A High-Performance Astronomical Data Distribution Method Based on MPI + CUDA (Postprint)

Authors: Fang Bing (1); Deng Hui (1); Zhang Xiaoli (1); Mei Ying (2); Shi Congming (1); Chen Xiaolin (1); Dai Wei (1,2); Wu Jingping (1); Wang Feng (1,2)

Date: 2017-09-26T00:00:00+00:00

Abstract

The emergence of massive astronomical data has brought numerous challenges to the development of astronomical software. In recent years, with the advancement of parallel computing technology, MPI + GPU has gradually become the primary paradigm for high-performance astronomical data processing. Focusing on the problem of improving reconstruction performance in solar high-resolution image reconstruction, this work conducts a systematic investigation into the data reading and distribution methods therein. In traditional MPI parallel processing, the master process partitions the original image into sub-blocks, which are then transmitted to individual sub-processes for reconstruction, with the reconstructed results subsequently returned to the master process. When the number of sub-processes is large while the number of compute nodes is limited, this data distribution process significantly increases communication time, thereby affecting the efficiency of the entire reconstruction workflow. This paper proposes a tree-structured data distribution method for MPI + CUDA, presenting the fundamental concepts and implementation approaches of the algorithm. Experimental results demonstrate that the tree-based distribution method can achieve nearly twofold speedup compared to the commonly employed parallel distribution approach, offering valuable insights for the development and processing of massive astronomical datasets.

Full Text

A High-Performance Astronomical Data Distribution Method Based on MPI + CUDA

Authors: Bing Fang[1], Hui Deng[1], Xiaoli Zhang[1], Ying Mei[1], Congming Shi[1], Xiaolin Chen[1], Wei Dai[1], Jingping Wu[2], Feng Wang[2]
Affiliations:
[1] Yunnan Key Laboratory of Computer Technology Application, Kunming University of Science and Technology, Kunming, Yunnan
[2] Yunnan Observatories, Chinese Academy of Sciences, Kunming, Yunnan

Abstract

The emergence of massive astronomical datasets has introduced numerous challenges for astronomy software development. In recent years, with the advancement of parallel computing technology, the MPI + GPU model has gradually become the dominant paradigm for high-performance astronomical data processing. This paper systematically investigates data reading and distribution methods to address the problem of improving reconstruction performance in solar high-resolution image reconstruction. In traditional MPI parallel processing, the master process divides the original image into sub-blocks and sends them to each sub-process for reconstruction; the results are then returned to the master process. When the number of sub-processes is large but computational nodes are few, this data distribution process significantly increases communication time and affects overall reconstruction efficiency. We propose a tree-structured data distribution method under MPI + CUDA and present the fundamental concepts and implementation approach of the algorithm. Experimental results demonstrate that the tree distribution mode can achieve nearly double the speed of the commonly used flat distribution mode, offering valuable insights for the development and processing of massive astronomical datasets.

Keywords: Image reconstruction; MPI + GPU; Data distribution

1. Introduction

The advent of massive astronomical data has greatly propelled the development of high-performance computing technologies, emerging as a highlight in current HPC research. As a new generation of high-density distributed computing architecture, Graphics Processing Units (GPUs) have become the most widely used general-purpose computing platform due to their excellent programmable architecture and complete service framework. NVIDIA' s Compute Unified Device Architecture (CUDA), based on the General-Purpose GPU (GPGPU) model [1], provides an ideal computing environment. The Message Passing Interface (MPI) is the most commonly used high-performance computing support environment [2]. Combining the advantages of both technologies, the hybrid MPI + CUDA programming model has become a mainstream paradigm for massive

scientific data processing. In this hybrid model, MPI handles task parallel distribution while CUDA manages parallel computation, making ultra-large-scale data processing feasible and significantly improving parallel efficiency.

A critical challenge in astronomical observation is how to rapidly process the massive image data generated daily. Rather than simply increasing storage capacity for later processing, astronomers increasingly require real-time processing of observational images to enhance overall observation efficiency. With the rapid development of GPUs and the maturation of MPI + GPU applications, researchers have begun employing this high-performance parallel model to solve astronomical big data problems. This paper examines data distribution issues encountered in developing high-performance astronomy software, focusing on distribution methods in the MPI + GPU model for astronomical massive data processing.

2. Overview of the MPI + CUDA Hybrid Programming Model

The MPI + CUDA hybrid programming model effectively combines MPI' s coarse-grained parallelism with CUDA's fine-grained parallel computing capabilities, representing the current mainstream parallel computing model. As shown in [Figure 1: see original paper], when using this hybrid model for large-scale matrix multiplication testing, significant acceleration effects can be achieved. However, to further improve system performance, one can either expand the hardware environment to build larger parallel clusters or optimize MPI + CUDA communication patterns to reduce communication overhead and enhance operational speed.

According to Amdahl' s Law [5], where a represents the proportion of parallel computation and n represents the number of parallel processing nodes, the maximum speedup achievable by a parallel system is limited to 1/(1-a + a/n). This demonstrates that acceleration is constrained not only by the serial component of the program but also by additional runtime overhead. Unlimited increases in processor numbers do not necessarily yield better results due to factors like communication time. Therefore, research on optimizing communication methods for the MPI + CUDA model is crucial for further improving its efficiency.

In inter-process communication, data transfer occurs directly between CPU memory and GPU memory rather than between CPUs across nodes, thanks to NVIDIA' s CUDA-aware MPI technology[1]. This enables direct access to GPU buffers, avoiding time-consuming CPU-GPU data copies, thereby further improving MPI + CUDA parallel efficiency and reducing programming complexity. As a result, the MPI + GPU model has gained wider recognition and application.

3. MPI-Related Work in Astronomical Data Processing

With the widespread adoption of MPI and rapid GPU development, significant progress has been made in applying these technologies to astronomical data pro-

cessing. Researchers have implemented a real-time full-disk cloud contamination detection and repair system [6], utilized GPUs for Mark5B data processing to meet real-time pulsar observation requirements [7], and achieved GPU-based phase diversity (PD) algorithms for ground-based telescope image restoration with dozens of times acceleration [8]. Studies on solar high-resolution image reconstruction algorithms on GPUs [9] and MPI-accelerated frame selection and stacking for solar high-resolution imaging (Level 1) [10] have provided new parallel implementation methods for astronomical image processing. These works demonstrate the potential of MPI + CUDA hybrid programming for astronomical image reconstruction, though they primarily focus on parallel methods rather than data distribution, which has not been thoroughly investigated. Some studies have noted that data distribution can impact system efficiency more than computation itself.

4. Data Distribution Methods in MPI + GPU-Based High-Resolution Image Reconstruction

During high-resolution image reconstruction, the original image must be divided into sub-images for algorithmic processing and then stitched back together. The conventional approach uses the master process to cut the original image on the CPU and send sub-blocks to each sub-process' s GPU for reconstruction. When multiple processes simultaneously request data from the same source, waiting occurs, reducing efficiency.

To improve reconstruction efficiency, we propose a tree-structured data distribution model suitable for environments with few computational nodes. The fundamental idea is to distribute original data into each GPU' s memory, then perform further cutting and sending to next-level sub-processes via GPUs. Image processing sub-processes fetch data from multiple upper-level processes, increasing parallelism in data distribution and reducing communication waiting time. Since both distribution and processing occur on GPUs, processing speed is further enhanced.

4.1 Flat Distribution Method In the flat distribution method, the master process cyclically cuts the image according to the number of processes and sends sub-blocks to each sub-process for GPU computation. Using CUDA-aware MPI, data is sent directly from CPU memory to GPU memory. As shown in [Figure 2: see original paper], the master process (rank 0) cuts sub-images based on sub-process ranks and sends them to corresponding GPUs. After computation, results are returned to the master process. The process continues until the entire image is processed.

4.2 Tree Distribution Method The tree distribution method, as the name suggests, distributes data hierarchically like a tree. The original data is divided into several blocks in the GPU memory of the master process (rank 0), then cut into sub-blocks on the GPU and sent to other sub-processes. As illustrated in

[Figure 3: see original paper], the master process first divides the original image into *devCount* portions sent to GPUs 1~2. These second-level sub-processes then cut the data on their GPUs and send sub-blocks to third-level sub-processes. After processing, third-level sub-processes return results to their parent second-level processes, which integrate the results and send them back to the master process for final assembly.

5. Experimental Design and Results Analysis

5.1 Experimental Environment The experimental environment comprised two servers with dual-core CPUs, running Ubuntu 14.04.4 LTS, OpenMPI 2.0.0, and CUDA 7.5. Each server was equipped with two dual-core GPUs, allowing access to four GPU devices total. The experiments processed large matrices to simulate the cutting and distribution of high-resolution images, with time-consuming matrix operations representing the image algorithm processing steps.

5.2 Experimental Results Experiments processed matrices of sizes 10,240 × 10,240 and 20,480 × 20,480, with sub-block sizes of 1,024 × 1,024 and 2,048 × 2,048. Sub-processes performed time-consuming operations on GPU sub-blocks and returned results to parent processes upon completion.

Single-server results: For the 10,240 × 10,240 matrix ([Figure 4: see original paper]), processing time increased with process count and communication overhead. The tree method showed better performance than flat distribution, though both exhibited increasing trends. For the 20,480 × 20,480 matrix ([Figure 5: see original paper]), the flat method showed a concave curve—efficiency improved with more processes up to a point, after which communication overhead dominated and efficiency decreased. The tree method maintained approximately double the speed of the flat method.

Cluster results: Processing the 20,480 × 20,480 matrix across two servers ([Figure 6: see original paper]) demonstrated the tree method' s superiority in cluster environments. The tree method curve also showed a concave shape when process counts were low, indicating underutilization of GPU resources. When process counts exceeded GPU availability, blocking occurred. Optimal performance requires process counts greater than GPU counts (preferably more than double) to fully utilize resources. In tree distribution, second-level process counts should match GPU counts to control data distribution and result aggregation. Excessive sub-process numbers should be avoided, as over-dividing images may increase reconstruction errors.

6. Conclusion

This paper focused on data distribution mechanisms in parallel computing. Experimental comparisons demonstrate that the tree distribution method offers significant advantages over flat distribution, providing a valuable reference for data distribution in high-resolution image reconstruction. Beyond the widely

studied MPI + OpenMP hybrid model, the MPI + OpenMP + CUDA model, which leverages shared-memory and message-passing paradigms, is also gaining attention and will be the subject of future research in astronomical massive data processing.

References

[1] Luebke D. CUDA: Scalable parallel programming for high-performance scientific computing. IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2008.
[2] Gropp W, Lusk E, Doss N, et al. A high-performance, portable implementation of the MPI message-passing interface standard. Parallel Computing, 1996, 22(6): 789-828.
[3] Xu Yanqin, Chen Qingkui. Research and implementation of MPI + CUDA model based on SMP clusters. Computer Engineering and Design, 2013, 34(10): 3408-3412.
[4] Liu Qingkun, Ma Mingwei, Yan Weichun. Parallel matrix multiplication based on MPI + CUDA asynchronous model. Journal of Computer Applications, 2012, 32(11): 3327-3330.
[5] Amdahl G M. Validity of the single processor approach to achieving large scale computing capabilities. AFIPS Spring Joint Computer Conference, 1967.
[6] Zhang Nengwei, Yang Yunfei, Li Ranyang, et al. A real-time image processing system for detecting and removing cloud shadows on H full-disk solar images. Astronomical Research & Technology, 2015, 12(2): 242-249.
[7] Li Jiagong, Xu Yonghua, Li Zhixuan, et al. An observation system for pulsars based on a GPU architecture and Mark5B. Astronomical Research & Technology, 2015, 12(3): 335-342.
[8] Shi Zheng, Xiang Yongyuan, Deng Hui, et al. High-speed implementation of a phase-diversity-based image restoration algorithm for ground-based telescopes on GPU. Science Bulletin, 2015, 60(15): 1408-1413.
[9] Xiang Yongyuan. Research on high-speed reconstruction algorithms for solar high-resolution images. Kunming: Yunnan Observatories, Chinese Academy of Sciences, 2015.
[10] Li Xuebao. Research on parallel processing technology for massive data from solar telescopes. Kunming: Yunnan Observatories, Chinese Academy of Sciences, 2015.

---

[1] https://devblogs.nvidia.com/parallelforall/introduction-cuda-aware-mpi/

Note: Figure translations are in progress. See original paper for figures.