CPU Parallel Computing Strategy

After the master scheduling thread allocates ray tracing tasks to the GPU according to the established strategy, the remaining CPU threads take on the tasks allocated to the CPU. These tasks are evenly distributed among all available CPU cores, ensuring balanced computational load distribution across CPU cores and maximizing CPU resource utilization.

In each ray's loop, the RT module first calculates the sound wave's propagation path, including its trajectory and reflections off obstacles. The DRT method then simulates sound wave behavior, accounting for variations in sound speed and wave properties across different media. This process focuses not just on the path but also on the sound waves' physical behaviors during propagation.In the core computations of the GBS (Gaussian Beam Summation) stage, the data from the RT module is first processed. This includes setting up observation points, initializing frequency parameters, and

multi-layer traversal processing for each ray segment and observation point. Key steps include calculating the propagation parameters of Gaussian beams using dynamic ray tracing data, processing the directivity function of the sound field, considering atmospheric absorption effects, and calculating and accumulating the contributions of each ray segment.

2.4 GPU Parallel Computing Strategy

Before computation, the master thread loads data into the GPU memory, using a block processing strategy to optimize memory use and efficiency. Ray tracing tasks are divided into blocks fitting the GPU memory capacity, with each block processing a set number of rays in a loop, ensuring computational loads match the GPU's memory.

On the basis of block processing, the master scheduling thread subsequently launches CUDA kernel functions to officially start the GPU's parallel computing process. Each CUDA thread within the kernel function is assigned an independent ray task and is responsible for executing the tracing process for that ray. During the ray tracing phase, CUDA threads calculate the propagation paths of each ray in complex environments, including interactions with various surfaces (such as reflection, refraction, and absorption).Particularly noteworthy is the handling of the GBS (Gaussian Beam Summation) computation stage. The GPU parallel programme framework diagram is shown in Figure 2.

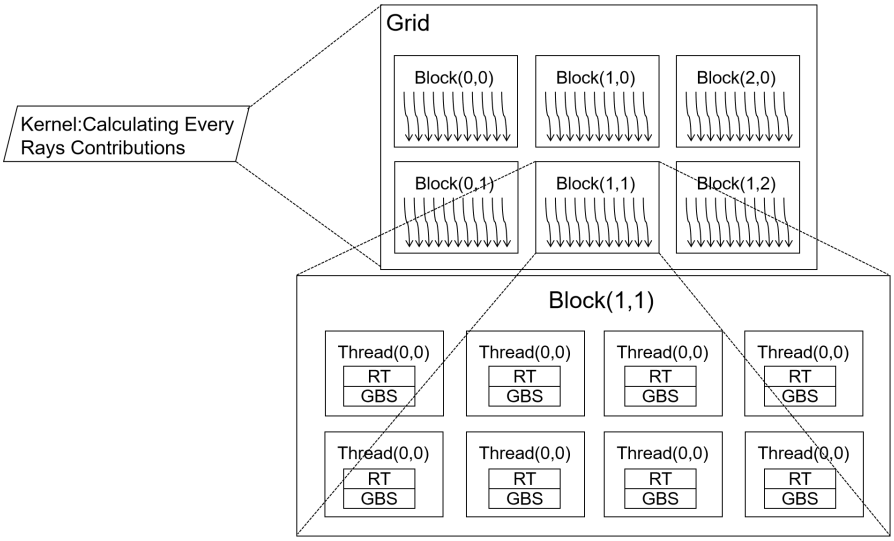


Figure 2. GPU program general architecture diagram.

After completing the GBS calculations, the next step is data aggregation, at which point the GPU has completed its computational tasks and has saved the results in device memory. Meanwhile, the master scheduling thread waits for the other CPU cores to complete their computational tasks. This synchronization ensures that all parts of the computation, both on the CPU and GPU, are complete before moving on to the final stages of the program.

2.5 Data Integration and Visualization File Output

After completing the GBS computation, CPU cores finish their tasks, and their results are collected by the master thread for data fusion. The master thread integrates these results with GPU outputs to construct a detailed sound field simulation, outputting files in VTK format. This format illustrates sound pressure levels across frequencies at observation points, offering a detailed view of sound field characteristics, such as pressure levels and distribution maps, serving as a basis for further visualization.

3. VERIFICATION OF ACCELERATION EFFECTS

To verify the acceleration effects of the Gaussian Beam Tracing (GBT) algorithm under the GPU and CPU parallel computing framework, we conducted tests using specific experimental setups and case parameters. Experiments were

conducted on a platform with an Nvidia RTX A6000 GPU (10752 CUDA cores, 48 GB memory) and an AMD 9754 CPU (128 cores, 256 threads). A cubic box model with a 10-meter edge was used, with rays uniformly radiated from the center to simulate sound propagation. Observation points within a cross-section captured pressure level changes and sound behavior.The inputs and outputs of the programme are shown in Figure 3.

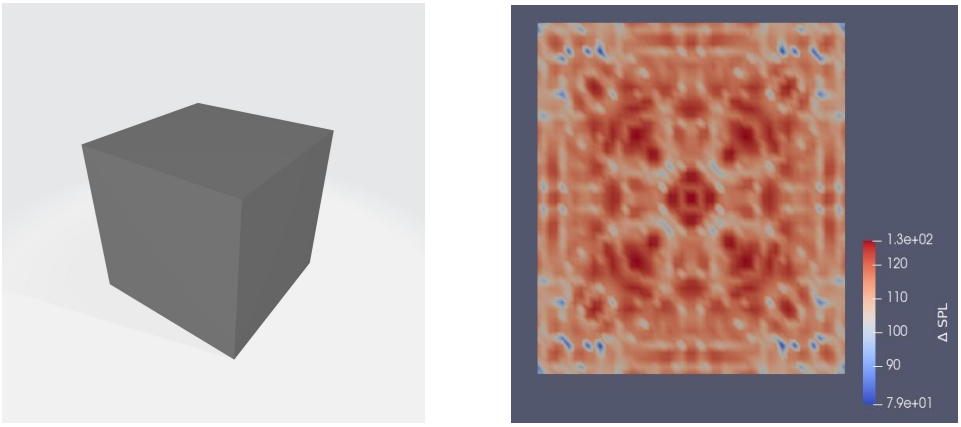


Figure 3. Inputs and results.

3.1 Ray Task Allocation Strategy

In a heterogeneous computing environment, accurately allocating ray tasks between the GPU and CPU is crucial for enhancing computational efficiency and ensuring the accuracy of results. Given the uncertainty of computing unit performance characteristics, this study conducted detailed performance evaluations of both the GPU and CPU, aiming to discover the optimal ray task allocation strategy. The strategy was supported by experimental data, with specific results displayed in Figure 4.

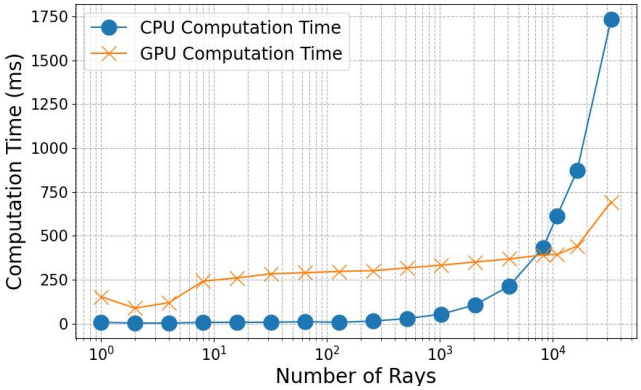


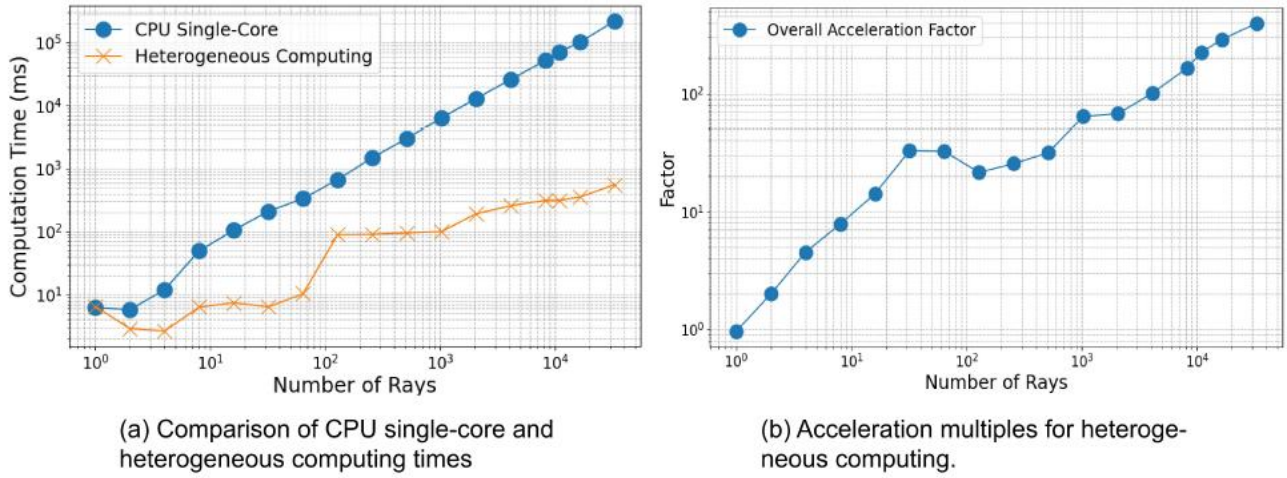
Figure 4. Comparison of CPU and GPU computing time

Accurate task allocation between the GPU and CPU is crucial for efficiency. Performance evaluations showed that for small-scale tasks (≤rays), the CPU performs better. For tasks with 64-1024 rays, a 70% CPU and 30% GPU allocation is balanced. For larger tasks (>1024rays), the GPU shows clear advantages, recommending a 30% CPU and 70% GPU allocation. For very large tasks (>16384rays), an 80% GPU allocation maximizes efficiency.

3.2 Analysis of Experimental Results

The performance evaluation results indicate that for smaller-scale ray tasks (no more than 64 rays), the CPU demonstrates higher computational efficiency compared to the GPU. Based on this, the study suggests prioritizing CPU for such tasks. When the volume of ray tasks ranges between 64 and 1024, even though the computation time on the

GPU shows a decreasing trend, the CPU's performance does not exhibit a significant advantage in comparison. Therefore, a task allocation ratio of 70% (CPU) to 30% (GPU) is recommended to achieve a balanced computational load and fully utilize the performance advantages of each computing unit. For larger-scale ray tasks (over 1024), the GPU begins to show a clear advantage in parallel processing capability, especially when the number of rays increases to 4096 and above, where the GPU demonstrates significant acceleration performance. In light of this, adjusting the task allocation ratio to 30% (CPU) and 70% (GPU) is appropriate to maximize the parallel computing potential of the GPU. For extremely large-scale ray tasks (16384 or more), further increasing the GPU task ratio to 20% (CPU) and 80% (GPU) is more suitable to effectively utilize the GPU's significant advantages in handling massive parallel tasks, aiming to reduce the overall computation time.The accelerated result shown in Figure 5.



(a) Comparison of CPU single-core and heterogeneous computing times

(b) Acceleration multiples for heterogeneous computing.

Figure 5. Accelerated result

Experimental data shows that when processing a smaller number of rays (ranging from 1 to 64), although single-core CPU already demonstrates relatively high efficiency, the heterogeneous computing environment can still achieve further performance improvement, with the acceleration factor varying between 0.97 to 4.52. This finding indicates that even in smaller-scale ray tasks, a reasonable allocation of tasks between the GPU and CPU can bring about performance optimization, although the parallel computing advantage of the GPU is not fully realized at this stage.As the number of rays gradually increases, the acceleration effect brought by the heterogeneous computing environment becomes more pronounced. Especially when the number of rays exceeds 64, the parallel processing advantage of the GPU begins to significantly surpass that of the single-core CPU, with a substantial increase in the acceleration factor. Notably, when the number of rays reaches 4096 and above, the acceleration factor of the heterogeneous computing environment compared to single-core CPU computation exceeds 100 times, fully demonstrating the excellent performance of heterogeneous computing in handling large-scale parallel tasks.

Most strikingly, when the number of rays expands to 32,768, the acceleration factor of the heterogeneous computing environment relative to single-core CPU approaches nearly 400 times. This result not only verifies the superiority of the GPU in processing massive parallel tasks but also highlights the key role of the heterogeneous computing strategy in maximizing the potential of computing resources. By properly allocating computational tasks between the GPU and CPU, even when facing tens of thousands of ray tasks, the proposed heterogeneous computing strategy can significantly reduce the overall computation time required, thereby greatly enhancing the overall computational efficiency of sound field simulation.

Through the experimental verification of the above case studies, we have successfully demonstrated the excellent acceleration performance of the GBT algorithm in sound field simulation tasks based on the GPU and CPU parallel

computing framework. This achievement not only provides reliable computational support for future more complex acoustic simulations but also opens new possibilities for the application of parallel computing technology in the field of acoustics.

4. CONCLUSION

This study investigated how sound field rendering algorithms, particularly the Gaussian Beam Tracing (GBT) algorithm, can be optimized in a heterogeneous computing environment. By conducting comprehensive evaluations using the NVIDIA RTX A6000 GPU and AMD 9754 CPU, we demonstrated that distributing ray tasks between the GPU and CPU greatly enhances computational efficiency. This is especially evident in large-scale ray tasks, where the advantages of heterogeneous computing significantly outperform traditional single-core CPU processing.

The core achievement of this research is the proposal of a heterogeneous computing framework that combines CPU and GPU resources to accelerate the sound field rendering process. Based on a deep analysis of the performance characteristics of these two computing resources, an efficient ray task allocation mechanism was proposed to achieve optimal resource configuration and maximize computational efficiency in the heterogeneous computing environment. Experimental validation showed that this framework and strategy could achieve nearly 400 times acceleration for tens of thousands of rays, substantially reducing the overall computation time for sound field simulation.

This study improved sound field rendering algorithm efficiency by allocating ray tasks across CPU and GPU and optimizing the framework. It highlights the importance of result convergence for efficiency and accuracy. Increasing rays doesn't always improve accuracy, suggesting future work should include a convergence evaluation mechanism. This mechanism would stop increasing rays when differences between results fall below a threshold, preventing resource waste. By timely evaluating the convergence of simulation results, not only can ineffective computational resource waste be avoided, but also a more efficient and accurate strategy for sound field simulation can be provided.

REFERENCES

[1] Pretzsch, A., Seidler, A., & Hegewald, J. (2021). Health Effects of Occupational Noise. Current Pollution Reports, 7, 344 - 358. https://doi.org/10.1007/s40726-021-00194-4.

[2] Jiang, B., Yu, J., Li, W., Chai, Y., & Gui, Q. (2023). A Coupled Overlapping Finite Element Method for Analyzing Underwater Acoustic Scattering Problems. Journal of Marine Science and Engineering. https://doi.org/10.3390/jmse11091676.

[3] Bian, H., Fattah, R., Zhong, S., & Zhang, X. (2020). An efficient rectilinear Gaussian beam tracing method for sound propagation modelling in a non-turbulent medium.. The Journal of the Acoustical Society of America, 148 6, 4037 . https://doi.org/10.1121/10.0002966.

[4] Bian, H., Tan, Q., Zhong, S., & Zhang, X. (2022). Efficient computation of broadband noise propagation using Gaussian beam tracing method.. The Journal of the Acoustical Society of America, 151 5, 3387 . https://doi.org/10.1121/10.0011399.

[5] Zhang, P., Holk, E., Matty, J., Misurda, S., Zalewski, M., Chu, J., McMillan, S., & Lumsdaine, A. (2015). Dynamic parallelism for simple and efficient GPU graph algorithms. Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms. https://doi.org/10.1145/2833179.2833189.

[6] Madariaga R. Gaussian beam synthetic seismograms in a vertically varying medium[J]. Geophysical Journal International, 1984, 79(2): 589-612.