Postprint of MPI-CUDA-Based K-T Algorithm for Solar High-Resolution Image Reconstruction

Authors: Yang Qiuping, Song Zhenqi, Deng Hui, Wang Feng

Date: 2018-05-15T00:00:00+00:00

Abstract

Atmospheric turbulence leads to image blurring in ground-based telescopes, and high-resolution image reconstruction represents an effective solution to this problem. The K-T high-resolution image reconstruction method is among the commonly employed approaches; however, it is constrained by factors such as large data volumes and computational complexity, resulting in extremely time-consuming reconstruction processes. This work investigates high-resolution image reconstruction on contemporary hybrid CPU-GPU architectures, utilizing MPI-CUDA hybrid parallel technology to ultimately implement a high-resolution solar image reconstruction system based on speckle image reconstruction techniques in a single-machine GPU environment. Experimental validation demonstrates that, compared with parallelization using Message Passing Interface alone, the processing speed for image sub-blocks has achieved substantial improvement, with the overall pipeline attaining a speedup ratio of 2 under 8 sub-processes. The experimental results substantiate the effectiveness of MPI-CUDA hybrid parallelism and can serve as a reference for large-scale computational tasks in astronomical research.

Full Text

Preamble

High-Resolution Solar Image Reconstruction with K-T Algorithm Based on MPI-CUDA

Yang Qiuping, Song Zhenqi, Deng Hui, Wang Feng
(Computer Technology Application Key Laboratory of Yunnan Province, Kunming University of Science and Technology, Kunming 650500, China)

Abstract

Atmospheric turbulence causes blurring in ground-based telescope imaging, and high-resolution image reconstruction is an effective solution to this problem. The Knox-Thompson (K-T) algorithm is a commonly used high-resolution reconstruction method, but it is severely constrained by large data volumes and computational complexity, making the reconstruction process extremely time-consuming. This paper investigates high-resolution image reconstruction under CPU-GPU hybrid architectures, employing MPI-CUDA hybrid parallel technology to implement a high-resolution solar image reconstruction system based on speckle imaging techniques in a single-machine GPU environment. Experimental validation demonstrates that compared with MPI-only parallelization, the processing speed of image sub-blocks has improved significantly, achieving an overall process speedup ratio of 2 with 8 subprocesses. The results confirm the effectiveness of MPI-CUDA hybrid parallelism and provide valuable insights for large-scale computational tasks in astronomical research.

Keywords: High-resolution reconstruction; MPI-CUDA; Parallel computing

Introduction

Ground-based telescope imaging is affected by atmospheric turbulence, which causes image scintillation and jitter, thereby limiting image resolution [1]. High-resolution solar observation images are essential for solar physics research, requiring telescopes to achieve diffraction-limited imaging [2]. Current post-processing methods mainly include blind deconvolution, phase diversity, and speckle statistical reconstruction. Speckle statistical reconstruction fundamentally relies on statistical calculations from a set of short-exposure images containing high-frequency information to reconstruct target high-frequency details [3], with the Knox-Thompson (K-T) algorithm being a representative approach. However, due to large observation datasets and computational complexity, the post-processing reconstruction is severely time-consuming and cannot meet real-time observation requirements. Advanced telescopes domestically and internationally typically employ high-performance computers or large-scale clusters for parallel high-resolution reconstruction—for instance, the Big Bear Solar Observatory uses an 80-core computing cluster [4]. Yet large-scale clusters are prohibitively expensive, while smaller clusters cannot satisfy real-time processing demands, creating an urgent need for more efficient computational methods.

In recent years, general-purpose GPU computing technology has developed rapidly and found widespread application in complex computational tasks. This paper studies high-resolution image reconstruction under CPU-GPU hybrid architectures, systematically investigating the implementation of K-T algorithm-based reconstruction using MPI-CUDA hybrid parallel technology, ultimately achieving a high-resolution solar image reconstruction system in a single-machine environment.

Funding: National Natural Science Foundation of China (U1531132, U1631129,

11403009, 11463003, U1231205); Yunnan Applied Basic Research Project
Received: 2017-09-15; Revised: 2018-01-02
Author: Yang Qiuping, Lecturer. Research interests: Distributed computing. Email: yangqiuping@astrolab.com
Corresponding author: Deng Hui, Professor. Research interests: Astronomical technology and methods. Email: dh@cnlab.net

1.1 CUDA Overview

Graphics Processing Units (GPUs) were originally developed to enhance real-time and efficient computer graphics rendering. They have evolved into multi-core processors with high density, massive parallelism, and multi-threading capabilities, possessing powerful computational capacity and extremely high memory bandwidth [5]. For example, the NVIDIA Tesla K80 achieves single-precision performance of 7 TFLOPS, far exceeding CPU capabilities. CUDA is a general-purpose parallel computing architecture that leverages GPU processing power to substantially improve computational performance, enabling developers to write programs directly in C/C++. CUDA parallel computing functions running on GPUs are called kernels, and a complete CUDA program consists of a series of kernel functions combined with serial processing sections on the host side. A kernel is organized as a grid of threads, where each grid comprises multiple thread blocks, and each block contains multiple threads. Kernels execute at the block level, with only threads within the same block able to communicate with each other. CUDA provides numerous application programming interfaces for complex computations, such as the CUBLAS and CUFFT libraries used in this work. CUBLAS offers extensive basic matrix operations, while CUFFT provides one- to three-dimensional Fourier transform functions for various data types.

1.2 MPI-CUDA Hybrid Programming

MPI is a message-passing protocol that enables inter-process communication to achieve parallel computing. MPI has been widely applied in complex computational tasks; for example, the Mingantu Radio Spectral Heliograph uses MPI for high-performance UVFITS data synthesis [6]. However, MPI parallelism only distributes computational tasks into multiple subtasks executed simultaneously to reduce overall computation time. If subtasks are data-intensive, CPU computational capacity may still fail to meet real-time requirements. The MPI-CUDA hybrid model uses MPI for task partitioning and CUDA for accelerating subtask computation, combining the strengths of both architectures to improve scientific computing efficiency.

2 Speckle Statistical Reconstruction Methods

Speckle statistical reconstruction is a method based on statistical information from short-exposure images, obtaining high-resolution reconstructions through statistical processing of target speckle patterns [7]. Algorithms are primarily cat-

egorized into frequency-domain and spatial-domain reconstruction. Frequency-domain methods include Labeyrie' s method based on second-order statistics [8], the Knox-Thompson (K-T) method [9], and the speckle masking method based on third-order statistics [10]. Spatial-domain methods include simple shift-and-add, iterative shift-and-add, and correlation-based shift-and-add with frame selection. This paper employs the K-T method for high-resolution solar image reconstruction.

2.1 Labeyrie' s Method

Under the isoplanatic assumption, each short-exposure image can be regarded as the target convolved with the optical point spread function. Its Fourier transform can be expressed as =O() · H() (2) , where O() and H() represent the Fourier transforms of the target and point spread function, respectively; is the two-dimensional spatial frequency variable. Labeyrie' s method, also known as speckle interferometry, recovers the target' s Fourier amplitude through ensemble averaging of speckle pattern energy spectra. Following Labeyrie' s statistical approach, the ensemble average of squared frequency-domain amplitudes from multiple short-exposure images yields , where brackets denote arithmetic averaging; is called the speckle interferometry transfer function (STF). Deconvolving the STF yields the target' s Fourier amplitude. Speckle interferometry loses phase information during statistical processing, enabling only amplitude reconstruction, while complete target reconstruction requires both amplitude and phase.

2.2 K-T Method

To reconstruct target phase, Knox and Thompson proposed an improved method in 1974 based on computing cross-spectra to obtain phase information, known as the Knox-Thompson (K-T) method or cross-spectrum technique. This method introduces a frequency shift when computing second-order moments, yielding a cross-spectrum expressed as , where * denotes complex conjugation and represents the spatial frequency shift. When =0, the cross-spectrum equals the energy spectrum. The shift magnitude should be smaller than , where is the atmospheric coherence length, is the observation wavelength, and f is the system focal length. The statistical result of speckle pattern cross-spectra is , where is the K-T transfer function—a significant real number near the diffraction-limited cutoff frequency—so the cross-spectrum phase represents phase differences between adjacent frequencies. Phase values at other frequencies are obtained through iteration starting from zero frequency.

3 MPI-CUDA Implementation of K-T Image Reconstruction

The entire system is implemented in a single-machine environment, with the computational flow shown in Figure 1 [Figure 1: see original paper], comprising

three main components: an MPI master process, multiple MPI worker processes, and GPU programs. The master process handles data distribution and collection, reconstructing and stitching sub-images from amplitude and phase data. Each worker process computes amplitude and phase for several sub-blocks, with computations primarily performed on the GPU.

3.1 Image Preprocessing and Blocking

Before computation, image data must be preprocessed through flat-field and dark-field corrections. In imaging systems, linear space invariance is difficult to maintain across the entire field of view, but it approximately holds within an isoplanatic patch region where identical point spread functions can be used. Therefore, a set of short-exposure images can be divided into multiple sub-blocks based on isoplanatic patch size, with each sub-block independently computing reconstructed amplitude and phase. During final stitching, only central portions of sub-blocks are used, so sub-blocks are overlapped during division—this work employs 50% overlap.

3.2 Data Distribution

Since computations between sub-blocks are independent, each sub-block' s reconstruction can be treated as a separate task. The MPI parallel computing framework parallelizes sub-block reconstruction by evenly distributing all sub-blocks among worker processes. To reduce inter-process communication overhead, particularly data distribution time from master to workers, raw image data is stored in shared memory. The master process only transmits control information, such as sub-block data index positions in the image array, enabling each worker to directly read image data from shared memory based on indices, thereby saving substantial communication time.

Although each process handles only sub-block computation, these remain data-intensive complex calculations. To further optimize performance, the CPU-GPU heterogeneous computing framework enables parallel data processing on GPUs. Before computation, data must be copied to GPU memory. Since host-device data transfer occurs via PCI bus and each worker processes multiple sub-blocks in loops, frequent data transfers severely impact performance. To minimize this impact, all sub-block data is copied to GPU memory before computation begins.

3.3 Amplitude Reconstruction

Amplitude reconstruction is typically performed in the frequency domain using Labeyrie's method to deconvolve the transfer function. The computation process for each sub-block is as follows: (1) Compute the fast Fourier transform of the speckle sub-block; (2) Calculate the sub-block's spectral ratio; (3) Determine the coherence length through matching; (4) Compute the speckle transfer function from ; (5) Obtain the target' s Fourier amplitude through deconvolution of .

First, sub-block images are transformed from spatial to frequency domain using CUDA' s CUFFT library. Step (2) is the most computationally intensive part, requiring arithmetic averaging of single-frame images within sub-blocks by summing values at all frequency points in the frequency domain. However, summation is data-dependent and not inherently parallel-friendly. Leveraging the characteristic that threads within a block can communicate via shared memory and synchronize, a reduction sum approach reduces computational complexity from to . Other computations at each frequency point are independent, so the total number of threads is set equal to the number of frequency points to maximize GPU parallel capability. Under Kolmogorov turbulence spectrum assumptions, the STF is solely a function of atmospheric coherence length. Theoretically computed values under various conditions serve as references for matching with step (2) results to obtain , from which the STF is derived and deconvolved to yield the target Fourier amplitude.

3.4 Phase Reconstruction

Complete image reconstruction requires both amplitude and phase. The K-T method computes cross-spectrum phase, representing phase differences between adjacent frequencies, with phase values at other frequencies obtained through iteration from zero frequency. The computation process for each sub-block is: (1) Compute FFT of speckle sub-block; (2) Compute cross-spectra for K different frequency shifts; (3) Iteratively compute sub-block phase under K different shifts; (4) Average the K phase results to obtain final phase.

To improve computational accuracy, K different frequency shifts are applied and their cross-spectra computed, significantly increasing computational load. Serial computation for each frequency point would require K×N loops, representing enormous computational demand. Analysis reveals that cross-spectrum computations for different shifts are independent, so the number of threads is set equal to the total frequency points, with each thread computing K different shifts for one frequency point. Since cross-spectra only provide phase differences between adjacent frequencies, target phase must be obtained through iterative summation from zero frequency. Phase at frequency (m,n) can be derived via different paths—for example, summing phase at (m-1,n) with its corresponding phase difference, or phase at (m,n-1) with its difference. For improved accuracy, results from different paths are typically averaged. Such data-dependent computations, where each point' s phase depends on previous calculations, currently lack effective GPU parallelization methods.

2.4 Image Stitching

From the obtained target Fourier amplitude and phase of each sub-block, reconstructed sub-images are generated through inverse Fourier transform. After reconstructing all sub-images, they must be stitched into a full field-of-view image. Since frequency-domain phase reconstruction uses the initial phase of

superimposed images, stitching generally introduces no offsets. Post-stitching processing includes image alignment and other corrections.

4 Experimental Analysis

Performance of the CUDA-MPI hybrid model for solar image high-resolution reconstruction was tested on a system configured with an Intel Xeon-2620v3 processor at 2.4 GHz, two NVIDIA Tesla K80 GPUs (each with two cores, 4,992 CUDA cores, and 12 GB memory), running Ubuntu 14.04.5 LTS. Test data comprised 100 frames of short-exposure solar images observed by the 1m solar telescope, with parameters shown in Table 1 .

Experiments tested both MPI-CUDA and pure MPI parallel implementations of the K-T algorithm using different process counts. With four GPU compute cores, subprocess counts were set as multiples of 4 for balanced workload distribution, with results shown in Table 2 . Comparisons reveal that with a single subprocess (serial computation), MPI-CUDA achieves significant speedup, reducing total computation time to one-third of pure MPI, demonstrating clear GPU acceleration benefits. As subprocess count increases, parallel advantages become more pronounced, with both approaches showing substantial time reductions, though MPI-CUDA consistently outperforms pure MPI. However, limited by GPU and CPU capabilities, computation time reaches a minimum before increasing with further process count escalation. MPI-CUDA achieves minimum computation time of only 12 seconds with 8 subprocesses, while pure MPI requires 20 seconds at its minimum, demonstrating clear superiority of the hybrid approach.

Furthermore, with 8 subprocesses, CPU and GPU processing times for each stage were compared, as shown in Table 3 . The comparison indicates that every algorithm component achieves speedup after GPU porting, though speedup ratios vary depending on implementation details and parallelization degree. Notably, averaging computation achieves 172.56× acceleration. In terms of time consumption, cross-spectrum computation dominates due to K different frequency shifts, but GPU acceleration yields 5.76× speedup, reducing its proportion in single sub-block computation from 75.3% to 31.1%, playing a critical role in overall acceleration. Since recursive phase computation with strong data dependencies cannot be GPU-parallelized, and the CPU must handle image stitching and logical control flow, plus host-device data transfer overhead, the MPI-CUDA approach ultimately halves the time compared with pure MPI parallelism. Due to architectural differences between GPUs and CPUs, extensive floating-point computations introduce minor precision errors, but the MPI-CUDA hybrid parallel reconstruction achieves expected results with detailed texture preservation.

Conclusion

This work implements K-T algorithm-based solar high-resolution image reconstruction using MPI-CUDA hybrid parallelism on a single machine, achieving

expected results. Tests with varying process counts demonstrate that the hybrid model outperforms pure MPI parallelism. Particularly, cross-spectrum computation in sub-blocks achieves 2.4× acceleration after GPU porting, substantially reducing overall processing time and fully leveraging GPU intensive computing capabilities. Astronomical observations generate massive datasets with complex processing algorithms that far exceed CPU computational capacity, making CPU-GPU hybrid high-performance architectures a valuable reference for large-scale astronomical computing tasks. Future work should further analyze and optimize algorithms, such as reducing sub-block overlap area to decrease sub-block count without compromising reconstruction quality. Deeper understanding of GPU architectures will enable more computations on GPUs, extending the MPI-CUDA hybrid approach to speckle-masking-based solar high-resolution image reconstruction.

References

[1] Yang Zhongliang, Liang Yonghui, Hu Haojun, et al. Theoretical and experimental research of lucky imaging technique about extended objects[J]. Laser & Optoelectronics Progress, 2010, 47(5): 051004-1-051004-6.

[2] Fang Yuliang, Jin Zhenyu, Liu Zhong, et al. A study of influences of defocus aberrations on high-resolution image reconstruction for data from the New Vacuum Solar Telescope of the YNAO[J]. Astronomical Research & Technology, 2015, 12(2): 183-188.

[3] Xiang Yongyuan, Liu Zhong, Jin Zhenyu, et al. High resolution solar image reconstruction[J]. Progress in Astronomy, 2016, 34(1): 94-110.

[4] Abramenko V, Yurchyshyn V, Goode P, et al. Statistical distribution of size and lifetime of bright points observed with the New Solar Telescope[J]. The Astrophysical Journal Letters, 2010, 725(1): L101-L105.

[5] Sanders J. GPU High-Performance Programming: CUDA in Practice[M]. Beijing: Mechanical Industry Press, 2011.

[6] Chen Tairan, Wang Wei, Wang Feng, et al. The study and application of a high performance UVFITS assembly system based on MPI[J]. Astronomical Research & Technology, 2016, 13(2): 184-189.

[7] Zhong Libo. Research on High-Resolution Solar Image Reconstruction Technology[D]. Beijing: University of Chinese Academy of Sciences, 2015.

[8] Labeyrie A. Attainment of diffraction limited resolution in large telescopes by fourier analysing speckle patterns in star images[J]. Astronomy & Astrophysics, 1970, 6(1): 85.

[9] Knox K T, Thompson B J. Recovery of images from atmospherically degraded short-exposure photographs[J]. Astrophysical Journal, 1974, 193(1): L45-L48.

[10] Weigelt G P. Modified astronomical speckle interferometry "speckle masking" [J]. Optics Communications, 1977, 21(1): 55-59.

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