Research on ROACH2-GPU-Based Cluster Correlator: Design and Implementation of the X-engine Module (Postprint)

Authors: Wang Qunxiong, Niu Chenhui, Haijun Tian, Wu Fengquan, Li Jixia, Chen Xuelei, Gao Jie

Date: 2017-10-20T00:00:00+00:00

Abstract

With the continuous advancement of radio interferometry technology, interferometric arrays are growing in scale and observational capabilities are gradually increasing, which consequently brings about the challenge of real-time processing of ultra-large data volumes. To address this problem, we have developed a general-purpose correlator based on a graphics processing unit (GPU) cluster and applied it to data processing for the "Tianlai Project" : First, according to the correlation computation characteristics of radio signals, computation tasks are allocated to different GPU nodes by frequency band, with reasonable balancing of network load across nodes; then, different GPU nodes independently complete their respective computation tasks and transmit the computation results to storage nodes in real time; finally, the system was successfully installed and deployed according to the design scheme of the GPU cluster general-purpose correlator, and performance testing was conducted based on the requirements of Phase I of the "Tianlai Project" . The computational performance of this GPU cluster correlator is approximately 46% of the theoretical peak performance; compared with correlators based on traditional schemes, GPU cluster-based correlators possess advantages such as short development cycles, strong scalability, and simple deployment.

Full Text

Research on the ROACH2-GPU Cluster Correlator: Design and Implementation of the X-engine Module

Authors: Wang Qunxiong[1], Hao Jie[2], Wu Fengquan[2], Li Jixia[2], Chen Xuelei[2], Tian Haijun[3], Niu Chenhui[2]
[1]China Three Gorges University, Yichang 443002, China, Email: 1276303919@qq.com;

[2]National Astronomical Observatories, Chinese Academy of Sciences, Beijing 100012, China; [3]Central China Normal University, Wuhan 430079, China; Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China

Abstract

As radio interferometry technology continues to advance, interferometric arrays are growing increasingly large, progressively enhancing observational capabilities but simultaneously creating formidable challenges in real-time processing of massive data volumes. To address this issue, we have developed a general-purpose GPU-cluster-based correlator for the "Tianlai" project, integrating the computational and transmission requirements of radio interferometer correlators with the characteristics of radio array signals. The design allocates computational tasks across different GPU nodes by frequency band based on the correlation properties of radio signals, while balancing network load across nodes. Each GPU node independently completes its assigned computations and transmits results to storage nodes in real time. The system has been successfully deployed according to the GPU cluster correlator design and performance-tested against first-phase Tianlai project requirements. The GPU cluster correlator achieves approximately 46% of theoretical peak performance. Compared with traditional correlator implementations, the GPU-cluster-based approach offers advantages including shorter development cycles, superior scalability, and simpler deployment.

Keywords: Radio interferometer; GPU correlator; GPU cluster; Real-time data processing; Frequency-division computation

1. Introduction

The primary processes in radio interferometric array signal processing include: analog-to-digital conversion of time-series signals, Fourier transformation of each signal path, calculation and calibration of interferometric visibilities, noise elimination, and Fourier reconstruction of sky maps. Among these, interferometric visibility computation is the most critical and computationally intensive component, performed by the correlator. The computational load for visibility calculation grows with the square of the array element count [1]. Current international radio interferometric arrays have reached hundreds to thousands of elements, with real-time processing requirements approaching trillions to quadrillions of operations per second. For instance, the Atacama Large Millimeter/submillimeter Array (ALMA) project [2] initially comprised approximately 50 dual-polarization elements and is now essentially complete, with Phase II planning nearly 1000 elements. The Square Kilometre Array (SKA) telescope, currently under international development, will include approximately 250,000 dipole antennas in its low-frequency array and about 200 dish antennas in its mid-frequency array in its first phase. Such massive arrays pose enormous challenges for data acquisition, transmission, and real-time processing, representing a significant international research focus [3].

Traditional solutions employ Application-Specific Integrated Circuits (ASIC) or Field-Programmable Gate Arrays (FPGA) for hardware implementation. However, these approaches suffer from long development cycles, poor scalability, difficult deployment, and high costs. In contrast, Graphics Processing Units (GPUs) not only provide high-performance graphics processing but also exhibit exceptional floating-point computational capabilities and extremely high memory bandwidth, making them highly promising for radio interferometer correlator development. This paper presents a correlator implementation using NVIDIA' s Compute Unified Device Architecture (CUDA) software model combined with hardware. Software-based design offers shorter development cycles, better scalability, simpler deployment, and lower costs compared to hardware implementations, making it a valuable alternative for radio signal cross-correlation.

Previous experiments [4] tested single-GPU computational performance, confirming the tremendous potential of GPUs. However, as element counts increase, single GPUs cannot meet practical interferometric data processing requirements, necessitating design of cross-correlation algorithms for GPU clusters to overcome individual GPU limitations through collective processing power.

2. GPU Cluster Correlator Architecture Design

For the intensive observational data collected by radio interferometric arrays, GPU cluster correlators face two critical issues in real-time processing: rational data distribution and scheduling, and optimization of cluster node computational performance. These challenges are interdependent—poor data distribution leads to unbalanced node loads, while suboptimal node performance affects overall data scheduling. Therefore, GPU cluster correlator architecture must address both simultaneously to achieve optimal system performance.

To solve the first problem, we adopted a frequency-division distributed computing signal processing mode, drawing on correlator technology from the Collaboration for Astronomy Signal Processing and Electronics Research (CASPER) [5]. For the second problem, previous experiments explored various GPU optimization methods.

2.1 Overall Architecture Since performing Fourier transformation first converts time-domain signals to frequency-domain signals, cross-correlation only requires correlation within the same frequency (different frequencies do not correlate). This characteristic enables frequency-division distributed processing with lower computational complexity. Figure 1 illustrates the overall framework of the GPU cluster correlator.

The correlator hardware platform employs the ROACH2 (Reconfigurable Open Architecture Computing Hardware) FPGA processing board developed by the CASPER group, featuring an open, reconfigurable architecture. The F-engine module runs on ROACH servers, handling data sampling and Fast Fourier Transform (FFT) operations. The X-engine module, which performs data cross-

correlation and outputs results, runs on GPU servers. A network switch connects the F-engine and X-engine modules.

Assuming the GPU cluster correlator contains N GPU server nodes corresponding to m frequency-domain signal paths, with each path containing F frequency channels after FFT processing on the ROACH board, each GPU node handles F/N frequency channels. The ROACH board distributes specific frequency bands through the switch to corresponding GPU nodes, which then further divide the channels among their n GPU cores for correlation computation. This frequency-division distributed computing architecture significantly reduces data transmission pressure and exchange complexity [5].

2.2 X-engine Module Structure Figure 3 shows the detailed structure of the X-engine module. In the cluster environment, the X-engine module performs cross-correlation calculations for m signal paths. Each GPU node adopts a CPU-GPU master-slave architecture, with GPU nodes forming a distributed system. The CPU and GPU units within each node share memory and video memory spaces using unified memory addressing [6].

The main program divides into serial and parallel sections. The serial portion executes on the CPU, while the parallel portion includes both inter-node cluster parallelism and intra-node thread parallelism. The CPU initially distributes tasks to GPU nodes, which further partition tasks among GPU cores. After GPU parallel computation completes, results are copied back to memory and output to storage devices by the CPU.

3. Hardware Implementation of the GPU Cluster Correlator

Based on the correlator architecture and computational performance requirements, we designed a GPU cluster-based correlator system. The key implementation challenge is determining the number of GPU nodes. Each GPU node contains n GPU cores (depending on power and space constraints). Testing revealed each GPU core achieves approximately 200 GFLOPS actual performance, giving each GPU node a peak performance of about 4.8 TFLOPS.

For Tianlai Phase I, the total computational requirement is 35.6 TFLOPS (F-engine stage computation is relatively small at 1.2 TFLOPS, requiring only 1 ROACH node). Therefore, the X-engine module requires at least 8 GPU nodes to complete the calculations.

For data transmission, we employ 10GbE network cards. To prevent network saturation during operation, the average transmission rate is maintained at 0.8 GB/s. Before transmission, the original signal undergoes bit truncation from 8-bit to 4-bit effective data. With this configuration, the F-engine and X-engine modules each require 6 10GbE network cards to receive all data completely.

Considering both computational and transmission requirements, the F-engine module uses 6 ROACH nodes, while the X-engine module uses 8 GPU nodes.

Each GPU server is configured with 2 GTX 690 GPUs (each with 2 compute cores), 4 10GbE ports, and at least 4 CPU cores. The 4 GPU cores independently execute parallel correlation computations for fixed frequency channels.

4. Software Implementation of the X-engine Module

Figure 4 illustrates the software architecture within a GPU node. The implementation comprises four main threads and three buffers:

1. **Network Thread**: Receives data from ROACH servers, reorganizes it according to frequency and antenna order, and stores it in GPU buffers. When a data block meets correlation requirements, it is copied to GPU memory and the buffer space is cleared.
2. **GPU Thread**: Performs cross-correlation computations and copies integrated results to CPU buffers.
3. **CPU Thread**: Restructures data in CPU buffers and stores final results in disk buffers.
4. **Disk Thread**: Writes final results to backend storage.

A status variable buffer enables real-time monitoring of thread and buffer states throughout the process.

4.1 Network Packet Format The X-engine module requires specific data packet structures. As shown in Figure 5, each packet header contains M_CNT (packet sequence), F_ID (frequency ID), and source/destination node information. The payload includes sampling times (t0, t1, ···) and signal channels (ch0, ch1, ···). Each packet contains 64 KB of data (excluding header), with adjustable sampling counts and signal paths based on transmission requirements.

5. Performance Testing of the GPU Cluster Correlator

5.1 Error Analysis and Correctness Verification The GPU correlator' s error sources include: (1) signal sampling and bit truncation in the F-engine module, and (2) cross-correlation in the X-engine module. The X-engine correlation, essentially floating-point multiply-accumulate operations, is performed in single-precision to leverage GPUs' superior single-precision performance. Single-precision floating-point (32-bit: 1 sign bit, 8 exponent bits, 23 mantissa bits) provides optimal precision of approximately 10^{-6} to 10^{-7}. During accumulation, floating-point addition requires alignment and normalization, causing rounding errors that accumulate. The maximum error between GPU single-precision and CPU double-precision results is about 10^{-12}. To minimize errors, we employ block-wise summation [7].

For correctness verification, we input simulated white noise with a known time delay and compared the phase-frequency variation with theoretical predictions. The theoretical phase slope is $K = 2\pi T$, where T is the time delay. With

an injected delay of 2.5 ns, the theoretical $K \approx 1.571 \times 10^{-8}$. The measured $K = 1.561 \times 10^{-8}$ yields an error of approximately 0.006365, confirming the correlator' s accuracy.

We also compared results with an FPGA+DSP correlator developed by the Institute of Automation. Figure 6 shows phase diagrams from both systems and their difference. The nearly identical phase patterns and near-zero phase difference (with minor non-zero portions due to noise) validate our GPU correlator' s correctness.

5.2 Computational and Transmission Performance Previous experiments tested single-GPU performance. Here we evaluate a GPU node with different antenna counts. Figure 7 shows computational performance and data transmission rates for the GTX 690. The GPU core performance peaks at 1200 GFLOPS with 120 antennas, approximately 25% of theoretical peak, while overall node performance increases with antenna count. The transmission bottleneck lies in host-device data transfer, limited by PCI-E bandwidth, consistent with earlier findings.

In the cluster implementation, each GPU node corresponds to 4 10GbE network cards with actual peak network rates around 4 GB/s. After bit-shifting operations (converting 4-bit to 8-bit data), the effective network peak becomes 8 GB/s, comparable to host-device transfer limits. Thus, the correlator' s data transmission is constrained by both network and PCI-E bandwidth.

5.3 Other Performance Metrics **Scalability**: Traditional FPGA-based correlators have fixed performance and poor scalability—array expansion requires complete redesign. Our GPU cluster correlator achieves performance-hardware separation through software. Tianlai' s expansion from 16 to approximately 100 dual-polarization antennas can be accommodated by simply adjusting GPU node counts and adding switches, demonstrating excellent scalability.

Development Cycle: GPU correlators use commercially available hardware without custom development. Software modifications accommodate different GPUs and parameters, yielding shorter development cycles than traditional approaches.

Deployment: GPU clusters are easily deployed via network switches, whereas traditional correlators require complex inter-machine wiring.

Linearity: Figure 8 shows the GPU cluster correlator' s linearity—the input power range where output remains linear. The linear range is approximately -12 dBm to 6 dBm, within which results are reliable.

6. Summary and Outlook

This paper presents a highly scalable, general-purpose GPU cluster correlator for large radio interferometric arrays, applied to the Tianlai project. Building

on previous work, we designed a GPU cluster correlator system for 16 dual-polarization inputs. The frequency-division distributed computing architecture, combined with hardware-software co-design, provides excellent scalability. We detailed module functions, implementation, and performance testing.

For Tianlai Phase I (16 dual-polarization antennas), the GPU cluster correlator achieves 35.6 TFLOPS actual performance—about 46% of theoretical peak. While current peak utilization is modest due to hardware procurement targeting this specific scale and suboptimal kernel function efficiency, the cluster' s computational capacity remains unsaturated. Future work will optimize data transmission and kernel functions to improve utilization toward 63 TFLOPS, preparing for Tianlai' s next expansion phase.

References

[1] Harris C J. A parallel model for the heterogeneous computation of radio astronomy signal correlation. Australia: the University of Western Australia, 2009.

[2] Chen Xuelei. Radio detection of dark energy—the Tianlai project. Scientia Sinica Physica, Mechanica & Astronomica, 2015, 45(12): 1358-1366.

[3] Clark M A, La Plante P C, Greenhill L J. Accelerating radio astronomy cross-correlation with graphics processing units. International Journal of High Performance Computing Applications, 2012.

[4] Tian Haijun, Xu Yang, Chen Xuelei, et al. A preliminary study on GPU-based correlators for a radio interferometer array. Publications of the National Astronomical Observatories of China, 2012, 9(2): 32-36.

[5] Huang Jinzeng, Chen Hu, Lai Lushuang. Research and implementation of task schedule method on heterogeneous GPU cluster. Computer Technology and Development, 2013, 23(9): 1531-1533+1539.

[6] Zhang Shu, Chu Yanli, Zhao Kaiyong, et al. GPU High-Performance Computing. Beijing: China Water & Power Press, 2009.

[7] Chen Tianchao, Feng Baiming. Research on error accumulative sum of single precision floating point. Journal of Computer Applications, 2011, 31(6): 1531-1533+1539.

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