中国科学院大学

University of Chinese Academy of Sciences

博士学位论文

射电天文数据实时计算的关键技术研究

作者姓名：戴　伟

指导教师：王锋　教授

中国科学院云南天文台

学位类别：理学博士

学科专业：天文技术与方法

培养单位：中国科学院云南天文台

2019 年 4 月

Research on Key Technologies in Radio Astronomical Data Realtime Processing

A dissertation submitted to

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Doctor of Philosophy

in Astronomical techniques and methods

By

Dai Wei

Supervisor: Professor Wang Feng

Yunnan Observatories

Chinses Academy of Sciences

April 2019

中国科学院大学

研究生学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。

作者签名：

日　　期：2019.5.31

中国科学院大学

学位论文授权使用声明

本人完全了解并同意遵守中国科学院有关保存和使用学位论文的规定，即中国科学院有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：　　导师签名：

日　　期：2019.5.31　　日　　期：2019.5.31

学位论文版权使用授权书

本人完全了解中国科学院大学有关保留、使用学位论文的规定，即：学校有权保存学位论文的印刷本和电子版，并提供目录检索与阅览服务；学校可以公布论文的全部或部分内容，可以采用影印、缩印、数字化或其它复制手段保存学位论文。

本人同意《中国优秀博硕士学位论文全文数据库》出版章程的内容，愿意将学位论文提交《中国学术期刊（光盘版）》电子杂志社，编入CNKI学位论文全文数据库并充实到“学位论文学术不端行为检测系统”比对资源库，同意按章程规定享受相关权益。

保密论文在解密后遵守此规定。

论文作者签名：[signature]　　指导教师签名：[signature]

日期：2019年 5月31日

摘 要

射电数据是射电天文观测的结果，是进行射电天文学研究的核心基础。随着射电设备和观测技术的发展，新一代射电望远镜的涌现，射电天文学家获取天文观测数据的能力得到了空前的加强。射电天文的数据已经以一种实时、顺序、海量和无限的方式到达。如何对观测数据进行实时处理使得射电天文学家面对着前所未有的挑战。

本文针对云南天文台 40 米射电望远镜脉冲星数字终端以及中国新一代厘米-分米波综合孔径望远镜即明安图射电频谱日像仪（MingantU SpEctral Radioheliograph, MUSER）的实时数据处理面临的问题开展研究，具体工作说明如下：

（1） 实时数据采集是天文数据实时处理流水线的起点，本研究分析了当前射电天文数据流传输的特点，针对传统 Linux 操作系统网络协议栈性能较低的问题，采用以内核旁路、零拷贝的用户态空间网络加速技术，实现了万兆以太网络下线速的无丢失的数据采集。该技术已成功应用于云台 40 米脉冲星的数字后端的数据采集。

（2） 近年来，图像处理单元（Graphics Processing Unit,GPU）技术得到了快速的发展，由于优异的浮点运算性能和较好的性价比，天文海量数据的实时数据处理除了采用传统的专用硬件芯片、中央处理器解决方案外，采用 GPU 用于实时数据处理已经成为必然的趋势。本文设计了一个基于 GPU 的天文海量数据处理的实时计算框架，将该计算框架应用于相干消色散的实时数据处理。实现了解码、转置、傅里叶变换、消色散、分通道、逆傅里叶变换、偏振检测、折叠、消噪、归档输出等数据处理模块，并在 40 米望远镜上进行了试观测。此外，针对 40 米望远镜脉冲星观测面临的射频干扰问题，根据观测信号中脉冲星信号和干扰信号的分布特点，利用独立成分分析对混合信号进行分解，分解出独立的射频干扰信号和脉冲星信号，消除射频干扰信号。

（3） 实时处理需要充分利用所有的计算资源，在异构环境下实现天文计算软件的无缝运行和快速迁移，能够提高计算资源的利用效率。本文对 OpenCL 技术实现异构环境下实时天文数据处理的可用性进行了研究，并在此基础上实现

了 MUSER 的射电干涉阵成像网格化算法和洁化算法，在保证了算法运行效率基本不变的基础上，其硬件不再限制于 NVIDIA 的 GPU 环境，为异构环境下实时数据并行处理进一步提供了扩展性。

（4） 采用分布式系统在集群上实时处理数据处理水平扩展计算能力也是一个主要研究内容。为了提高分布式实时计算环境的弹性部署和自动扩展的效率，本文研究了基于轻量级容器 Docker 技术的天文实时处理集群敏捷化构建与部署，并使用容器技术封装 MUSER 现有系统，在不同硬件模式下与物理机和其他虚拟机技术上的封装进行性能测试比较。

论文研究了目前海量天文数据实时处理的若干问题，实际应用性较强。基于用户空间态的数据采集技术可应用数字终端前后端的超高 IO 的数据通讯。利用 GPU 和 OpenCL 构建的异构计算平台下可为实时数据处理提供加速。轻量级容器的虚拟化，可用于实时计算集群的灵活部署和扩展。本文研究内容为天文数据的实时处理提供了参考，为下一步相关工作打下了良好的基础。

关键词： 海量数据，实时处理，实时采集，相干消色散，射电干涉成像

Abstract

Data is the result of astronomical observations and the core basis for conducting astronomical research. With the development of astronomical equipment and observation technology, a new generation of astronomical telescopes has emerged, and the ability of astronomers to acquire astronomical observation data has been unprecedentedly strengthened. Astronomical data has arrived in a real-time, sequential, massive and infinite way. How to process observation data in real time through distribution and parallel means that astronomers face unprecedented challenges.

This paper studies the problems faced by the Yunnan Observatory's 40-meter radio telescope pulsar digital terminal and the real-time data processing of new generation cm-decimeter aperture telescope, MingantU SpEctral Radioheliograph (MUSER). The detailed work description is as follows:

(1) Real-time data acquisition is the starting point of most real-time processing pipelines for astronomical data. This study analyzes the characteristics of current radio astronomy data stream transmission, and uses kernel bypass and zero-copy user-space network acceleration technology to solve the performance of traditional Linux operating system network protocol stack. This paper realizes the lossless data acquisition of line rate under 10 Gigabit Ethernet. The data acquisition framework based on this technology has been successfully applied to the data acquisition of the digital back end of the YNAO 40-meter pulsar.

(2) In addition to the use of dedicated hardware chips, the central processor's traditional solution to deal with astronomical massive data, due to excellent floating-point performance and better cost performance, the use of graphics processing units (GPU) for real-time data processing has become an inevitable trend. A real-time massive data processing framework based on GPU was designed. The framework is applied to the real-time data processing of coherent dedispersion, which realizes decoding, transposition, Fourier transform, dedispersion, subchannel, inverse Fourier

transform, polarization detection, folding, denoising, and archiving. The data processing module was tested on YNAO 40-meter telescope. In addition, for the radio frequency interference problem faced by the 40-meter telescope pulsar observation, the mixed signal is decomposed by independent component analysis, and the independent RF interference signal and pulsar signal are decomposed according to the distribution characteristics of the pulsar signal and the interference signal in the observed signal.

(3) Real-time processing needs to make full use of all computing resources to achieve seamless operation and fast migration of astronomical computing software in a heterogeneous environment, which can improve the utilization efficiency of computing resources. We have studied the application of OpenCL technology in real-time astronomical data processing in heterogeneous environments. Based on this, MUSER's imaging gridding algorithm and cleaning algorithm are implemented, which guarantees that the algorithm's operating efficiency is basically unchanged. It is no longer limited to NVIDIA's GPU environment, providing further scalability for real-time data parallel processing in heterogeneous environments.

(4) Real-time processing of data processing horizontal expansion computing capability on a cluster using a distributed system is also a major research content. In order to improve the efficiency of flexible deployment and automatic expansion of distributed real-time computing environment, the astronomical real-time processing cluster agile construction and deployment based on lightweight container Docker technology is studied, and the MUSER existing system is encapsulated by container technology, and in different hardware. The performance is compared with the physical machine and other virtual machine technology.

The paper studies several problems in the real-time processing of massive astronomical data. The data acquisition technology based on the user space can apply the data communication of the super high IO at the front and rear ends of the digital terminal. Data processing based on GPUs and OpenCL with heterogeneous computing platforms can greatly help accelerate computation and can enable better, faster science. The virtualization of lightweight containers makes it easy to build real-time computing clusters. The research content provides a reference for the real-time processing of

astronomical data, laying a good foundation for the next step of related work.

Key Words: massive data, realtime processing, realtime acquisition, coherent dedisperison, radio interferometric imaging

目 录

图目录

表目录

第1章 引言

天文学历史悠久，是最古老的自然科学基础学科之一，自人类文明诞生以来，就在科学的殿堂占有重要的地位。天文学关注的是科学里最具有前瞻性的问题，重大的天文发现经常推动着自然科学理论向前发展，是基础科学发展的动力和引擎，进入 21 世纪以来，先后有 X 射线、中微子、引力波天文中的射电天文学，已经成为诺贝尔物理学奖的摇篮[1]。

天文学是一门观测的科学，天文学的发展，离不开天文观测工具，也就是望远镜以及其后端的接收设备。从 1609 年伽利略的第一架天文望远镜开始，到 1932 年美国人央斯基自制的射电望远镜，随着探索技术和空间技术的发展，天文观测从最初的可见光、射电波段，发展到红外、紫外、X 射线和伽马射电等整个电磁波谱，再加上中微子、引力波等观测窗口，强有力的观测手段，使得天文学发展到一个全新的阶段，人类对物理世界的认知达到了前所未有的深度。

随着望远镜技术、探测器技术以及信息计算的快速发展，人类来获取天文观测数据的能力得到了空前的提高，天文学已经进入海量数据时代，或者说是数据雪崩的时代。以数据为中心，为数据所驱动已经成为天文学研究的新特点[2]。

1.1 研究背景

目前，新一代各类望远镜所产生的数据已经远远超出了几百年来天文观测数据的总和。

如 Sloan Digital Sky Survey(SDSS)产生的巡天数据达到了 80TB 的数据量；Large Synoptic Survey Telescope(LSST)每晚产生 30TB 的数据，10 年的总数据量达到了 30PB[3]；据估计，低频射电阵 Low-Frequency Array（LOFAR）的 36 个天线站每天产生的数据超过了 100TB[4]；而最终拥有 3000 个天线的 SKA，产生的数据更是其五个数量级之上[5]。国内的澄江一米红外太阳望远镜(NVST)的观测数据一天可以达到近 2TB 的数据；明安图射电频谱日像仪（MingantU SpEctral Radioheliograph, MUSER）MUSER-I 数字接收系统每分钟产生 1.92GB 的数据，

按照每天 10 小时观测时间计算，每天的原始数据量可达 1.2TB。预计到 2020 年，全球将会有累计超过 60PB 的观测数据[6]。天文的数据已经以一种实时、顺序、海量和无限的方式到达。

为了存储并处理如此庞大的数据，研究与发展新的数据处理方法成为当前天文领域的重要课题。海量数据的存储、并行计算、数据挖掘与知识获取成为这个领域的研究重点，面临的关键技术问题包括：海量存储中的数据检索与处理计算、并行算法、基于计算集群或高性能计算环境的科学数据处理、面向海量数据的数据挖掘技术。这个方向已经得到了多年的发展，从国内来看，以 LAMOST、NVST 等新一代望远镜为平台的天文海量数据处理工作已经有较好的发展。特别是LAMOST在数据中通过知识获取技术，对银河系和多波段天体认证等方面取得了长足进步。

针对某些大型望远镜如 SKA 等，根据其科学目标和望远镜自身的特点，出现了海量数据处理的另一个思路，就是尽可能的避免单纯的只保存数据并寄希望于后续处理。而是尽可能的利用当前信息技术的最高水平，通过分布与并行手段把所获得的观测数据实时处理，进而降低数据存储量，确保在最短的时间获得最有效的科学结果。这一思想提出以来，除了推动了对观测数据的处理以外，更对当前天文望远镜整体信息系统带来了新的一种尝试与思考。这一计算模式超越了望远镜的差异，无论是光学、射电、地基或空基均可以有效的利用这一思想结合科学目标提高望远镜的科学产出。

1.2 研究目的与意义

由于近年来的技术发展，阵元数据的不断增加，从拥有 66 面天线的 ALMA，到近千个阵元的天籁计划，以及多国合作的 SKA 项目，面临着高速数据采集、传输和实时处理的巨大挑战。

与批处理海量科学数据处理不同，实时数据处理除了同样面对数据量大这一问题以外，更关键的是需要在一定的时间范围内完成数据采集、处理分析，并根据需要进行快速的数据展现，控制决策。除了对观测数据处理外，实时信息处理技术也是解决天文望远镜的观测控制系统基于多信息源和事件驱动的智能决策调度的关键，并成为一个大型天文望远镜提高观测精度与效率的根本保障。在这

方面进行关键技术的研究，突破当前望远镜各个模块独立工作，缺少统一调度控制这一问题成为了目前推动我国望远镜发展的一个重要工作。

天文信息技术的发展是当今天文科学数据处理的基础。本研究工作拟综合应用当前信息领域的先进技术，包括用户态的网络技术、GPU 和 OpenCL 的并行技术、轻量级容器计算等技术的经验与理论，对海量数据处理和大型望远镜信息系统中所需要的分布式实时信息处理技术进行深入研究，解决当前天文望远镜的实时数据处理、信息可靠交换、数据展现等问题，提高国内大型天文望远镜的研发能力，提高望远镜的科学产出。

1.3 论文主要研究内容

海量天文实时数据使得天文学成为率先迈入大数据时代的学科，虽然海量数据为天文学家的创新研究和科学发现提供了极大的便利，也带来了非常大的技术挑战。海量天文数据涉及光学天文、射电天文、空间天文以及数值模拟和理论计算等多个领域，本文主要对射电天文的海量数据处理的需求进行分析，并针对其中几个具有通用性的关键问题进行分析和研究。

射电天文的数据处理主要由数字终端完成。目前国际上主流的数字终端主要分为三类：专用硬件阵列、密集型 FPGA 处理阵列、FPGA+GPU/CPU 处理阵列。和专用硬件阵列相比，后两类由于采用开源或商用现成技术 COTS(Commercial Off-The-Shelf)，具有相对低廉成本和技术迭代的灵活性，已经成为主流的趋势。海量的数据对传统的射电数据采集、传输、融合、处理、存储与备份、归档与检索、发布与可视化等各个环节提出更高的性能需求。高 IO 的网络互联技术、基于通用体系的计算平台、基于 SIMD 的硬件加速、以及大规模分布式并行计算与云计算等新兴技术的应用，也对射电天文实时处理的软硬件体系、系统架构、执行框架和配套软件算法提出了新的需求。

本论文针对以上需求，主要研究了以下四个部分的内容：

1. 基于用户态的网络加速技术在天文海量观测数据实时传输和采集上的应用；
2. 基于 CUDA 的天文海量观测数据的实时处理；

3. 基于 OpenCL 的天文海量观测数据的实时处理；

4. 基于轻量级容器的实时天文计算集群的快速部署和调度。

通过本文的研究，拟解决云台 40 米射电望远镜脉冲星的观测数据的实时采集和处理以及 MUSER 的观测数据的实时处理需求，以及使用虚拟化技术部署分布式计算集群的需求。

整篇论文研究如何解决当前观测技术的进步而带来的海量天文数据在通用计算平台上的实时采集、实时处理、集群快速部署等问题。由于在论文中结合了具体观测设备的数据处理和分析，论文中也对天文数据处理的相关算法如脉冲星的消色散去干扰算法，射电干涉成像算法的具体实现和优化进行了研究。

本论文结合我国云南天文台 40 米射电望远镜的脉冲星相干消色散终端的海量数据采集和处理需求以及明安图射电频谱日像仪(MUSER)的海量数据处理需求，开展了以下方面的研究工作。

(1) 针对脉冲星相干消色散终端的海量数据采集需求，结合当前在射电领域广泛面临的数据采集需求，对当前在互联网领域的网络加速技术进行了分析比较。从脉冲星相干消色散数据采集的需求对各种网络采集技术进行了深入的分析，设计了一个基于 DPDK 技术的高性能的数据采集框架，并将此数据框架应用于脉冲星的相干消色散后端数据处理集群中；

(2) 设计了一个基于 GPU 的天文海量数据处理的实时计算框架。设计的思想基于底层的硬件加速技术以及环形缓冲队列构建的 pipeline 模型，区别于互联网应用中各种开源分布式的高层框架，以降低开销提升性能为主要目的。并将该计算框架应用于相干消色散的实时数据处理中，实现了从解码、转置、傅里叶变换、消色散、分通道、逆傅里叶变换、偏振检测、折叠、消噪、归档输出等数据处理模块，并在 40 米望远镜上进行了试观测，观测结果表明，该计算框架可以满足当前消色散数据处理的性能要求，该计算框架的升级扩展能力也可满足未来更高带宽的观测。

(3) 实时处理需要充分利用计算资源，在异构环境下实现天文计算软件的无缝运行和快速移植，能够提高计算资源的利用效率。由于在第二部分研究内容中，基于 GPU 和 CUDA 环境为主、CPU 多线程为辅的异构并行存在着算法在不同环

境下重复实现的局限性，本文对 OpenCL 技术实现异构环境下实时天文数据处理的可用性进行了研究，并在此基础上实现了 MUSER 的射电干涉阵成像网格化算法，实验结果表明基于 OpenCL 实现算法对实时处理天文数据具有较高的灵活性，在保证了算法运行效率基本不变的基础上，其硬件不再限制于 NVIDIA 的 GPU 环境，能够在其他图像处理单元以及多核处理器的环境下进行并行，为异构环境下实时数据并行处理进一步提供了扩展性。

(4) 天文数据的实时处理除了在单机环境下尽可能重新利用多核计算资源，提高单机的数据处理能力外，更多的情况是利用多处理机在集群环境中进行分布式计算。为了提高分布式实时计算环境的弹性部署和自动扩展的效率，研究了基于轻量级容器 Docker 技术的天文实时处理集群敏捷化构建与部署，并使用容器技术封装 MUSR 现有系统，并在不同硬件模式下与物理机和其他虚拟机技术上的封装进行性能测试比较。

1.4 论文章节安排

本文的研究工作围绕着天文实时数据处理中的若干问题展开研究，论文围绕着云南天文台 40 米射电望远镜脉冲星相干消色散数据采集和处理以及 MUSER 的数据处理相关技术展开讨论，以如何解决海量天文数据的超高 IO 通信环境下的无丢失数据采集、基于 GPU 的高性能实时数据处理、基于 OpenCL 的异构环境实时数据处理、以及基于轻量级容器的实时分布式计算集群管理等问题为研究重点。

论文的组织结构以各个研究点为支撑，每个章节针对一个研究点进行描述。各个章节相互独立，主要研究的问题说明在章节开头进行，实验结果和工作小结在章节的结尾进行。

第二章对射电望远镜的原理，单天线以及干涉天线阵的原理进行了介绍。并结合本文的实际工作对脉冲星观测和综合孔径成像进行了详细介绍。结合射电观测数据处理现状和发展趋势，探讨新的技术对射电数据处理的意义，包括网络加速技术、CUDA、OpenCL 以及 Docker 容器在未来分布式异构实时数据处理的应用前景。

第三章从云南天文台 40 米射电望远镜脉冲星相干消色散对数据采集的需求角度出发，对目前射电领域常见的后端设计以及面临的数据采集的共性问题进行了讨论，调查研究了目前海量天文数据采集的相关技术。并选取了开源的 DPDK 技术来实现对天文数据实时采集的支持，尤其是满足超高 IO 环境下无丢失的 UDP 数据流。并针对 40 米射电望远镜脉冲星相干消色散的实际数据格式和网络环境，设计和实现了数据采集系统的框架，从整体设计思路，底层关键技术，协议实现等方面进行了介绍。

第四章针对云南天文台 40 米射电望远镜脉冲星相干消色散对数据处理的需求，对基于 GPU 的通用计算在天文数据处理的加速进行了讨论，在第二章数据采集系统的基础上，结合消色散算法原理，描述了在 GPU 上的数据处理流水线的实现，以及在多个服务器多个 GPU 上的并行。并对其中解码、转置、傅里叶变换、消色散、分通道、逆傅里叶变换、偏振检测、折叠、消噪、归档输出等进行了详细说明。

第五章就利用 OpenCL 技术在异构环境下的数据处理进行了研究。讨论了使用 OpenCL 技术移植的天文数据处理程序，在其他图形计算环境以及多中央处理器环境下具备跨平台性和通用性。并结合 MUSER 的数据处理，使用 OpenCL 实现了洁化算法，对其算法效率和可用性进行了介绍。

第六章在前面章节的研究基础上，讨论如何利用虚拟化技术，在分布式计算环境下，对各个节点使用容器技术进行快速部署的可行性。结合 MUSER 的数据处理软件，分析了容器对数据处理性能的影响以及带来的快速部署和弹性扩展的优势。

第七章总结了整个论文的研究工作，指出了当前已经完成的关键技术和在具体实现后依然存在的问题，对下一步的工作进行了展望。

第2章　现代射电天文数据处理相关研究

2.1　射电望远镜原理

与光学望远镜相比，射电望远镜工作在射电波段，其适用性更强大，观测电磁波频率范围更广，在 20KHz 至 3GHz。射电望远镜主要用于测量天体射电的强度、频谱和偏振等，同时收集射电波的定向天线，放大射电信号的高灵敏度接收机等。根据天线总体结构的分类，射电望远镜可分为采用单天线的经典射电望远镜和以干涉技术为基础的干涉仪和综合孔径射电望远镜。

2.1.1　单天线望远镜

和光学反射望远镜类似，单天线射电望远镜的天线大多是抛物面，其的基本原理是，投射来的电磁波被精确镜面反射后，同相到达公共焦点。用旋转抛物面作镜面，更容易实现同相聚焦。为了保证观测效果，射电望远镜表面和理想抛物面的均方误差不得超过 λ/16~λ/10，这样望远镜就能在波长大于 λ 的射电波段上有效地工作。米波或长分米波观测，用金属网作镜面。厘米波和毫米波观测则要求更高，，金属板需要光滑精确，这样的镜面必须达到一定功率，接收来天体投射并汇集到望远镜焦点的射电波，才能为接收机所检测。检测技术要求最弱的电平一般应达 10~20 瓦。射频信号功率首先在焦点处放大 10～1000 倍，并变换成较低频率（中频），然后用电缆将其传送至控制室，在那里再进一步放大、检波，最后以适于特定研究的方式进行记录、处理和显示。

射电望远镜的天线收集来自天体的射电辐射，接收机工作是对射电信号采集、加工、转化成可供记录、显示的标准文件格式，终端设备把信号记录下来，并按特定的要求进行某些处理然后显示出来。评价表征射电望远镜性能有两个基本指标：空间分辨率和灵敏度，前者反映区分两个天球上彼此靠近的射电点源的能力，后者反映探测微弱射电源的能力。射电望远镜通常要求具有高空间分辨率和高灵敏度。

望远镜的分辨率是由它的口径决定的，，分辨角$\theta = 1.22\lambda/D$，其中 D 是口径，是波长。随着望远镜口径的增加，望远镜的灵敏度和仪器的分辨率都会随

之提高，从而能观测到天体的更多细节。因此，大口径望远镜成为天文台的主流，例如澳大利亚 Parkes 天文台的 64-m 望远镜、英国 Jodrell Bank 天文台的 76-m 望远镜、德国马克斯普朗克无线电天文研究所（Max Planck Institute for Radio Astronomy）的 100-m 射电望远镜 Effelsberg 以及美国阿雷西博 Arecibo 305-m 望远镜。2017 年，世界上最大的单口径望远镜 FAST 在中国贵阳建成，FAST 望远镜充分利用了当地特有的天然喀斯特地貌及诸多先进控制技术，望远镜口径达到了 500 米，比 Effelsberg 灵敏度提高将近 10 倍，比美国阿雷西博 Arecibo 305-m 望远镜灵敏度高出 2.25 倍，同时 FAST 的观测仰角从 Arecibo 的 20°提高到40°{Di, 2012 #325}，标志世界射电望远镜发展的新高。Effelsberg 单口径望远镜在厘米波段的分辨率仅和人眼相当[1]，Arecibo 和 FAST 虽然提高了望远镜分辨率，但大型单口径望远镜的控制具有巨大的难度，望远镜的口径也受技术限制达到了极限。

2.1.2 干涉仪与综合孔径望远镜

困扰 射电天文最大的问题是射电望远镜分辨率远不如光学望远镜，缺失天体的细节。无法像光学望远镜那样获得天体的精细照片。早期射电天文学的射电望远镜的口径都比较小，由于分辨率低下，相邻的几个射电源分辨成为一大难题，不可能得到一个射电源结构的信息。

射电干涉仪可以解决以上问题，它由两面抛物面天线构成，两路射电信号由传输线引到接收机进行干涉，结果取决于两路电波到达会焦点的相位两路电波有路程差BC，而且BC随天体的周日运动而变化来自射电点源的单频信号不能同时到达两面天线，要相差一段路程。若这段路程差正好等于波长的正数倍，则这两路信号是同相相加，若这段路程差正好等于半波长的奇数倍，则这两路信号是反相相加，相互抵销。由此当天体作周日运动时，路程差路程差在不断的变化，相加后的输出形成干涉条纹。获得比单天线高得多的分辨率。分辨角不再由单天线的口径决定，使得天文学家有可能利用小口径的天线获得高

[1]https://www.strw.leidenuniv.nl/radioastronomy/lib/exe/fetch.php?media=radio_astronomy_lec_4_ma_garrett.pdf

分辨能力。干涉仪的基线至少可达30.5千米，分辨率比305米天线高100倍，达到光学望远镜的分辨率。

双天线干涉仪只有一维分辨率，不能给出天体的图象。1962 年，英国科学家 Martin Ryle 首次提出综合孔径成像技术(Ryle, 1962)，射电天线阵的建设得到了极大的发展。

综合孔径成像至少要有两面天线，也就是双天线干涉仪。不同的是，其中一面固定，以它为中心，画一个圆，等效于一个“大天线”，另一面可以移动，逐次放到“等效大天线”的各个位置，每放一个地方进行一次射电干涉测量。也可以由许多天线来实现，几面固定，几面移动，甚至全部都固定。各种间距取向的干涉仪测量资料通过数学方法可以求得天空射电亮度的二维分布。也就是得到了被观测天区的射电天图。综合孔径射电望远镜不需要制造口径特别大的天线，用两面或多面小天线进行多次观测达到大天线所具有的分辨率和灵敏度。而且，得到的是所观测的天区的射电天图。设想把抛物面分成许多小单元，小单元的两两组合相当于许多副干涉仪。在馈源上汇集所有两两组合的干涉波。把每个小单元用一小天线代替，由许多小天线组成的许多对干涉仪所得到的信号相加，和一个完整巨大天线的一样。综合孔径要处理异常多的观测数据，计算量特别大，随着计算机的发展，综合孔径射电望远镜的发展才有了可能。

综合孔径成像分辨率$\theta \sim B/\lambda$，其中B为最长基线长度，λ为波长。天线阵的主要排列方式包含东西排列，十字阵，T 行阵，Y 形阵，环形针，螺旋阵和不规则阵。

位于中国密云的综合孔径望远镜和荷兰射电天文研究所（ASTRON）建成的 WSRT（the Westerbork Synthesis Radio Telescope，14 面天线）是一字阵、日本野边山日像仪（Nobeyama Radioheliograph，NoRH）是由 84 面天线组成的 T 形阵，甚大干涉阵（Very Large Array, VLA）采用的 Y 形阵是对十字阵和 T 形阵的突破，Cornwell 的环形天线阵 Crystalline Antenna Arrays。随着位于智利阿塔卡马沙漠的 ALMA（Atacama Large Millimeter/sub-millimeter Array）望远镜、跨越欧洲的大型射电干涉阵 LOFAR（The Low-Frequency Array）、中国明安图

射电频谱日像仪 MUSER 以及即将建设的世界上最大的射电望远镜（螺旋阵列）平方公里阵望远镜 SKA 的建设和提出，使得射电望远镜在灵敏度和分辨率这两个重要指标上得到了前所未有的提高。

2.2 脉冲星观测

脉冲星是快速旋转的中子星，具有极强的磁场，因其辐射束周期性快速扫过地球而得名。自 1967 年发现第一颗脉冲星以来，相关研究就一直是现代天文学的重要内容和热点领域之一。脉冲星提供了在地球实验室中无法得到的极端物理环境，为极端环境下的理论物理研究提供了一个极为难得的研究场所。以脉冲星作为观测对象，可开展高精度计时和守时、天体动力学和天体测量、强场下的引力物理、太阳系外行星、星系和星际介质、超致密物质、以及极端环境下的等离子物理等多方面的研究[7, 8]。

2.2.1 脉冲星观测

Lorimer 认为和其他大多数的天文观测相比，脉冲星观测对数据获取系统(Data Acquisition System)和数据处理系统（Data Handle System）有着更高的要求[7]。例如，获得高精度的到达时间，解析单脉冲内毫秒以及纳秒级的结构，搜寻高速旋转的脉冲星，需要更高的时间分辨率。由于星际介质的影响，脉冲星的信号在传输过程中高频信号会比低频信号先到达，造成轮廓的展宽，降低了信噪比，从而产生色散的现象，影响对脉冲星的观测。例如图 2.1 所示。

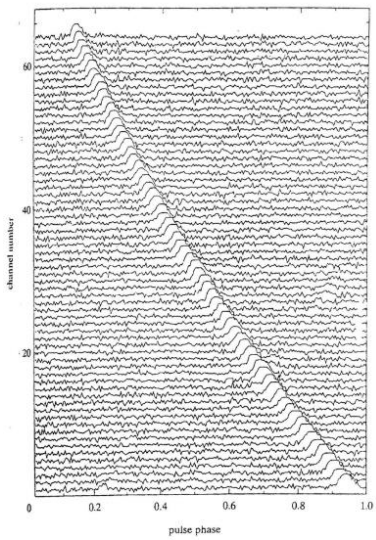


图 2.1 色散引起的信号展宽

通过信号处理去除不同频率上的延迟，对齐信号的这个过程称为消色散。在脉冲星观测当中重要的课题之一就是如何对观测信号进行消色散。

目前消色散的方法主要分为两种：非相干消色散和相干消色散。非相干消色散， 即通过选择一个时间 / 相位起点作为基准点，将子通道（带宽）内的时延计算出来，并对齐到同一相位，最后按周期折叠[9]。如图 2.2 所示。

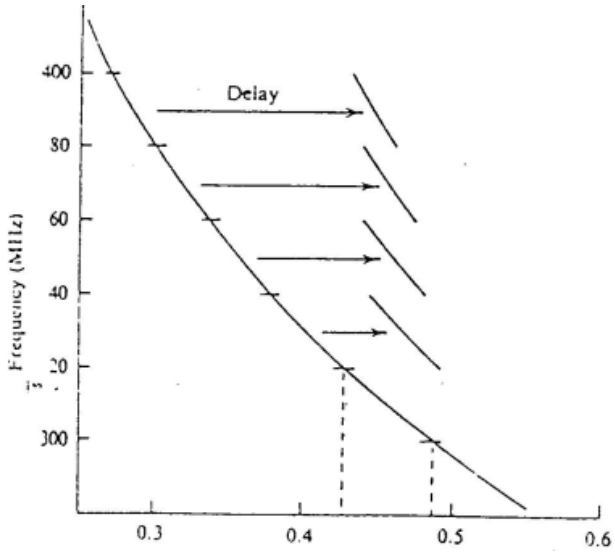
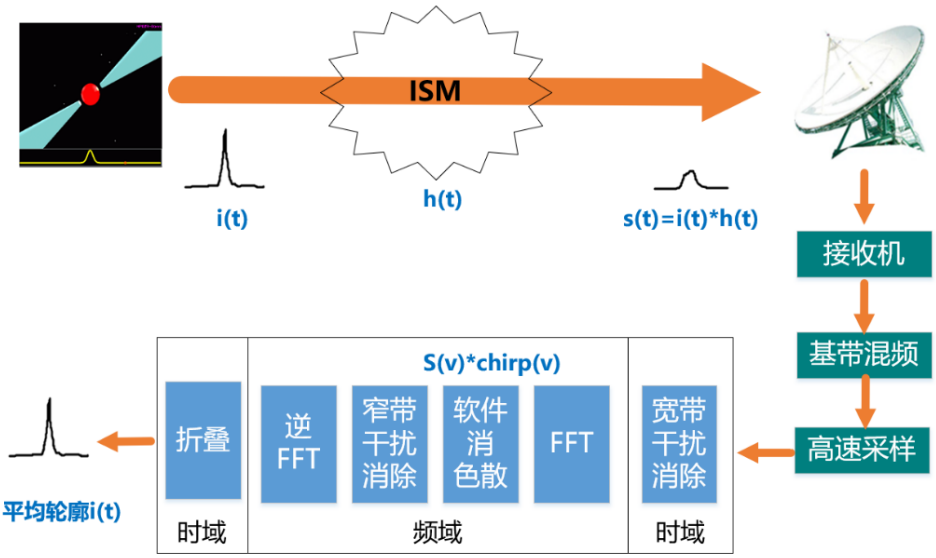


图 2.2 非相干消色散原理

和非相干消色散相比，相干消色散可以完全消除带内色散，观察到脉冲轮廓的细节，有极高的时间分辨率（1/B），缺点是运算量大幅增加。

相干消色散终端的结构如图 2.3 所示。随着近年来脉冲星观测设备向着更高



的频率覆盖和观测带宽演进，灵敏度和观测效率大大提高，但同时也对相干消色散模式下脉冲星观测数据的实时处理的后端设计提出了技术挑战[10-12]。

图 2.3 脉冲星相干消色散数字终端

随着近几年的国内大型射电望远镜的兴建，目前迫切需要自主研发的脉冲星观测终端。国内目前用于脉冲星观测后端设备主要是从澳大利亚 ATNF（Australia Telescope National Facility）引进的 PDFB（Pulsar Digital Filter Bank ）系统，以及从美国国家射电天文台引进的 DIBAS（Digital Backend System)系统，我国目前的脉冲星观测后端设备的研发主要集中于非相干消色散观测模式，并取得了一定的成果，而相干消色散观测模式的研制还处于起步阶段。中国科学院国家天文台云南天文台射电天文研究组正在以 40 米天线为观测测试基础，开展脉冲星观测工作，已经有了较好的前期工作基础[13]。

早期的消色散主要是使用基于硬件的 DSP 设备来实现。随着技术的进步，当前主流的脉冲星观测系统开始采用异构并行模式，结合高速 ADC 采集技术，在 FPGA 硬件平台上利用多相滤波器实现基带转换，万兆以太网技术被用于前端装置和后端处理的数据传输，后端接收被封装在网络数据包中的多通道的基带信号，按照约定的格式解析数据进行后继的存储或处理。后继的数据处理基于 GPU 结合 CPU 的集群进行运算。如图 2.4 所示。

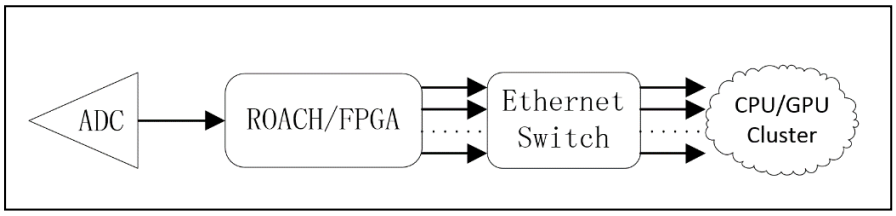


图 2.4 基于异构平台的脉冲星观测系统

如 GASP（Green Bank Astronomical Signal Processor）使用了 20 台 CPU 服务器的集群，实现了 64M 带宽的消色散。在此工作基础上，GUPPI(Green Bank Ultimate Pulsar Processing Instrument)基于 CASPER 的 FPGA，使用了 8 台 GPU 集群，处理 800MHz 带宽的消色散。

2.2.2 相干消色散原理

星际介质是低温的等离子体，其频率响应函数为：

$$\boldsymbol{H(v_0+v)=e^{i\frac{2\pi Dv^2}{(v+v_0)v_0^2}}} \tag{2.1}$$

其中v_0为观测的中心频率。而其中色散常数D由下式得到：

$$\boldsymbol{D=\frac{DM(pc\ cm^{-3})}{2.41\times10^{-4}}(sMHz^2)} \tag{2.2}$$

其中DM为色散值：

$$\boldsymbol{DM=\int_0^d n_e\,dl} \tag{2.3}$$

l 是脉冲星距观测者的距离，n_e是空间中的电子密度。

频率响应函数的倒数称为 chirp 函数。相干消色散就是通过傅里叶变换将接收到的脉冲星信号从时域变换到频域，与 chirp 函数相乘，再将得到的结果通过逆傅里叶变换到时域，得到消除色散后的脉冲星信号。显然多次傅里叶变换是相干消色散处理中最为耗时的部分。实时地对海量数据完成这些运算对后端计算机软硬件系统提出了很高的要求。

2.3　射电干涉阵成像

在干涉仪发展的基础上，1952 年 Ryle 提出综合孔径的概念，并逐步发展为综合孔径成像原理。综合孔径成像原理是由多个小口径天线按照一定拓扑结构进行排列，再对目标进行观测，并进行数据处理，使之等效于一个大口径单天线射电望远镜的观测效果[14]。随着综合孔径成像技术的发展，各国相继建造了综合孔径射电望远镜。

综合孔径射电望远镜是基于综合孔径成像原理，通过一定数目的小口径天线，按照一定的规则排列，组成天线阵，代替一个大口径天线来进行观测。其理论可以概括为八个字：“化整为零、聚零为整”[15]。“化整为零”是指将一面大口径天线简化成许多小口径天线；“聚零为整”是指将许多小口径天线的观测数据合成为观测图像。

许多小口径天线的观测数据为什么能够合成观测图像？根据傅里叶级数，在一定条件下的某个函数可以表示成三角函数（正弦函数或余弦函数）的积分的线性组合。已知，数字图像可以表示为空间域上的二维数组，即，数字图像可以被

分解成许多正弦波或余弦波分量，反之，如果在频域上得到一组正弦波或余弦波，可以通过傅里叶逆变换在空间域上得到对应的图像。综合孔径成像原理中的小天线就是基于这个原理进行图像合成成像的。

在一个由 N 面天线组成的观测天线阵中，任意两面天线可以组成一个基线矢量，通过连接两面天线构造一架相关干涉仪，一共能组成 N*(N-1)/2 个基线（不考虑共轭基线）和 N*(N-1)/2 架双天线干涉仪。干涉仪的输出响应是天体辐射的亮度分布的一个傅里叶分量[16]。通过不同的基线(基线空间长度或指向)可以得到不同的天体辐射的亮度分布的傅里叶分量。通过综合这些傅里叶分量，然后使用傅里叶逆变换就可以获得天体辐射的亮度分布。

一架固定基线的相关干涉仪能够测量到天体辐射的亮度分布的一个傅里叶分量，通过改变基线的长度或空间指向便可以得到天体辐射的亮度分布的一系列傅里叶分量。干涉仪的输出响应也称为可见度函数，记为$V(u,v)$，将天体辐射的亮度分布记为$I(x,y)$，则有:

$$\boldsymbol{V(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}I(x,y)\cdot e^{-i2\pi(ux+vy)}dxdy} \tag{2.4}$$

理论上，可以通过对可见度函数$V(u,v)$直接进行傅里叶逆变换，得到天体辐射的亮度分布$I(x,y)$，即：

$$I(x,y)=\frac{1}{(2\pi)^2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}V(u,v)\cdot e^{i2\pi(ux+vy)}dudv \tag{2.5}$$

遗憾的是，由于天线数目有限，干涉仪的输出响应是有限的，即无法获得天体辐射的亮度分布所有的傅里叶分量，导致可见度函数的不完整性，不能直接通过对不完整的可见度函数进行傅里叶逆变换得到天体辐射的亮度分布。这种不完整性，实际上是由于天线阵只是对可见度函数进行了采样。根据数字信号处理原理，在一定条件下，通过采样数据就能够近似恢复出真实信号。同样地，对于综合孔径射电望远镜，通过对采样的可见度函数进行一系列处理，也能够近似恢复出天体辐射的亮度分布。

理论上，采样点越多，恢复出的天体辐射的亮度分布越真实。那么如何才能增加采样点的数目呢？一是增加天线的数目，天线数目的增加，组成的干涉仪数

目也就越多，干涉仪的输出响应也就越多，即可见度函数采样值越多；二是增加空间频率 u 和 v 的数目，u 和 v 的值取决于天线位置、基线数目、观测时间以及观测波长等，将在下文更为详细介绍。

总而言之，综合孔径成像原理为射电天文开启了建造综合孔径射电望远镜的大门。综合孔径射电望远镜基于综合孔径成像原理，对许多小口径天线采集到的观测数据进行一系列数据处理，来等效一个大口径射电望远镜的观测结果。以较低的造价和成熟的制造技术建造的综合孔径射电望远镜，实现了比传统单口径射电望远镜更好的观测效果。

2.4　射电数据处理

射电数据是射电天文观测的结果和进行射电天文学研究的基础。随着射电天文设备和观测技术的发展，新一代射电天文望远镜和终端设备的涌现，观测灵敏度和分辨率的不断提高，大口径天线和天线阵，超宽带馈源，以及多波束接收等技术的发展，使得射电天文学家获取射电观测数据的能力得到了空前的加强。射电天文观测大数据时代的来临，给射电天文技术的发展带来了巨大的挑战和机遇。

射电望远镜通过天线接收电磁信号，再通过接收机进行频率迁移和放大后，记录数据进行离线处理，或者直接进行数据的实时处理。起初射电信号处理终端是模拟终端，模拟终端的工作带宽受到电路器件的限制，加之设计难度较大，一旦定型很难改动。1961 年，Weinreb 首次采用数字量化和采样来设计光谱探测仪[17]。之后，随着计算机技术和信号处理技术的高速发展，数字终端系统已经取代模拟终端，将模拟信号数字化进行处理。数字化的终端与模拟终端相比，具有很高的灵活性，既降低了开发难度，又提高了升级维护的灵活性，具有很大的优势。

射电终端系统进入数字系统以来，已经出现了中央处理器(Central Processing Unit,CPU)、专用集成电路(Application Specific Integreated Circuit，ASIC)、现场可编程门阵列(Field-Programmable Gate Array, FPGA)等技术。

CPU 由于只有单个或少数核心，适合串行顺序计算，整体计算能力和计算效率不高，故被采用于离线数据处理或者实时性要求不高的准实时的应用场合。

ASIC 技术，顾名思义，是为专门目的设计的集成电路，其特点是定制需求，优势是性能高，功耗低，体积小。如 ALMA[18]和 eVLA 的 WIDAR 相关器[19]，都是采用了定制的 ASIC 芯片来解决计算密集的数据处理问题。但是由于硬件开发成本较高，开发周期也较长，面临着后期升级维护困难的问题。

FPGA 解决了 ASIC 芯片的不够灵活的问题，可以通过可编程架构，对数百万的逻辑门进行实现灵活定制，能够以并行的方式实现算法进行数据的并行处理，而且由于其处理时间延时固定，适合进行实时数据处理，得到广泛的应用，特别在数字采样和预处理方面。目前，射电天文广泛使用的 FGPA 设备有美国美国伯克利大学天文信号与电子学研究中心 CASPER(Collaboration for Astronomy Signal Processing and Electronics Research)平台[2]、欧洲的 JIVE(Joint Institue for VLBI in Europe)设计的 Uniboard 平台[3]、澳大利亚国立射电天文台 ANTF(Australia Telescope National Facility)的 PDFB3、以及澳大利亚联邦科学与工业研究组织 CSIRO(Commonwealth Scientific and Industrial Research Organisation)的 Redback[20]。其中 CAPSER 的平台应用最为广泛。CAPSER 的 FPGA 硬件经过了十几年的发展，从早期的 iBOB(Interconnect Break-out Board)、BEE2(Berkeley Emulation Engine II)，发展到后来的 ROACH（Reconfigurable Open Architecture Computing Hardware）、ROACH2（Reconfigurable Open Architecture Computing Hardware 2），以及为平方公里阵 SKA(Squared Kilometer Array)设计的 SKARAB（Square Kilometre Array Reconfigurable Application Board）。澳大利亚 Parkes 望远镜、美国 Green Bank 射电望远镜、射电望远镜阵列 ATA 和 MeerKAT,美国下一代数字 VLBI 终端 RDBE 都采用 CAPSER 系统。随着 FPGA 硬件持续的技术进步，逻辑处理单元的数量从数万增加到近千万，内存带宽从数 G 增加到数百 G，ADC 单元的采样精度和采样带宽也不断提高。

由于传统的 CPU、ASIC、FPGA、GPU 都有着自己明显的优缺点，为了在性能、成本、延迟和吞吐量上寻找平衡，现在的趋势是通过异构的计算平台来构建射电天文仪器的数据处理平台。使用 FPGA 硬件组成的并行可重构的计算单元

[2] http://casper.berkeley.edu
[3] http://www.radionet-eu.org/uniboard

(Reconfigurable Computing Elements,RCE)，做为前端负责信号采集与预处理。由通用 CPU/GPU 的计算机集群做为后端，CPU 负责控制数据流和串行为主外围计算，GPU 负责信号数据的复杂的运算。前后端通过工业以太网进行互联，前端将预处理的数据发送给后端进行处理。前端根据需要对数据进行约简、重新排列和打包操作。另外，在后端的 CPU/GPU 上将具体的算法进行并行化，如射电信号的相关、射电干涉与干涉成像、脉冲星搜寻以及暂现源探测等，都需要在并行计算系统上进行并行和优化。

2.5　射电数据处理技术与发展趋势

2.5.1　射电数据采集技术

现代的天文望远镜的观测灵敏度得到极大的提高，同时也带来海量数据实时传输和采集的需求。目前普遍采用工业标准的互联解决方案如以太网来作为数据传输技术。如支持了世界各地数十个射电望远镜数字终端的 CASPER 平台，其核心设计思想，就是使用工业标准的商业的互联技术，连接通用的计算单元。科学家们在开源的通用的硬件平台、软件库、开发工具进行仪器设备的快速设计、升级和更新，减少工程成本和时间。

使用以太网作为互联技术简化了不同的数据处理硬件之间的连接。用以太网技术将负责信号采集的 FPGA 单元，和 GPU 集群连接起来，可以充分利用 GPU 的强大的计算能力进行数字信号处理，提高性价比和灵活性。

由于观测需求要求更高的测量带宽和采样精度，导致数字基带转换器产生的观测数据激增。为了提高数据传输速率，新的数据传输系统已经开始逐步采用万兆甚至 4 万兆的以太网技术。万兆以太网技术的引入，对在 CPU/GPU 集群上运行的数据采集的设计提出了更高的要求。必须充分考虑到硬件底层、系统架构和操作系统的细节，最大的优化数据的传输和处理效率，允许在更高的带宽动态范围和精度上进行观测。

目前比较流行的开源的后端数据采集软件有 guppi_daq 数据采集系统、psrdada 等。guppi_daq 数据采集系统是 GUPPI 脉冲星后端的数据采集软件。guppi_daq 数据采集系统的代码由 Paul Demorest 开发，其开发源码可在

https://github.com/demorest/guppi_daq 数据采集系统获取[21]。数据在 10G 以太网环境下封装为 UDP 包发送给数据采集机[22, 23]。由运行在数据采集系统上的 guppi_daq 数据采集系统负责接收数据并保存至磁盘上。guppi_daq 数据采集系统主要基于 C 语言开发，部分监控和诊断程序基于 Python 开发。guppi_daq 数据采集系统使用多线程模式工作，线程之间通过基于共享内存的 RingBuffer 传输数据。由网络处理线程负责接收 FPGA 生成的 UDP 数据包，由于 UDP 和 TCP 相比速度更快，但是是不可靠的协议，网络处理线程需要检查数据包内自定义的序号来判断数据是否有丢失，将收到的数据保存在缓冲区内，如果数据丢失则填充为 0。写盘处理线程将已经写满的缓冲区的数据转换为 PSRFITS 格式保存在磁盘上。

PSRDADA 是一个开源软件项目，用于支持分布式系统中数据采集和分析的开发,主要是用于脉冲星天文观测中的基带数据记录和处理[24]。PSRDADA 的模块化设计包括：使用多线程和进程模式分别处理数据的传输、命令和控制、以及数据简化处理；进程间的数据传输使用了基于共享内存的 ringbuffer 和 internet 的传输协议；通过基于 web 的用户接口、基于文本的 socket、配置文件和脚本程序来监控各个进程。PSRDADA 由澳大利亚 Swinburne 大学的脉冲星的研究组负责维护开发，已经被用于 CASPSR、BPSR、APSR 后端的数据采集和处理。CASPSR（CASPER Parkes Swinburne Recorder）是新一代的基带数据记录和处理系统，由 IBOB 板，2 个节点的 CPU 服务器群和 4 个节点的 GPU 服务器群构成，IBOB 板负责模数转换，输出双路最大 400Mhz 带宽的数据，数据通过 2 个 10GBE 连接到 CPU 服务器，CPU 服务器负责缓冲数据并利用 UDP 数据包将数据流分发到 GPU 的计算节点中。采用 UDP 而不是专用基于 DMA 硬件来传输数据是为了降低成本并且提高了系统重部署和开发的灵活性。APSR（ATNF Parkes Swinburne Recorder）是 Swinburne 大学和 ANTF(Australia Telescope National Facility)联合研制的新一代带数据记录和处理系统，用于高精度脉冲星时间观测，由 PDFB3 负责输出双路最大 1Ghz 带宽的数据，通过 4 个 10GbE 万兆以太连接输出到由 20 个节点的集群中[25]。BPSR（The Berkeley Parkes Swinburne Recorder）是用于 Parkes 多波束接收器的高分辨率的数字录波器组的数据获取和处理系统，13 个 IBOB 板

采集的信号通过 10GbE 网络直接分发到 13 个服务器节点上[26]。CAPSR、BPSR、APSR 均使用 psrdada 实现数据的分发管理和监控。

guppi_daq 数据采集系统和 psrdada 均是开源软件，其源代码可以从互联网下载。通过对 guppi_daq 数据采集系统和 psrdada 的源代码分析可以看出，两个软件均是基于 socket 的 udp 编程来实现数据的接收和发送，在编程时工作线程可以针对 CPU 的亲和性（affinity）进行调度，使用基于 POSIX 或者 System V 的共享内存和信号量机制实现的 ring buffer 控制进程或线程间的数据通讯。随着观测数据的速率的增加，传统的网络编程方式已经不能满足万兆网络下高速率的数据包的处理，如 guppi_daq 数据采集系统在 800MB/s 的数据采集速率下，已经出现了数据的丢包[23]。

万兆以太网理论最大的数据传输速率约为 1.25GByte/s，但是传统的网络编程模式已经不能满足万兆以太网下数据包的采集和处理，对操作系统进行系统参数的调优，采用更大的 MTU 值，只能有限的提高网络吞吐率。为了最大限度的利用万兆网络适配器的带宽，允许前端以更高的带宽和采样精度输出数据，必须借鉴新的方法和技术。Alession Magro 在为 Medicina BEST_II Array 设计的数据处理流水线中，使用了万兆以太网连接了基于 ROACH 板和基于软件的处理节点。数据被封装在自定义的 SPEAD(Streaming Protocl for Exchangging Astronomical Data)格式的 UDP 数据包中。SPEAD 的格式主要为射电天文装置定义标准的 UDP 流的数据输出格式。每个 beam 的输出速率为 640Mbps，8 个 beam 的总输出达到 5.12Gbps。在设计数据接收程序时，为了避免内核和用户空间的多次复制以及 read 或者 poll 造成的多次系统调用的上下文切换严重影响接收效率，没有使用 socket 编程接口，而是采用了 PACKET_MMAP 技术接口，提高了数据接收的效率[27]。Ammendola Roberto 在研究高能物理试验中的在线事件处理系统中，使用了 PFRING 和 Direct NIC Access(DNA)驱动来减少从网络适配器采集数据到复制到 GPU 内存中的传输延迟，有效的增加了内存和网络链路的吞吐率[28]。这些技术主要基于内核旁路（Kernel bypass）、零拷贝技术(Zero-copy）的技术，将数据包的处理放入内核态或者用户态进行，获得了极大的性能提升。如图 2.5 所示。

图 2.5 传统的网络协议栈与内核旁路

2.5.2 GPU 与 CUDA

图像处理单元 GPU 与中央处理器 CPU 类似，只不过 GPU 的设计是专门为执行复杂的数学和几何计算。这种复杂的计算是图像渲染所必需的，最初设计 GPU 的目也正是负责图像渲染。随着 GPU 的不断发展，拥有强大计算能力的 GPU 已经发展成为高度并行化和多线程的多核处理器。这意味着，GPU 已经不再局限于图像渲染，在并行计算和浮点计算等方面，GPU 的计算性能可以是 CPU 的计算性能的上百倍，也因此被很好地应用了在人工智能、深度学习、大数据等技术领域。

CPU 与 GPU 在硬件构造上有所不同，如图 2.6 所示。CPU 中大多数的晶体管被用于缓存和控制逻辑，其运算单元并不多，而 GPU 则相反，GPU 中大多数的晶体管被用于运算单元，一个 GPU 中包含上百个甚至上千个处理器核心，而通常一个 CPU 中只有几个处理器核心，缓存和控制逻辑被大大减少。

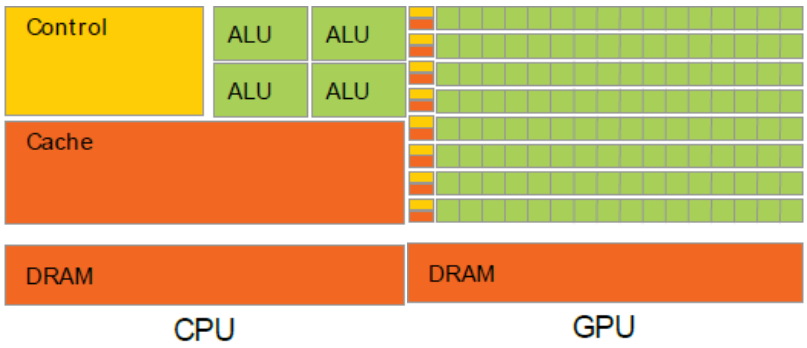


图 2.6 CPU 与 GPU 硬件构造

这种硬件结构设计上的差异，导致了 GPU 和 CPU 的计算性能存在巨大差异。特别地，GPU 适合处理那些能够表示为并行计算的问题，由于在每个运算单元上执行的计算任务相同，对复杂逻辑控制的要求比较少，也不需要太大的数据缓存，在这种计算情况下，GPU 的计算性能比 CPU 的计算性能要高得多。

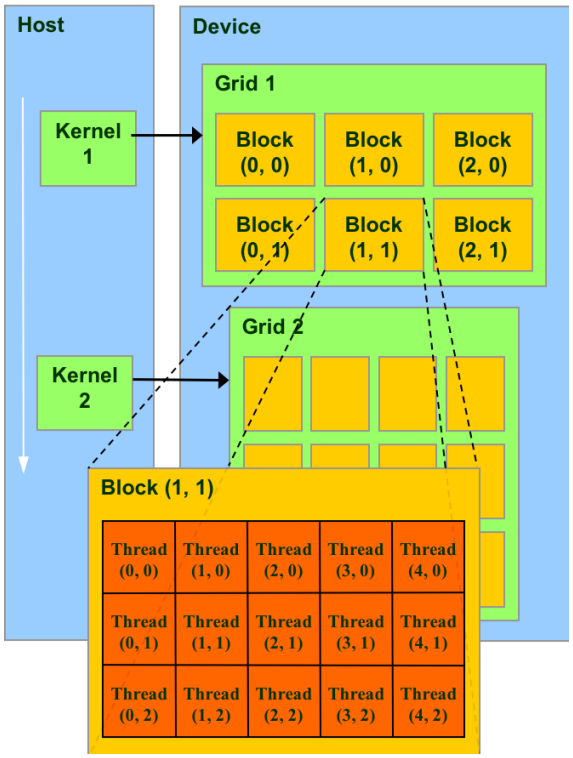


图 2.7 CUDA-GPU 编程模型[4]

这种计算性能上的差异，为计算机软件开发人员提供了一条加速程序执行的思路。通过借助 GPU 强大的并行计算能力进行并行计算，将各个计算任务或数据映射到 GPU 的每个运算单元中，实现任务或数据并行计算，从而提高程序执行效率。通过这种“多个工人同时干活”的思路，让适合 GPU 进行并行计算的程序执行效率大大提升。随着高性能处理器的发展，除 CPU 和 GPU 外，其他类型的处理器，例如 CELL、DSP、FPGA，也逐渐向多核处理器方向发展。现今，在 PC 以及移动设备中一般包含多核的 CPU 和众核的 GPU 这两种处理器。

为了便于计算机软件开发人员借助多核或众核处理器快速开发并行程序，一系列 GPU 通用计算技术随着产生。2003 年，提出 GPU 通用计算（General Purpose GPU，GPGPU）[29]，但是在 2007 年 NVIDIA 推出通用统一计算设备架构（Compute

[4] http://cuda-programming.blogspot.com/2013/01/thread-and-block-heuristics-in-cuda.html

Unified Device Architecture，CUDA）[5]之前，GPGPU 受限于硬件可编程性和开发方式的制约，并行程序开发难度较大，GPGPU 技术没有被广泛地关注与应用。随着这类 GPGPU 技术的不断发展与完善，并行计算编程变得越来越容易，并行计算技术在大规模数据处理领域得到了广泛的应用。

2.5.3 OPENCL

统一计算设备架构 CUDA 是一种由 NVIDIA 公司推出的通用并行计算架构，该架构使 GPU 能够解决复杂的计算问题，让并行计算编程变得简单。从 CUDA 1.0 版本到最新的 CUDA 10.0 版本，CUDA 架构在并行计算中已经相对成熟，并被广泛应用在各个研究领域。随着并行计算的需要，CUDA 架构让 GPU 在高性能计算领域中发挥着巨大优势。采用 CUDA 架构让用户通过简单、常用的高级编程语言对 GPU 进行控制，从而让并行程序高效率地执行。美中不足的是，CUDA 架构只能适用于 NVIDIA 公司制造的 GPU，不适用于其它公司制造的 GPU 或其他多核处理器，导致采用 CUDA 开发的应用程序可移植性较差。基于此，开放运算语言 OpenCL 应运而生。

OpenCL 最早由 Apple 公司提出，由 Khronos Group 制定相关技术标准，2008 年 12 月制定了 OpenCL 1.0 规范，最新发布的是 OpenCL 2.2 规范。Khronos Group 中包括很多厂商，例如 Intel、AMD、IBM、Apple 等等，值得注意的是，NVIDIA 也是 Khronos Group 的成员，并且参与了 OpenCL 规范的制定。特别地，Khronos Group 通过协调各个厂商，从而使 OpenCL 具备跨平台特性。

OpenCL 是异构系统的开放式开发语言，它为并行计算提供了一个跨平台的统一标准，并提供一种通用计算的应用程序接口（Application Program Interface，API），来调用多核计算设备进行并行计算[30, 31]。采用 OpenCL 进行并行计算编程，开发人员通过调用相关 API 来获取可用的多核计算设备，从而让 OpenCL 并行程序在多核计算设备中执行。在异构系统中，OpenCL 的这种跨平台特性被广泛地应用在很多并行计算领域。值得期待的是，机器学习框架 TensorFlow 也正在积极支持 OpenCL，到时将不再苦恼于没有 GPU 而不能并行执行机器学习算

[5] https://developer.nvidia.com/cuda-toolkit

法的烦恼。

CUDA 和 OpenCL 虽然具有相同的目标，都是借助高性能计算设备进行并行计算，从而实现对程序进行加速。但是，借助 OpenCL 的跨平台特性，可以让并行程序在不同的平台或设备上运行，甚至包括普通的多核 CPU 以及其他类型的多核处理器。总的来说，CUDA 是具备完整工具包、发展成熟的开发平台，但只针对单一供应商 NVIDIA，而 OpenCL 适用于并行计算的同时，也是一个开放的标准，具备了跨平台特性。CUDA 与 OpenCL 对比见表 2.1。

表 2.1 CUDA 与 OpenCL 对比表

对比项	CUDA	OpenCL
目标	并行计算	并行计算
支持者	NVIDIA	Khronos Group
计算设备	NVIDIA GPU	多核 CPU、GPU、FPGA 等
最新版本	CUDA 10.0	OpenCL 2.2
编程方式	多线程编程	多线程编程
编程语言	C、C++、Java、Python 等	C、C++、Java、Python 等
特性	开源、高效、稳定	开源、高效、跨平台

2.5.4　云计算与分布式

目前，基于虚拟化技术进行应用的快速部署是比较通用的方法。虚拟化技术所提供的环境隔离与资源共享非常适合于复杂天文软件的应用场景，它通过隔离资源，加以抽象实现多用户共享，可实现资源最大化的利用率。但当前主流的虚拟化系统，如 VMware 的 ESX/ESXi、微软的 Hyper-V 等，均是价格不菲的商业产品。开源的 OpenStack 一直面临较难部署和管理的问题，其可用性、稳定性和易用性均有待考察。

通常情况下，虚拟机采用的虚拟化技术都需要提供完整的虚拟硬件环境，以便让一个完整的操作系统安装在虚拟机中。但这种方法存在一定缺陷：(1)多个虚拟机无法共享底层操作系统的功能；(2)在天文数据处理应用仅需某些软件和部分资源来运行的情形下，虚拟完整的操作系统会造成相当程度的资源浪费。因此，

虚拟化技术虽然可以在一定程度上解决快速的天文软件部署与应用问题，但整体来看，这样的实现方法效率较低。

新出现的容器技术提供了一种新的轻量级虚拟化技术解决方案，其中 Docker 是当前应用最广的容器技术之一。Docker[32]是一种基于 Linux 容器技术的应用容器引擎，由 dotCloud 开源，让开发者将应用程序、所需依赖的运行库文件打包并移植到一个新的容器中，然后发布到运行着 Linux 操作系统的机器上使用(目前也支持 Windows 和 Mac 操作系统)。一方面，Docker 拥有优秀的隔离性和可移植性，可以在保持环境一致性的同时，实现软件的敏捷部署；另一方面，Docker 的易操作性可以降低学习曲线以及搭建维护成本。因此，Docker 容器的出现可以很好地满足对复杂天文软件的部署搭建需求。

此外，Docker 还可支持集群上的容器部署与运行，这就意味着可以通过 Docker 技术，实现单机软件的集群化部署，实现并行化的管理和运行[33]。

第3章　基于 DPDK 的无损高性能分布式数据采集技术

3.1 DPDK 技术

高性能的射电数据采集系统系统必须让每个计算节点以最小的开销、高速、无丢失地捕获前端 FPGA 硬件的数据。占用过高的系统资源必将大大影响后继的实时数据处理。

前端和后端的通讯一般都是高速以太网链路，使用 UDP 协议。在前端，每组的观测数据按照自定义的数据帧格式进行封装。数据帧的首部信息包括观测时间、通道信息、序号等控制字段，数据部分包括固定数量的采样数据。为了提高传输效率，UDP 包的大小一般为 9K 字节，采用以太网的超长帧进行传输。数据采集系统系统运行在后端的每个计算节点上，从高速以太网链路上可靠地接收封装在 UDP 协议包中的观测数据、检测数据丢包、按照策略缓冲数据、将数据保存到磁盘上或者交由数据处理系统进行处理。

超高的速率给数据采集系统系统带来了巨大的压力。使用了传统的 socket 接口，如 psrdada 和 guppi_daq 数据采集系统等，在较高的网络输入输出环境下会出现频繁的数据丢包和较高的系统资源占用。只能通过在 Linux 环境下调整内核参数和网络参数来改善。改善程度有限，无法充分利用网络的传输性能。而目前出现了一些网络加速技术，则可以让网络流量绕过操作系统内核，减少网络数据在内存中的拷贝次数。这些基于内核旁路（Kernel bypass）、零拷贝技术(Zero-copy）的技术已经被应用于一些数据采集系统系统，大幅提升了数据采集系统系统的性能。例如 CHIME 系统使用了 NTOP 公司专有的 PF_RING ZC 技术，VEGAS 的 HashPipe 使用了 packet_mmap 技术。此外还有，Melanox 公司的技术 libvma，可用配合该公司的网络适配器使用。

DPDK(Data Plane Development Kit)是 Intel 公司推出的开源技术，提供了高性能网络开发函数和驱动的支持，用于在通用计算平台上支持高速的网络数据传输处理。目前已经发布的版本是 18.08 版本。虽然国外也有人关注到这一技术的发展，但还没有正式使用到射电观测系统当中。

DPDK 除了使用了零拷贝和内核旁路技术来进行高速的数据包处理外，还提

供了多核架构下的线程和进程的调度和执行框架，支持 NUMA 体系下管理线程和内存的亲和性来进行并行程序的开发。可以将各个不同任务的线程绑定到不同的 CPU 核心，节省线程调度、高速缓冲访问以及内存访问的性能开销。DPDK 的内存管理使用了巨页技术，减少了 TLB Miss 和缺页中断，显著提高应用程序的性能。DPDK 设计了无锁的环形队列，支持面向 CPU 核心的对象池的管理。为了解决异步的中断模式的效率问题，DPDK 在接收网络适配器数据时不再依赖硬件中断，而是通过轮询去获取网络适配器的数据。此外，DPDK 对硬件有着良好的支持，支持 x86，POWER 和 ARM 等主流 CPU 架构，支持大多数主流厂商的网络适配器，可以在多个平台下移植。基于 BSD License 开源，目前已经成为 Linux 基金会下项目之一，拥有广泛的社区和多个开源的应用。

3.2 基于 DPDK 的实时数据采集

3.2.1 框架设计

在数据采集系统的实际部署和使用中，面对目前射电天文数字终端的数据特点，直接选择现有的开源的互联网的解决方案不能完全解决问题。在射电天文领域需要一种新的数据采集框架，可以满足越来越高的 IO 数据的采集的需求。框架关注于数据的实时采集和输出，所以采用 C 和 C++等语言实现，而不采用解释性或者脚本语言开发，能够实现接近网络适配器理论带宽的线速的数据采集。

数据采集系统框架基于多线程构建，包括主控线程、网络线程和输出线程。主控线程负责初始化运行环境、管理其他线程、与远程控制程序进行通讯、接收执行指令并响应；网络线程负责从网络适配器接收以太网帧中的数据；输出线程负责将内存中的数据写入到存储介质用于离线存储或者输出到环形队列中用于构建高速的数据处理流水线。

主控线程只有一个副本。网络线程和输出线程可以有多个副本运行。可以解决各种情况下，网络数据的接收和分发。如图 3.1 所示。

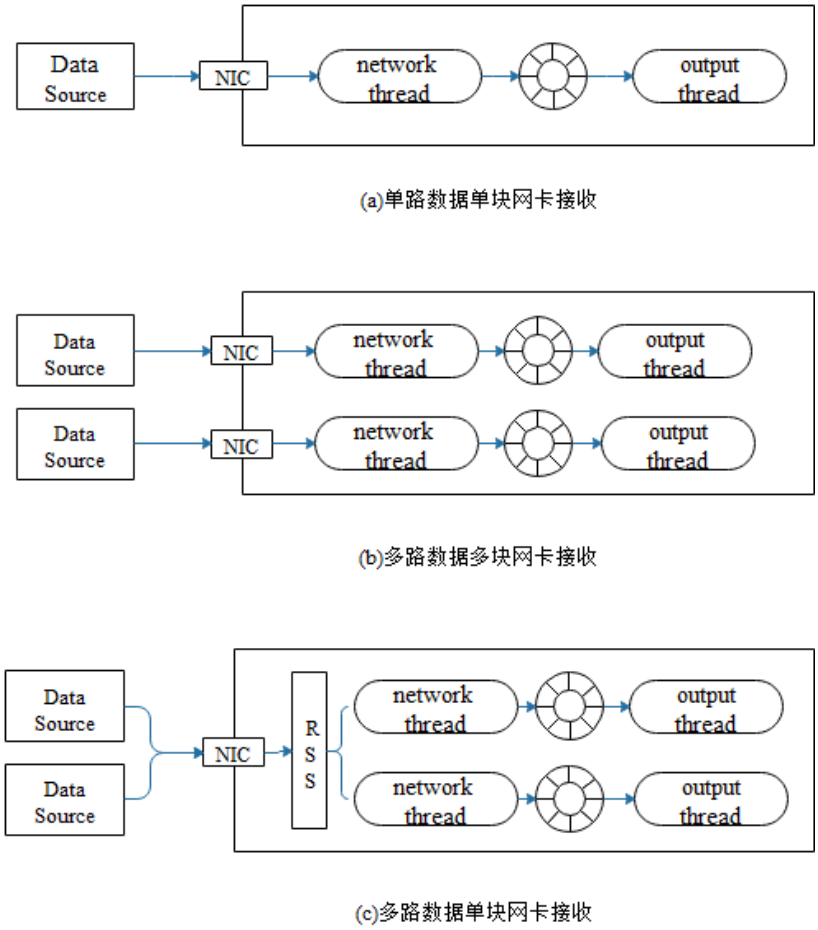


图 3.1 数据采集模式

图(a)是典型的 1 对 1 数据接收的模式。单路数据流通过单个网络适配器，由 1 个网络线程负责接收，由 1 个输出线程负责数据的存储或者分发。

图(b)是 1 对 1 数据接收模式的扩展。可以由单个计算节点通过不同的网络适配器接收不同的数据流分别进行处理。

图(c)是多对 1 数据接收模式。可以由单个计算节点通过同一块网络适配器接收不同的数据流分别进行处理。通过网络适配器的多队列技术，RSS(Receive-Side Scaling)接收方扩展，将数据帧分配到不同的接收队列中，每个接收队列分配一个网络线程进行数据接收。这种模式适用于射电相关器或者射电频谱仪的数据传输。

本文利用 DPDK 技术设计和实现了面向射电天文的高速数据采集系统框架。框架的计算模型基于 DPDK 的线程库设计，其多线程的启动和管理模式如图 3.2 所示。

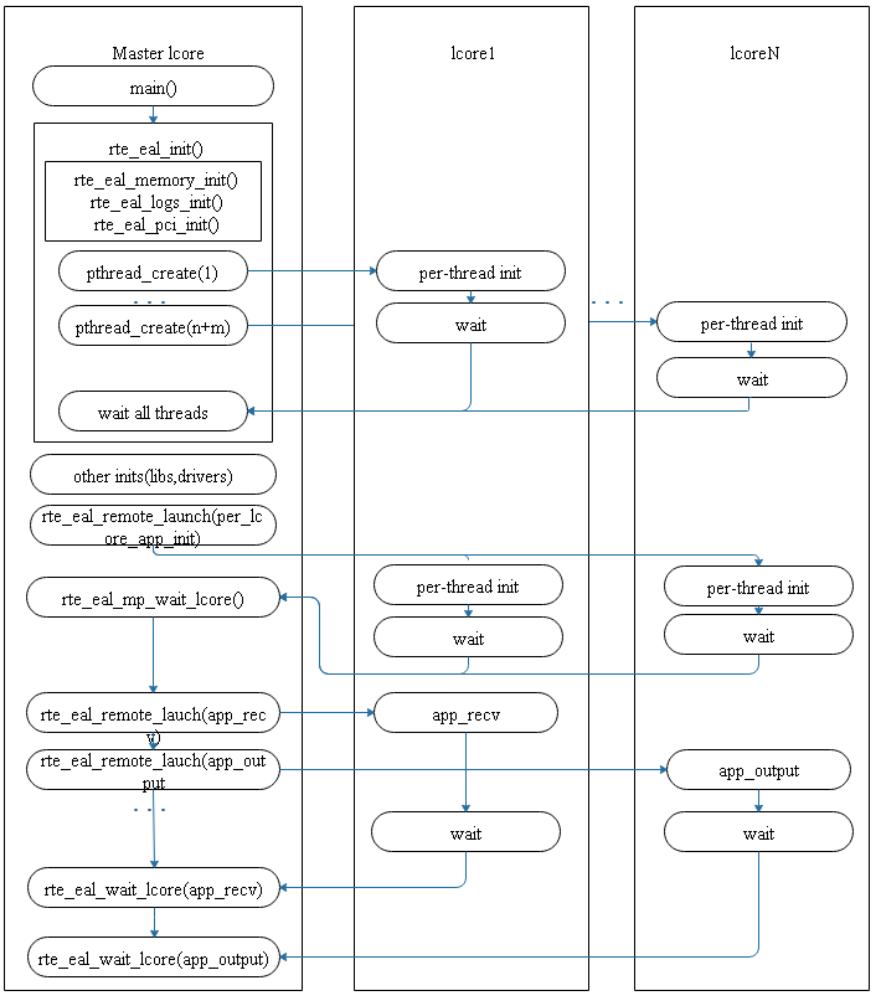


图 3.2 DPDK 环境下多线程的启动和管理模式

通过在通用服务器的多核架构上实现水平扩展的方法来提高数据采集系统的性能。创建的多个线程，每个线程都绑定在独立的核上，可以避免线程调度上下文切换的开销，提高系统的性能。一般将主控线程绑定在 MASTER 核上，解析系统的环境变量、命令行参数、进行数据初始化，将执行参数传递给网络线程和输出线程。网络线程和输出线程也各自独立绑定在预留的核心上。

执行过程解释如下，首先由 DPDK 建立的多核线程的基础运行环境，以 main 函数为运行入口，由线程掩码参数指定了参与整个程序运行的逻辑核集合。并在 MASTER 核上运行主控线程，由主控线程调用 rte_eal_init()进行环境初始化,包括运行内存初始化、日志环境初始化、PCI 设备初始化。

然后启动多核运行环境，遍历所有可以使用的逻辑核心，在每个核心上创建线程，并完成每个线程内部的初始化工作，再通过 rte_eal_remote_launch()让每个

线程调用各自的例程入口，启动网络线程和输出线程。

3.2.2 关键技术

3.2.2.1 多线程并行优化

由于在超高 IO 流量的数据采集下，网络线程的负载极高，让线程在特定的 CPU 长时间运行，避免调度到其他处理器上，可以有效减少上下文执行环境切换造成的开销和处理延迟，这对实时处理网络数据流非常重要。数据采集系统的主控线程在启动时，从启动参数中获得调度策略，将网络线程和归档线程指定分配到单独的 CPU 核心上运行，这样利用了 CPU 和线程的亲和性管理，避免线程在 CPU 核心上迁移的开销。

此外，主控在为网络线程和归档线程选择 CPU 核心时，依据所在服务器的非一致性内存访问（NUMA）多处理器架构的拓扑结构进行选择。这是因为在 NUMA 架构下，每个处理器都有自己本地的资源，虽然这些资源是全局可访问，但是跨处理器之间的访问开销要远远大于本地访问开销。所以，主控线程不仅将网络线程绑定在接收数据的网络适配器所在的 NUMA 节点的核心上，而且也在同一个 NUMA 节点上绑定归档线程，并使用该节点的内存资源创建和管理内存池和环形缓冲。

3.2.2.2 采用轮询模式的数据包处理

传统的中断模式下，网络适配器每接收到一个帧都会产生一个中断来通知 CPU，CPU 切换上下文后进行处理，这种方式只适用于异步的、数据量少的通讯。按照每个以太网帧 9000 字节计算，在 10Gb/s 的速度下，每秒的中断次数为 110 万次。虽然可以通过 ethtool 工具配置网络适配器工作在 interrupt coalescing 模式下，累计多个数据帧再进行一次中断处理来提高吞吐量，缺点是增加了延迟。采用 DPDK 的轮询模式从网络适配器获取数据包，屏蔽了硬件发出的中断，可以最大限度的提供性能，每一个逻辑核可以分配一个发送和接收队列，将收到的数据包平均放在网络适配器的接收队列中，以此实现负载均衡。

3.2.2.3 用户态的协议栈实现

由于网络线程使用轮询模式接收数据，直接访问网络适配器的接收队列进行处理，绕开了内核对数据包的处理，提高了处理速度，但 DPDK 本身主要面向底层数据流的高速转发，却不提供网络协议栈的支持，只能工作在 OSI 七层模型的第二层上，没有提供第三层以上的网络协议支持。所以，对于需要使用 UDP/IP 协议的应用程序，为了和数据源正常通信，系统至少需要支持 ARP 协议、IP 协议、ICMP 协议、UDP 协议的部分子集，出于最大化提升数据采集效率的考虑，系统并未采用 BSD 协议栈移植。另外其他的开源用户态网络协议栈，均有着各自特定的应用场景，但没有 Linux 内核中提供的协议栈稳定。本论文针对射电天文目前广泛采用的 UDP 协议数据流的特点，直接在网络线程中实现高效的用户态的 UDP/IP 协议栈支持。如图 3.3 所示。

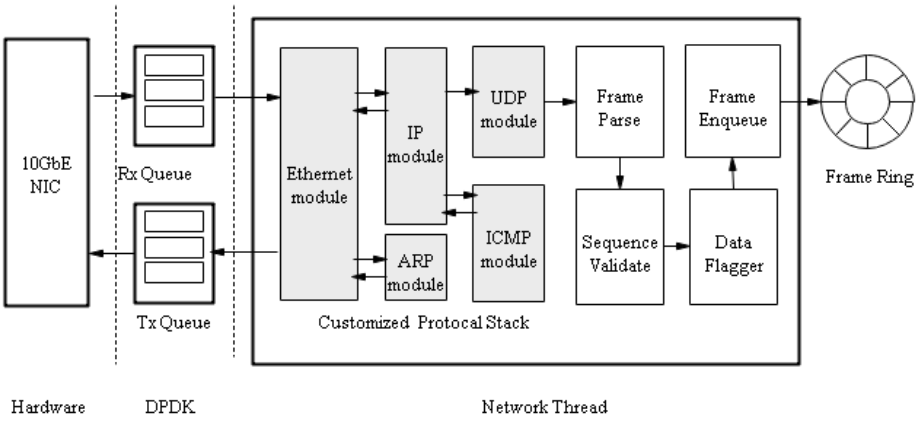


图 3.3 用户态的协议栈示意图

主控线程在启动网络线程时需要为工作的万兆网络适配器配置 IP 地址、子网掩码等参数，协议栈根据这些网络参数进行通讯。下面对支持的协议以及相关协议操作的实现进行描述。内存管理和 Ring buffer

数据采集系统系统使用了 mempool 和 ring buffer 来管理多个线程之间的通讯。mempool 用于存放预先分配的固定数目的内存缓冲区，网络线程从 mempool 获取空闲缓冲区，当缓冲区对象写满后，将其放入 ring buffer，再从 mempool 中获取新的空闲缓冲区。而归档线程从 ring buffer 中获取缓冲区对象，处理完毕后将缓冲区对象放回 mempool 中。使用 mempool 节约了反复申请和释放内存的开

销，而 ringbuffer 做为无锁环形缓冲区队列保证了各个线程之间的通讯。此外，在采用的巨页方式的内存管理降低了页表项的大小，减少了 TLB-Miss，提高了缓冲区的访问效率。

3.3　云台 40 米脉冲星实时消色散数据采集

云南天文台 40 米射电望远镜位于云南昆明，于 2006 年 5 月投入运行，脉冲星射电观测一直是其重要的一个科学研究内容。望远镜现配备有 S、C、X 波段接收机，均为双极化，S 波段为常温接收机，设计带宽为 2150Mhz-2450Mhz，C 波段为制冷接收机，设计带宽为 4000Mhz-8000Mhz，观测带宽为 70MHz-1024Mhz。目前使用从澳大利亚 ATNF 引进的 PDFB4 系统，采用非相干消色散模式在 S 波段进行日常脉冲星观测研究。射电天文研究组目前开展了脉冲星相干消色散观测系统构建的相关研究，目前已有了基于 CASPER 的 ROACH2 平台和开源软件离线处理结合的前期工作[34]。

云台 40 米目前新构建的相干消色散观测系统，主要硬件由天线、接收机、和数据处理设备构成，其硬件结构如图 3.4 所示。整体架构采用 CASPER 的 FGPA 与 GPU 的通用架构进行设计。FPGA 选用 ROACH2 平台，ROACH2 平台提供 FFT、DSP、多相滤波器设计等信号应用库，配有基于 PowerPC 处理器的嵌入式操作系统，并且为用户提供交互式控制接口。硬件性能指标为 2 路模数转换器模块 EV8AQ160(最高采样速率为 5Gbps)、2 路千兆网、8 路 SFP+万兆网接口和用于程序调试的 JTAG 接口、串口和 IIC 接口。进行输入带宽 512MHz、128 通道、8bit 采样和双极化输入(1024MHz)的多相滤波器基带数据采集。数据采集终端的数据输出速率为 16Gbps,考虑到后端数据传输和计算需求,通过 4 路 SFP+万兆网口输出,每路 SFP+的数据率为 4Gbps。4 路 SFP+万兆网口对应的 IF 信号分别为 1MHz~128MHz、129MHz~256MHz、257MHz~384MHz 和 385MHz~512MHz,每路 SFP+输出 32 通道。

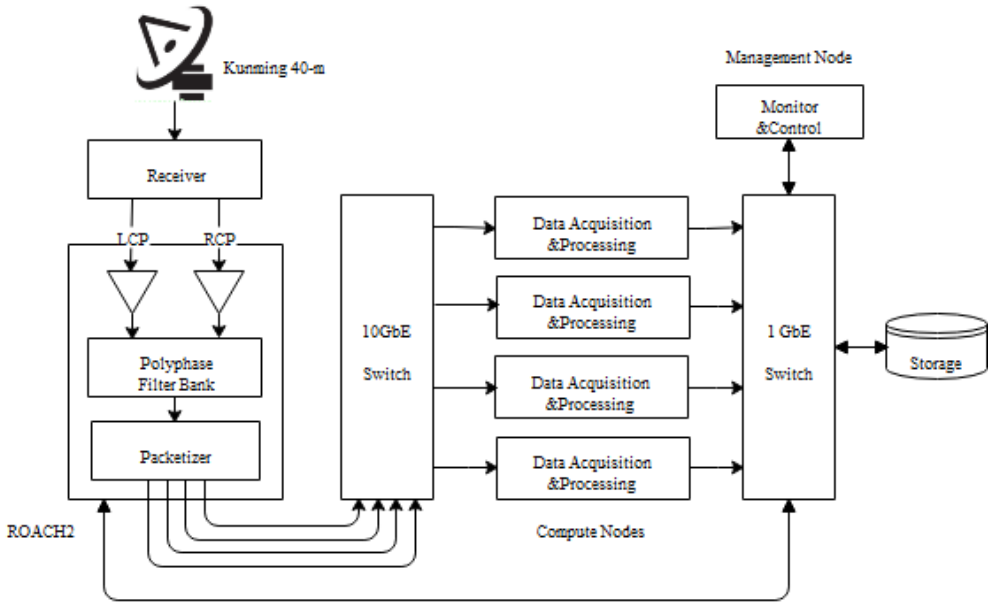


图 3.4 云台 40 米脉冲星相干消色散示意图

后端高性能计算节点配置由双路至强处理器，两块 NVIDIA Titan X GPU 图形计算适配器，操作系统为 CentOS Linux 6.8，CUDA 环境为 8.0。256GB DDR3 内存。GPU 和网络适配器通过 PCIE3.0 lanes 连接。由数据采集和处理程序实现数据的采集、解码、消色散、分通道、偏振计算和折叠以及 PSRFITS 格式输出等。

使用万兆以太网做为底层互联技术，必须充分考虑到硬件底层、系统架构和操作系统的细节，最大的优化数据的传输和处理效率，才能允许在更高的带宽动态范围和精度上进行观测，这对在 CPU/GPU 集群上运行的数据采集和数据处理软件的设计提出了更高的要求。高性能的数据采集是宽带脉冲星数据实时处理流水线的第一个重要环节。当前迫切需要解决在通用软硬件环境下（x86 架构、Linux/Unix）以及高 IO 网络环境下(10GbE/40GbE)的数据采集技术，能够充分利用网络传输能力，有效降低系统资源占用，便于后继的数据存储、调度分发和并行处理。此外，随着工业标准的 10Gbe、40Gbe 以太网络在射电天文后端系统作为基础互联架构的广泛使用，解决高 IO 网络环境下的数据采集问题对许多射电天文后端系统的 Data Acquisition System 系统的设计和改进有着积极的借鉴意义。

3.3.1 脉冲星高速数据采集

云台 40 米脉冲星相干消色散观测系统的后端使用了四台高性能的服务器。数据采集系统系统分别在四台计算服务器上运行，分别捕获前端 ROACH2 发送

的观测数据流。前端发送的观测数据格式如图 3.5 所示。

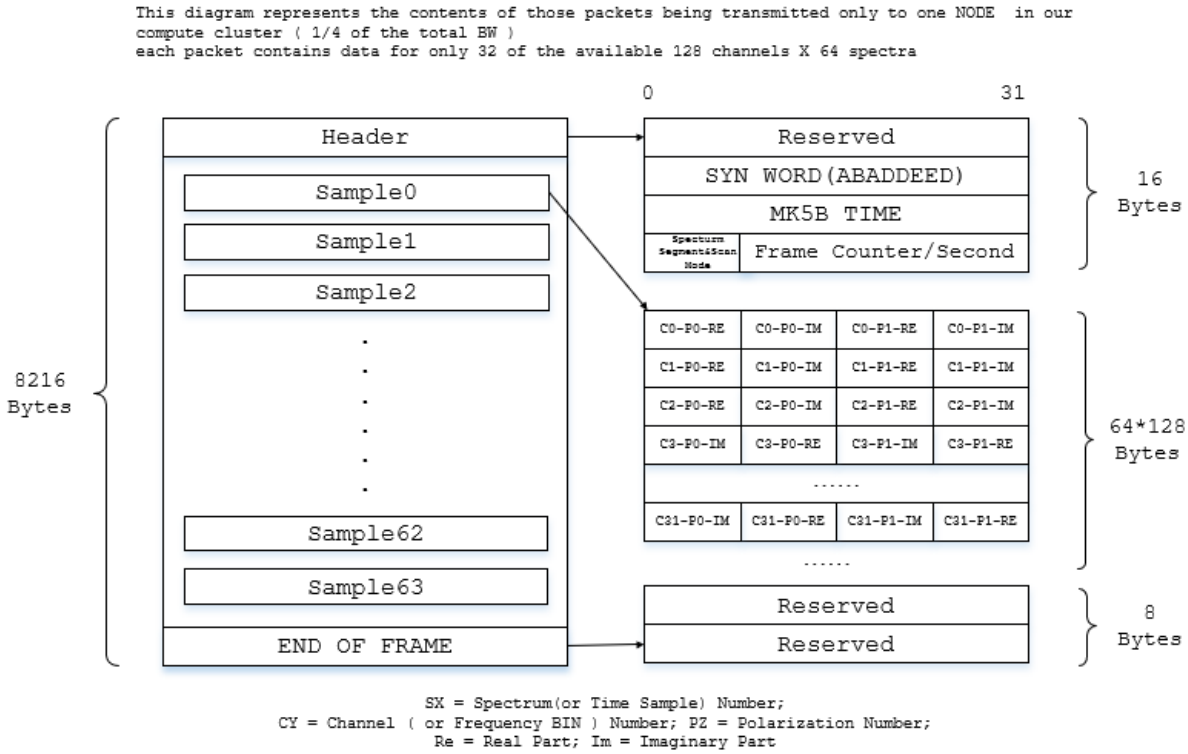


图 3.5 前端输出观测数据格式

每个数据帧由首部、数据和尾部构成。首部中包括保留字段、同步字段、通道分组编号、MK5B 格式的采样时间、秒内序号。秒内序号用于检测数据丢包和排序，前端每发送一个数据帧，秒内序号会增 1，为了避免溢出，每到整秒，秒内序号会归零重新计数。

数据部分包括 64 个时间点的复采样，每个采样包括 32 个通道和两种极化。尾部仅有保留字段。每个数据帧大小为 8216 字节。后端服务器的网络适配器以及交换设备均支持超长帧的传输，最大传输单元 MTU 设置为 9000 字节。

3.3.2　性能比较

我们对数据采集系统的性能进行了测试。我们同时也实现了基于套接字的数据采集系统，和基于 DPDK 的数据采集系统进行性能比较。

测试环境由数据发送服务器、数据接收服务器和万兆交换机构成。服务器硬件配置为双路 Intel E5-2630V3@2.40G 处理器，256GB 内存，Intel 82588ES 万兆网络适配器。服务器操作系统为 CentOS 7.4，内核版本为 3.10.0。交换机型号为 H3C S6300-42QT。

传统的基于 socket 的数据采集程序，我们在测试之前，对操作系统内核和网络参数做了常见的调优，以提高其性能，具体优化命令如下。

```
echo 1 > /proc/sys/net/ipv4/conf/all/arp_filter
echo 1 > /proc/sys/net/ipv4/tcp_tw_recycle
echo 10 > /proc/sys/net/ipv4/tcp_fin_timeout
echo 16777216 > /proc/sys/net/core/wmem_max
echo 16777216 > /proc/sys/net/core/rmem_max
echo "4096 87380 16777216" > /proc/sys/net/ipv4/tcp_rmem
echo "4096 87380 16777216" > /proc/sys/net/ipv4/tcp_wmem
echo 0 > /proc/sys/net/ipv4/tcp_sack
echo 1 > /proc/sys/net/ipv4/tcp_no_metrics_save
echo 3000 > /proc/sys/net/core/netdev_max_backlog
```

鉴于在实际观测中，普遍使用超长帧来传输 UDP 报文以提高传输效率，我们选用了长度为 9K 字节的数据帧进行测试。然后分别测试在 1Gb/s 至 10G/s 的不同数据速率下接收数据帧的丢包率，测试时长为 60 秒。实验结果如表 3.1 所示。

表 3.1 丢包率(%)比较

数据速率 Gb/s	1	2	3	4	5	6	7	8	9	10
基于 socket 数据采集系统丢包率	0	0	0.04	0.2	0.3	0.4	0.5	0.8	0.9	1.2
基于 DPDK 数据采集系统丢包率	0	0	0	0	0	0	0	0	0	0

从测试结果可以看出，与基于 socket 的数据采集系统相比，基于 DPDK 的数据采集系统在高带宽下仍然可以做到 0%的丢包率，能够在万兆以太网上实现接近理论最大带宽 10Gb/s 下的无丢包数据传输。此外，我们也对基于 DPDK 的数据采集系统在 10Gb/s 的数据速率下的稳定性进行了测试，在 2 个小时持续的测试中，丢包数为 0，也表明基于 DPDK 的数据采集系统具有非常好的稳定性，能够满足脉冲星观测系统前端和后端高 IO 数据传输和采集的需求。

3.4 本章小结

本章中针对现代射电天文面临的数据传输的挑战，提出了一种新的解决方案。通过采用用户态空间技术的数据采集框架，可以极大的提高数据的采集性能。从云南天文台 40 米射电望远镜脉冲星相干消色散对数据采集的需求角度出发，对实现对天文数据实时采集的支持，尤其是满足超高 IO 环境下无丢失的 UDP 数据流。

第4章　基于 GPU 的高性能实时数据处理

4.1　脉冲星消色散

4.1.1　云台 40 米脉冲星实时消色散数据处理

相干消色散的主要数据处理流程为对前端输出的基带数据进行解码和数据重组，然后将数据变换到频域，乘以星际介质函数进行消色散处理，然后变换回时域，计算偏振，再根据预测的周期进行折叠，最后将折叠后的文件保存为标准 psrfits 格式进行归档。整个处理流程如图 4.1 所示。

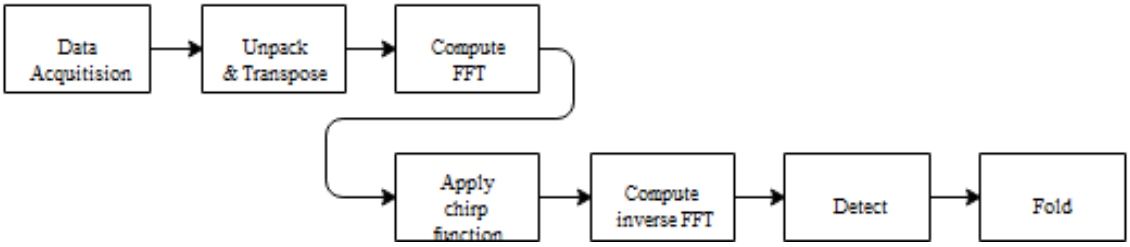


图 4.1 消色散处理流程图

在脉冲星消色散的数据处理流程中，涉及了实时数据采集、实时数据处理、基于 GPU/CPU 的加速、分布式调度和数据归档等功能。虽然脉冲星相干消色散的算法和原理已经较为成熟，但构建一个可用的海量数据的实时采集和处理平台依然面临着具体的困难，包括以下几个方面：

1）海量实时观测数据

云台脉冲星观测前端输出的数据量计算如下：ROACH2 平台输入左右旋各为 512M，总带宽 1024M，采样频率是 2048M,进行 8bit 采样，速率为 16Gbps，再经基带转换后（增加了虚部）为 32Gbps，由于通道的对称性，取一半数据，所以前端输出的基带数据速率为 16Gbps，目前是分四路输出，每路 4Gbps，目前由 4 个服务器分别接收处理。

2）基于 CPU/GPU 异构集群的海量数据实时处理

脉冲星的宽带观测带来海量的数据处理需求，对海量的观测数据进行实时相干消色散带来海量的实时计算需求。目前海量的观测数据不适用于大规模离线保存和事后处理，主要采用的是实时处理模式。与非相干消色散相比，实时相干消色散对计算能力的要求更高，其主要处理步骤包括解码、傅里叶变换、

消色散、分通道、逆傅里叶变换、偏振计算、折叠等，由于多数脉冲星的信号非常微弱，为了提高信噪比获得稳定的平均轮廓还必须保证足够的观测时间进行积分。当前迫切需要解决在 CPU/GPU 异构集群计算环境下，实现并优化相干消色散算法，研究算法在不同延迟的内存层级、不同规模的计算能力的自适应和负载均衡，满足目前的 16Gb/s 的实时数据处理性能要求。

目前国外最常见的脉冲星观测数据处理的开源软件有 DSPSR 和 SIGPROC，其主要功能都包含了消色散的处理。这些软件可以使天文学家专注于数据，而无需陷入复杂的算法的编程工作。

SIGPROC 由 Duncan Lorimer 开发[35]，基于 C 语言开发，包括多个独立运行的数据处理程序，用于处理不同阶段的不同格式的数据文件。该程序已经在 WAPP (Wideband Arecibo Pulsar Processor)和 PSPM(the Penn State Pulsar Machine)等观测设备使用。消色散的程序 dedisperse 目前只支持非相干消色散，而且没有提供基于 GPU 的实现。

DSPSR（Digital Signal Processing for Pulsars）是另外一个著名的、广泛使用的脉冲星数据处理软件。比 SIGPROC 开发的时间更晚，但功能更为强大。DSPSR 基于 C++语言开发，支持更多的数据格式，支持相干消色散、滤波处理[36]。Swinburne 技术大学的脉冲星设备均在使用，包括 APSR（ATNF Parkes Swinburne Recorder）实时相干消色散系统、BPSR（Berkeley Parkes Swinburne Recorder）Parkes 多波束接收机的数据采集和处理系统、CASPSR（CASPER Parkes Swinburne Recorder）[37]。

在射电天文数据处理研究的近几十年中，基于框架来构建射电天文数据处理 pipeline，科研工作者们已经做了众多的尝试。

如牛津大学电子研究中心开发的 PELICAN（Pipeline for Extensible Lightweight Imaging and CAlibratioN）是一个高性能、轻量级的 pipeline 框架，用于流式天文数据应用的开发[38]。LOFAR 和 Medicina 射电望远镜都使用此框架开发过数据处理管道应用程序。SKA 早期也计划用此框架处理非成图的基于 GPU 的实时射电脉冲检测如脉冲星消色散搜索[39, 40]。其缺点是虽然提供了高级抽象的 API，但是只适合用于静态的准实时的数据处理。NIST 的 HTGS，基于 C++开发，通过在图中定义节点，将节点绑定到 CPU 线程，以支持在 CPU/GPU 上混合计

算[41]。

PSRDADA 开源软件项目，主要是用于脉冲星天文观测中的基带数据记录和处理[42]。PSRDADA 使用多线程和进程模式管理数据传输、控制指令、数据处理，基于共享内存的 ringbuffer 和 internet 的传输协议。PSRDADA 由澳大利亚 Swinburne 大学的脉冲星的研究组负责维护开发，已经被用于 APSR（ATNF Parkes Swinburne Recorder）实时相干消色散系统、BPSR（Berkeley Parkes Swinburne Recorder）后端的数据采集和处理、Swinburne 大学的脉冲星网格 Supsr Grid[37]。PSRDADA 也被 LEDA（Large Aperture Experiment to Detect the Dark Ages）采用，作为 X-Engine 的数据处理管道的框架[43]。

CHIME Pathfinder 射电干涉望远镜使用混合架构 FPGA/GPU 来构建了 correlator，F-Engine 和 X-engine 通过 10GbE 互连。Andre Recnik 使用 C 语言开发的软件框架 kotekan 上实现了用于管理 X-engine 实时的数据流的 pipeline，包括接收来自 F-engine 的 UDP 数据包，GPU 实时相关、处理结果的存储[44]。

VEGAS（Versatile Green Bank Spectrometer）的 HASHPIPE 基于 C 语言开发，用于构建 VEGAS 高性能集群的 pipeline，实现了内存环形缓冲管理，共享内存段、信号量等系统功能管理，应用程序作为 HASHPIPE 的共享库插件的形式进行开发，可构建多线程的 pipeline[45]。

Bitfrost 使用 ring buffers 将 task 链接在一起，构成实时数字信号处理的管道。其设计理念和 CASPER 的 FPGA 的工具流,GNU 的 GnuRadio 非常相似。

上述这些框架，PELICAN 适用于准实时的数据处理，PSRDADA 和 HASHPIPE 不够灵活，只提供低层的 C 的 API 调用。HTGS 也只是在 GB 刚刚开始原型构建。国内，目前对类似框架的研究只是刚刚起步。

4.2 基于 CPU/GPU 的实时消色散

主要功能包括：支持合成滤波器的相干消色散；支持双极化观测数据的偏振计算；支持使用常数或者使用脉冲星测时软件预报的模型进行折叠；可定义子积分长度；支持多个 GPU/CPU 的并行计算；使用高速环形缓冲与数据采集系统无缝集成。

4.2.1 管线设计

如上文图中所示，由于进行相干消色散涉及到大量的浮点运算，如果要实时的进行数据处理，必须要对相干消色散的数据进行并行处理。目前的并行在两个级别进行，第一级是前端数据预处理时，将观测数据通过不同的 UDP 数据流分发到各个计算节点上的并行，并行的粒度取决于 FPGA 的数量以及 FPGA 使用多相滤波器划分通道分组的数量。第二级的并行是每个计算节点上的并行，为了充分利用计算节点的计算能力，在实时观测模式下，计算节点上默认使用所有的 GPU 设备和指定数量的 CPU 核心同时进行计算。如图 4.2 所示：

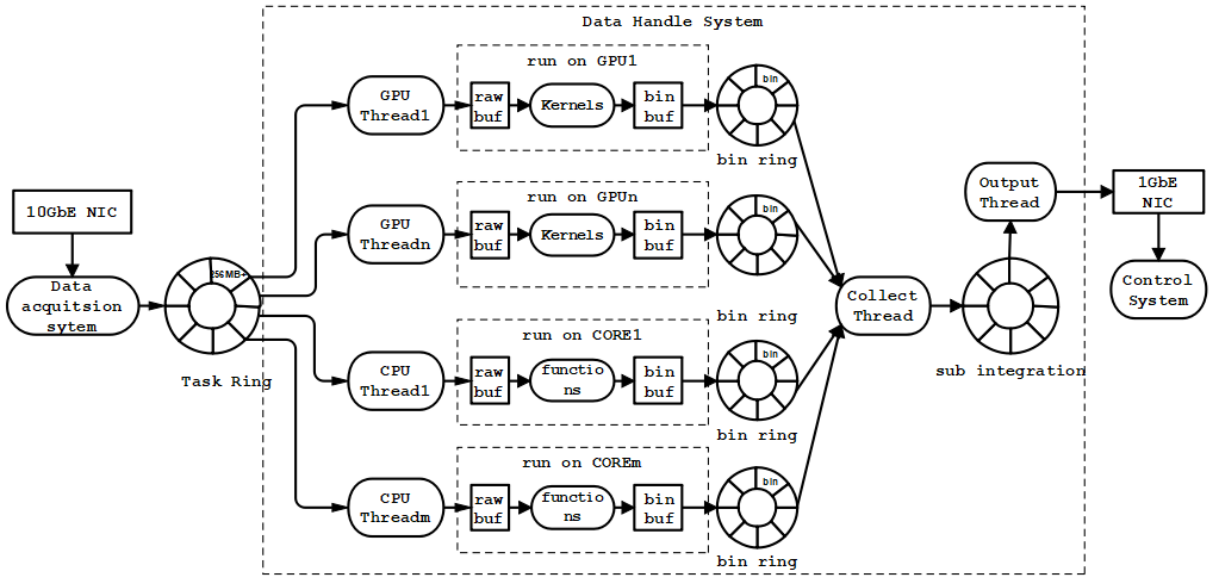


图 4.2 单台计算节点并行处理

每个计算节点的数据处理程序包括多个处理线程，1 个归集线程，1 个输出线程。数据处理系统的线程由数据采集系统的主控线程管理。并行调度的核心组件是环形缓冲。主控线程根据 GPU 卡的数量以及可用 CPU 核心的数量创建处理线程。数据采集系统将采集的待处理数据放入环形缓冲(task ring)中，每个缓冲区都是一个自定义的 Task 对象，Task 对象中保存观测数据和观测参数，由多个处理线程从环形缓冲取出数据进行消色散处理，最后将折叠结果放入每个线程单独的环形缓冲（bin ring）中，由归集线程按照时间先后顺序，按照子积分时间长度进行归集合并，并将子积分放入子积分环形缓冲中(subint ring)，由输出线程通过普通的网络适配器通过 TCP 连接发送给主控线程进行合并后输出。

为了充分利用每个计算节点的计算能力，保证消色散的实时处理，实时并行处理系统采用了以下的设计和优化：

4.2.1.1　处理线程的并行优化

主控线程在创建 GPU 和 CPU 处理线程时，依据 CPU 的亲和性，将线程分配到独立的 CPU 核心上运行，主控线程不会创建超过计算节点物理核心数量的处理线程。在为 CPU 创建处理线程时，优先绑定数据采集系统的网络线程所在 NUMA 节点，也就是万兆网络适配器所在 NUMA 节点上的核心。在为 GPU 的创建处理线程时，则依据 NUMA 节点的拓扑，将优先绑定 GPU 所在 NUMA 节点上的 CPU 核心。在 GPU 处理线程和网络线程不在同一 NUMA 节点的情况下，GPU 处理线程对 task ring 环形缓冲的访问将通过高速总线(Intel QPI 或者 AMD HT)。

4.2.1.2　重叠数据的预处理

在相干消色散的处理流程中，对于每一块要处理的采样数据，将会丢弃n_{DM}个采样点。为了避免丢弃的采样点对处理结果的影响，每一块要处理的采样数据，与其前后的数据块，都有n_{DM}个采样数据的重叠。由于使用了多个处理线程，多个处理线程从 task ring 中获取 task 的顺序取决于各个处理线程的处理速度，具有随机性难以预测。所以在数据采集系统的网络线程中，将依据主控线程计算的n_{DM}值，在连续的两个 task 结构中，有n_{DM}的采样点是重叠。

相干消色散的主要处理步骤均在 GPU 上实现。为了提高 GPU 的计算效率，首先应该尽可能减少主机内存和 GPU 的设备内存之间的传输的数据量和调用次数，通过较大的数据缓冲，获得更大的带宽。系统实现了基于 GPU 的解码和数据重组，消色散，综合滤波，偏振计算以及折叠的全部过程。以 32 通道双极化为例，1GB 的 task，传输至 GPU 上经过解码后数据大小为 4GB，经过消色散后按照 1024 个 bin 进行折叠后的轮廓数据为 524KB。

为了最大化 GPU 的处理能力，对算法的各个步骤都进行了调优。使用 DPDK 分配的巨页内存可以直接被 GPU 的锁页内存的异步处理函数调用，系统将算法的各个步骤使用 cuda 的流进行管理。此外，由于相干消色散算法中涉及了大量的 FFT 和逆 FFT 运算，在使用 NIVIDIA CUDA 的 CUFFT 库时，使用批模式对

数据进行傅里叶运算进行加速。

4.2.2 解码

数据采集系统从前端系统接收的数据是按照时间、频率和极化排列的多维采样数据。需要进行解码和数据重组的预处理。解码是将 nbit 采样的基带采样的量化数据映射为对应的浮点数据。数据重组是为了便于后继的相干消色散处理，将数据按照频率和极化和时间的维度重新排列，保证每个频率和极化都是连续的时间序列。

4.2.3 消色散

对解码和重组后的每个通道和极化的连续的基带采样数据，进行相干消色散处理。算法主要基于 Lorimer 和 Kramer 以及 Bhattacharya 提出的方案。如上文所述，相干消色散主要是将以奈奎斯特速率复采样的数据通过星际介质的反向传输函数的滤波器。进行消色散处理的信号持续的时间必须大于高频和低频信号之间的延迟时间。对于离散采样的数据，相干消色散需要的最少的采样数和色散延迟相关。故$n_{DM} = t_{DM}\Delta f$。此外，n 个采样信号的 FFT 变换依赖于前后的 n/2 采样点。因此，采样数据必须在前后额外使用$n_{DM}/2$个采样点填充。所以，傅里叶变换的最少的长度为$2n_{DM}$。

对基带采样数据进行相干消色散的主要算法流程如下：

1）读取 n 个基带数据采样；为了提高计算效率，n 通常远远大于$2n_{DM}$；

2）计算 n 个数据点的星际介质函数$Chirp$；

3）对 1）中的 n 个采样傅里叶变换到频域，并且乘以步骤 2）计算的星际介质函数消除色散；

4）对 3）的计算结果进行反傅里叶变换到时域；5）对 4）获得时间序列数据，去掉其头尾$n_{DM}/2$的数据点，保留数据中间$n - n_{DM}$的数据点作为输出；

6）保留 1）序列中的最后 nDM 数据点，并后续时间的基带采样数据中获得$n - n_{DM}$个数据点，组成新的n_{DM}个数据点的时间序列，重复 3）开始的数据处理步骤。

4.2.4　分通道

通过滤波器组将每个通道划分为若干子通道，可以提高频率分辨率。既有利于偏振校准，也有利于后期的窄带的射频干扰的移除。对基带采样数据进行合成滤波与消色散的计算可以一起完成，主要算法流程如下：

1）读取K个基带数据采样点序列；$K = N_c \times N'$，N_c是划分子通道的数量；N'是每个子通道傅里叶变换的长度；

2）对 1）中的K个采样傅里叶变换到频域，将变换结果划分为N_c个通道，每个子通道的点数为N'

3）对 2）中的每个子通道进行如下计算：

a)乘以子通道的星际介质函数；

b)执行逆傅里叶变换到时域；

c)对于 b 获得的时间序列数据，丢弃头尾的n'_d的数据点，保留数据中间的$N' - n'_d$进行输出；

4）保留 1）序列中的最后$N_c \times n'_d$数据点，并后续时间的基带采样数据中获得$K - N_c \times n'_d$个数据点，组成新的 K 个数据点的时间序列，重复 2）开始的数据处理步骤。

4.2.5　偏振检测

使用两个正交的极化观测时，可以使用消色散后的两个正交极化的复采样数据来计算信号的偏振。

线偏振使用下面的公式通过X和Y极化来计算斯托克斯四个参数I，Q，U，V。

$$\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{Q} \\ \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} |\boldsymbol{X}|^2 + |\boldsymbol{Y}|^2 \\ |\boldsymbol{X}|^2 - |\boldsymbol{Y}|^2 \\ \boldsymbol{2Re(X * Y)} \\ \boldsymbol{2Im(X * Y)} \end{bmatrix} \tag{4.1}$$

圆偏振使用下面的公式通过左旋极化L和右旋极化R来计算斯托克斯四个参数I，Q，U，V。

$$\begin{bmatrix} \boldsymbol{I} \\ \boldsymbol{Q} \\ \boldsymbol{U} \\ \boldsymbol{V} \end{bmatrix} = \begin{bmatrix} |\boldsymbol{L}|^2 + |\boldsymbol{R}|^2 \\ 2\boldsymbol{Re}(\boldsymbol{L} * \boldsymbol{R}) \\ 2\boldsymbol{Im}(\boldsymbol{L} * \boldsymbol{R}) \\ |\boldsymbol{R}|^2 - |\boldsymbol{L}|^2 \end{bmatrix} \tag{4.2}$$

4.2.6 折叠

脉冲星通常是非常微弱的射电源。需要对消色散后的时域信号按照周期进行积分，既折叠，提高信噪比，才能观察到清晰的脉冲轮廓。消色散后的时间序列，采样间隔为t_{samp};将脉冲星积分轮廓均分为 n 个部分，每个部分称为一个 bin。每个 bin 对应着脉冲的特定的相位 phase，第 i 个 bin 对应的相位中心为$(i - 0.5)/n_{bins}$;计算每个采样所对应的相位;计算该相位所对应的 bin，将该采样累计到 bin 中。目前系统使用脉冲星测时软件 TEMPO2，依据观测目标和望远镜的地理位置坐标来生成一个多项式来近似脉冲星的时间模型$P_{\emptyset}(t)$，将每个信号映射到对应的 bin 中进行累加。

4.2.7 psrfits 文件输出

折叠后的平均脉冲轮廓的多通道全极化的数据保存为 PSRFITS 格式。PSRFITS 是脉冲星数据文件的标准格式。PSRFITS 格式从广为使用的标准 FITS 格式扩展而来。观测望远镜的标识和坐标，接收参数，观测时间等信息保存在 HDU 中。各个子积分的积分数据保持在扩展 HDU 中。

4.2.8 pipeline 设计

实时数据采集和实时处理系统通过环形缓冲连接，在每个计算节点上的流水线设计示意图如图 4.3 所示。

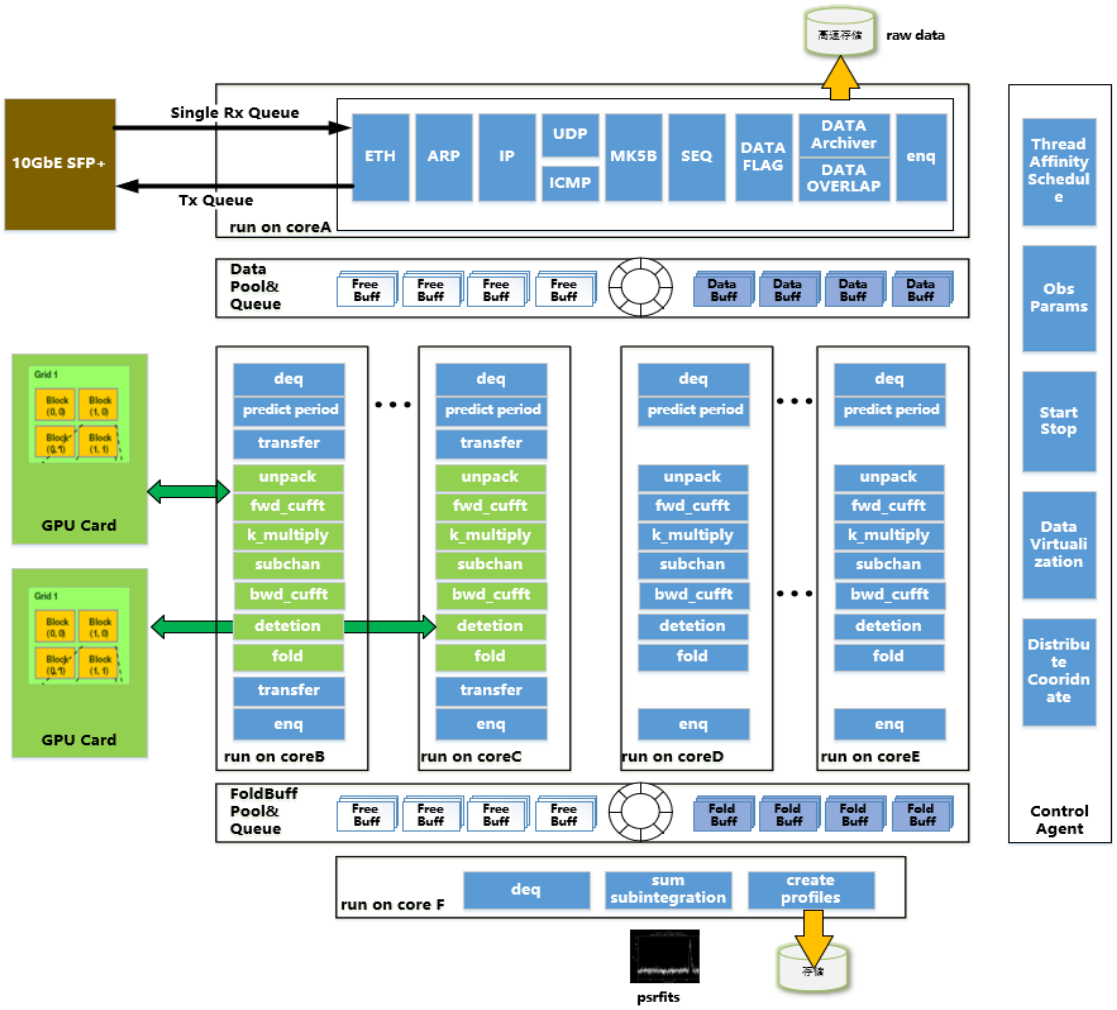


图 4.3 计算节点 pipeline 示意图

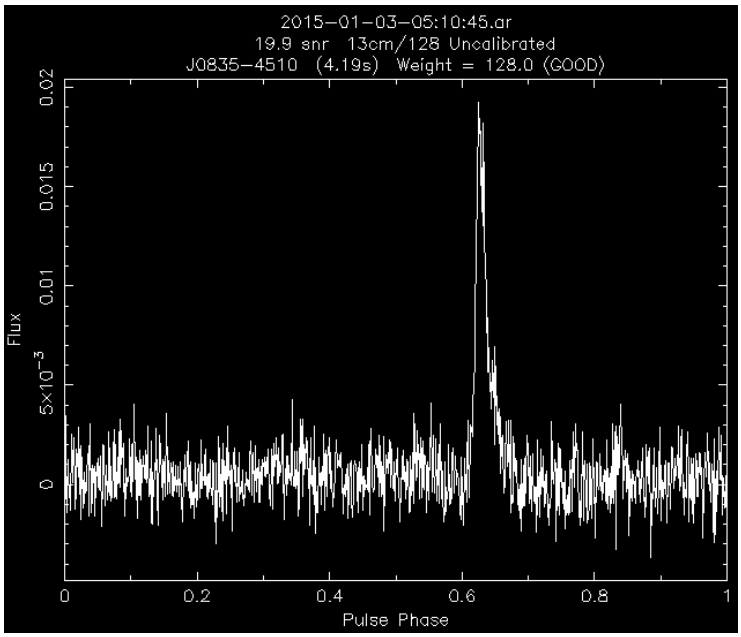
4.2.9　离线文件测试

在单线程 CPU 调度单 NVIDIA GeForce GTX TITAN X 的 GPU 卡上使用 8.2 秒的时间处理了 2GBytes 大小数据文件，共 163840000 个采样(等同于 4 秒钟的实时数据)，程序运行结果如图 4.44 所示。

```
[root@hb72 SingleThread-1.0]# time ./ynpulsar -u -f /root/dspsrdata/0835_14b.data
/root/dspsrdata/0835_14b.data has 16384000 samples
Detected 2 CUDA Capable device(s)
dsp::Shape::resize  npol=2  nchan=128  ndat=512  ndim=1
dsp::Dedispersion::build
  centre frequency = 2219
  bandwidth = 128
  dispersion measure = 68
  Doppler shift = 1
  ndat = 512
  nchan = 128
  centred on DC = 0
  fractional delay compensation = 0
Turn 1 load 8387572 samples(1073609216bytes) in 0.327203 seconds
Turn 1 gpu unpack 8387572 samples(1073609216bytes) in 0.402148 seconds
Turn 1 gpu filterbank 8387572 samples(1073609216bytes) in 1847.71 ms
Turn 1 gpu detection 8387572 samples(1073609216bytes) in 33.2965 ms
Turn 1 fold setbin 2096830 samples( in 0.017221 seconds
Turn 2 load 7996428 samples(1023542784bytes) in 0.292506 seconds
Turn 2 gpu unpack 7996680 samples(1023575040bytes) in 0.614049 seconds
Turn 2 gpu filterbank 7996680 samples(1023575040bytes) in 1626.74 ms
Turn 2 gpu detection 7996680 samples(1023575040bytes) in 31.6584 ms
Turn 2 fold setbin 1999107 samples in 0.015632 seconds
real    0m8.297s
user    0m3.679s
sys     0m3.604s
```

图 4.4 单 GPU 卡处理 4 秒钟观测数据

数据的最终处理结果如图 4.55 所示。使用 psrchive 的 pazi 命令查看，可以清晰地看到 J0835-4510 的轮廓图。



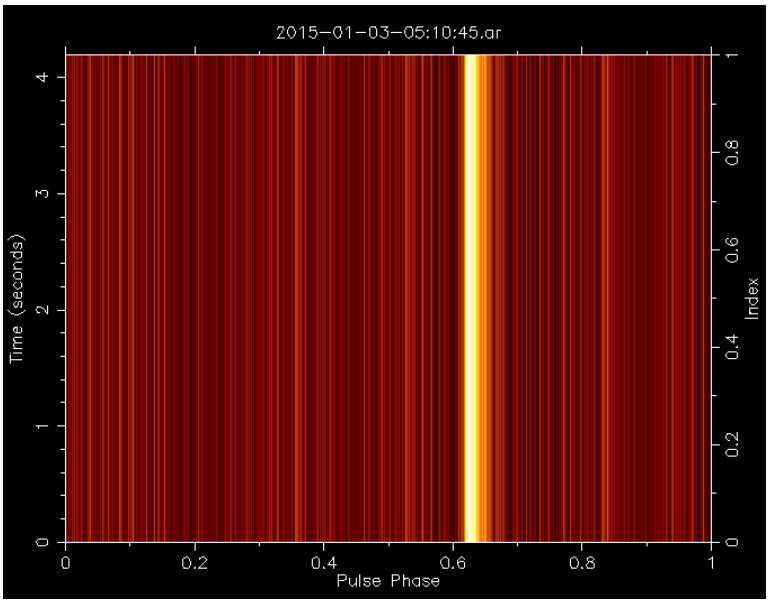


图 4.5 J0835-4510 的时域和频域轮廓图

4.3 性能分析与比较

在数据接收服务器上配置了 NVIDIA 的 K80 图像处理单元。K80 图像处理单元由两颗 GK210 组成，共 24G 显存，4992CUDA 核心。由于在进行消色散处理时选择傅里叶变换点数以及变换的重叠区域长度均不相同，所以需要的计算量也不同，我们选择了 6 颗不同色散值的脉冲星的离线数据做为测试数据。另外，为了测试数据处理系统在多个图像处理单元环境下的并行能力，分别测试了使用 1 个 GK210 和 2 个 GK210 情况下的性能。6 颗脉冲星的离线数据时长均为 60 秒，每秒4×10^6个复采样，每个采样包括 32 个通道 2 种极化，采样长度为 8 比特。实验结果如表 4.1 所示。

表 4.1 消色散处理耗时比较

脉冲星源	色散值（$cm^{-3} \cdot pc$）	傅里叶变换次数	傅里叶变换点数	处理时间（秒）	
				1*GK210	2*GK210
1136+1151	4.864	5000000	64	86	42
0332+5434	26.833	566038	512	90	45
1645-0317	35.727	265487	1024	93	46
0835-4510	67.990	131291	2048	96	48
0837-4135	147.290	66225	4096	97	49

1544-4559	478.800	16155	16384	113	57

从实验数据可以看出，在相同的数据量下，色散值越大，消色散处理耗时也越长。数据处理系统使用 2 个 GK210 进行计算时，所消耗的时间大约为 1 个 GK210 的 50%。所以，如果要满足实时消色散的计算需求，除了选用浮点运算能力更高的图像处理单元外，也可以通过扩展多个图像处理单元来并行实现。

4.4 试观测结果

系统部署于云南天文台 40 米射电望远镜，和现有的基于 ROACH2 前端进行了联合测试。前端将 128MHz 带宽、32 通道 2 种极化、8bit 复采样的数据，通过 1 路 SFP+万兆网口输出，数据输出速率为 4Gbps。

使用了 1 个计算节点，硬件配置为双路 Intel E5-2698V3@2.30G 处理器，256GB 内存，Intel 82588ES 万兆网络适配器，2 颗 NVIDIA 的 GeForce GTX Titan X 图像处理单元。Titan X 图像处理单元配有 12G 显存和 3072CUDA 核心。服务器操作系统为 Ubuntu 16.04.5 LTS，内核版本为 4.10.0。

在计算节点上查看了 NUMA 拓扑信息，确定了 GPU 以及万兆网卡所在的 NUMA 节点，并在执行脚本中进行了相应的优化。

NUMA 拓扑结构查看结果如图 4.6 所示。

```
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$ ./cpu_layout.py
============================================================
Core and Socket Information (as reported by '/proc/cpuinfo')
============================================================

cores =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
sockets =  [0, 1]

        Socket 0        Socket 1
        --------        --------
Core 0  [0, 32]         [16, 48]
Core 1  [1, 33]         [17, 49]
Core 2  [2, 34]         [18, 50]
Core 3  [3, 35]         [19, 51]
Core 4  [4, 36]         [20, 52]
Core 5  [5, 37]         [21, 53]
Core 6  [6, 38]         [22, 54]
Core 7  [7, 39]         [23, 55]
Core 8  [8, 40]         [24, 56]
Core 9  [9, 41]         [25, 57]
Core 10 [10, 42]        [26, 58]
Core 11 [11, 43]        [27, 59]
Core 12 [12, 44]        [28, 60]
Core 13 [13, 45]        [29, 61]
Core 14 [14, 46]        [30, 62]
Core 15 [15, 47]        [31, 63]
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$
```

图 4.6 计算节点的 numa 拓扑

万兆网卡所在 NUMA 节点可用下面的命令进行确定。

```
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$ lspci |grep -i eth
81:00.0 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
81:00.1 Ethernet controller: Intel Corporation I350 Gigabit Network Connection (rev 01)
82:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
82:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Network Connection (rev 01)
```

```
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$ cat /sys/devices/pci0000\:80/0000\:80\:02.0/numa_node
1
```

```
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$ lspci |grep -i nvidia
02:00.0 VGA compatible controller: NVIDIA Corporation GM200 [GeForce GTX TITAN X] (rev a1)
02:00.1 Audio device: NVIDIA Corporation Device 0fb0 (rev a1)
83:00.0 VGA compatible controller: NVIDIA Corporation GM200 [GeForce GTX TITAN X] (rev a1)
83:00.1 Audio device: NVIDIA Corporation Device 0fb0 (rev a1)
```

```
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$ cat /sys/devices/pci0000\:80/0000\:80\:03.0/numa_node
1
lzx@lzx-amax15:~/dw/dpdk-stable-16.11.2/tools$ cat /sys/devices/pci0000\:00/0000\:00\:02.0/numa_node
0
```

我们使用万兆网卡位于 numa 的 0 号节点，两个 GPU 卡分别位于 0 号节点和 1 号节点。我们将使用 0 号节点的核心来运行数据处理程序。以 J1136+1151 为例，其观测脚本如下。

```
lzx@lzx-amax15:~/dw/trunk201712/trunk/net/DPDKrecv/1136+1551$ more 1136.sh
#!/bin/bash
export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64/:/home/lzx/software/tempo2/lib/
export PATH=$PATH:/home/lzx/software/tempo2/bin
export TEMPO2=/home/lzx/software/tempo2
../build/DPDKrecv -l 32,33,34,35,36 -- --ulocalip=192.168.3.2 --uremoteip=192.168.3.16 --uport=8003 --second=180
0 --freq=2320 --length=10 --ephem=./1136.par --cuda=2 $*
```

脚本中我们使用了序号为 32，33，34，35，36 共 5 个 CPU 核心运行程序，运行包括主线程，网络接收线程，两个 GPU 处理线程和 1 个归集线程。由于内核旁路没有协议栈的支持，脚本中指定了数据源 IP(192.168.3.2)以及本地的工作 IP(192.168.3.16)和端口 8003，一共接收 1800 秒的数据，工作频率为 2320MHz，子积分时间为 10 秒，使用两个 cuda 设备，脉冲星的参数文件存在在 psrcat 命令生成的 1136.par 文件中。

我们选取了三颗不同的源进行观测，观测目标分别为 J1136+1551、J1932+1059、J2022+5154。J1136+1151 的观测时长为 30 分钟，J1932+1059 和 J2022+5154 的观测时长为 15 分钟，子积分时间均为 10 秒。

数据处理系统同时使用两个图像处理单元进行处理，调用 TEMPO2[46]进行

周期预报并按照 1024bin 进行折叠，最后将数据发送至管理节点输出为 PSRFITS 格式文件。J1136+1551 的输出文件的大小为 177MB，J1932+1059 和 J2022+5154 的输出文件大小为 88MB。管理节点生成的频域、时域和总的积分轮廓图，包括从 EPN(European Pular Network)检索的平均轮廓，如图 4.7 所示。

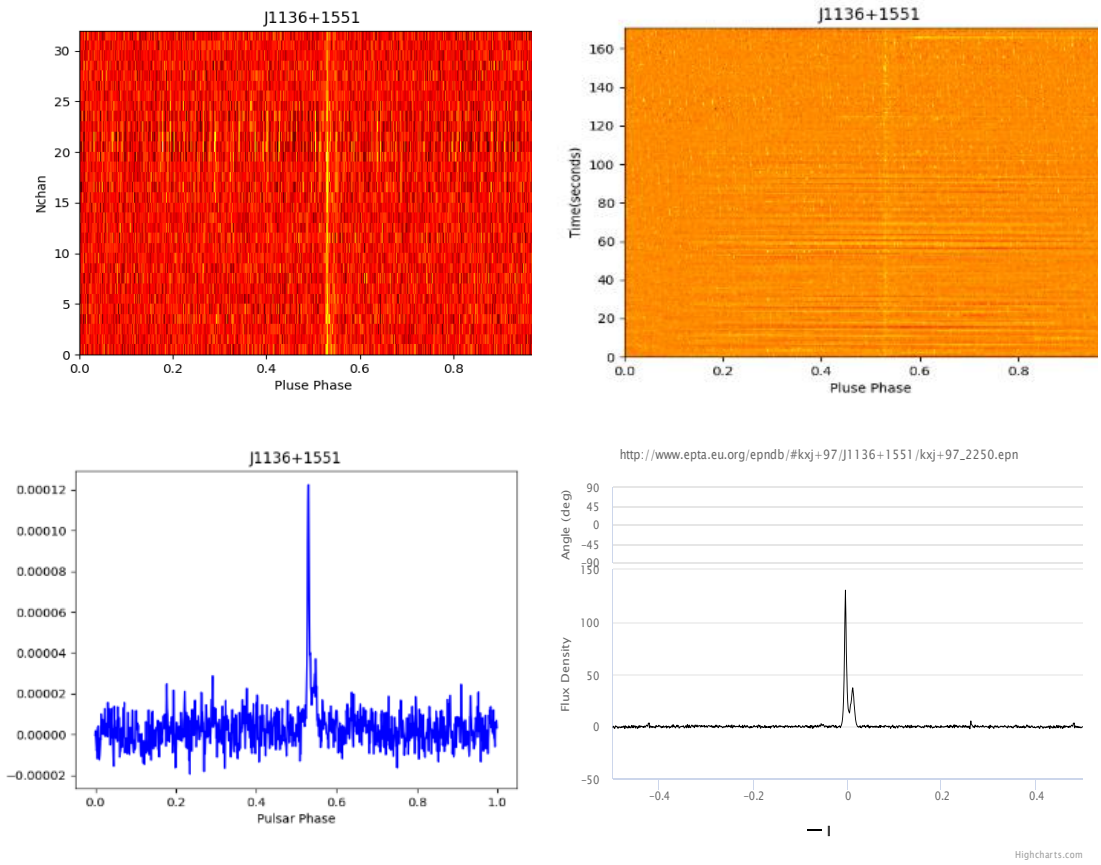


图 4.7 J1136+1551、J1932+1059、J2022+5154 柱状图和轮廓图

在观测过程中，管线能够满足实时采集和消色散处理的性能要求，通过数据帧的时间戳和秒内序号的检查逻辑未发现数据采集系统有丢包现象。数据处理系统也可以实时处理所有采集数据，长时间积分时，处理结果也正确的包含了所有的子积分数据。从频域柱状图和时域柱状图均可以清晰的看到脉冲星的脉冲。在平均轮廓图中可以看到 J2022+5154 的单峰结构，以及 J1136+1551、J1932+1059 的双峰结构。经分析比较，与 EPN(European Pular Network)检索的平均轮廓结构保持一致。

4.5 射频干扰信号消除

云南天文台 40 米射电望远镜配有 S 波段（2150MHz~2450MHz）、C 波段（4000MHz~8000MHz）和 X 波段接收机（8000MHz~9000MHz）；S 波段和 C 波段系统温度分别为 80K 和 30K。利用 S 波段开展脉冲星观测任务，观测中心频率 2256MHz。2006 年 5 月投入运行之初 RFI 相对较少，但由于该望远镜距离昆明市区较近（距离市中心直线距离 8.2 公里），且随着城市建设不断发展（距东三环直线距离 1.3 公里），包括 2G、3G、4G 手机信号频段和 WIFI 频段的 RFI 越来越多，这些干扰信号严重影响了日常射电天文观测。例如，图 4.88 展示了云南天文台 40 米射电望远镜观测到的脉冲星信号。其中除期望的脉冲星信号外，可清楚观测到 RFI。RFI 产生的来源和射电望远镜运行的机理决定了几乎所有射电望远镜均面临 RFI 问题。观测数据的好坏关系到科学成果的质量甚至结论的真伪，开展 RFI 抑制和消除方法研究对射电天文发展具有重要理论意义与实际应用价值[47]。

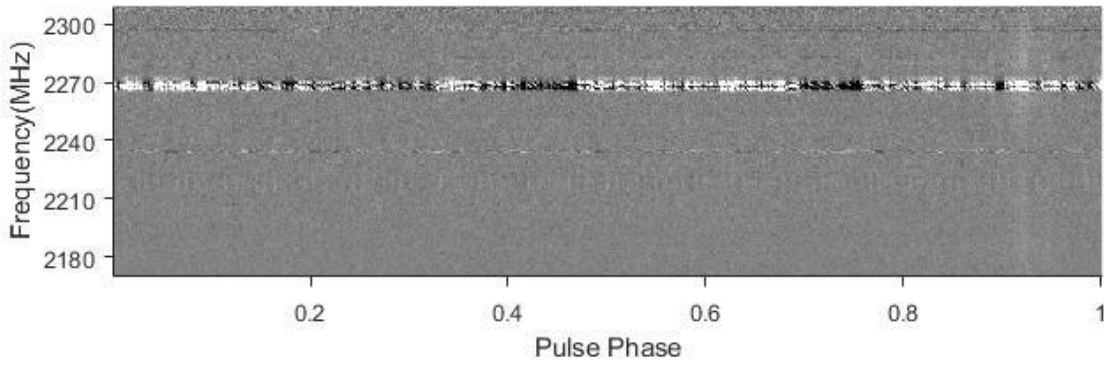


图 4.8 脉冲星 J0332+5434 频域柱状图

RFI 消除目的在于尽量保持天文信号的前提下，消除射电干扰。依据 RFI 消除方案在射电天文观测阶段的不同可分为四个环节[48]：(1)观测站预防，即观测站在选址、建造和运行期间采取的射电频率保护条令、预防和监管措施；(2)预检测，通常指的是接收机系统中使用的预检测方法，可以屏蔽掉已知的强 RFI 信号；(3)预相关，在将接收数据发送到相关处理机之前，对接收数据基于硬件或软件进行实时消减操作；(4)应用于干涉数据的后相关方案，包括对干涉合成数据的实时和离线处理。强 RFI 在以上 4 个环节都容易识别，而弱 RFI 必须在数据干涉合成提高信噪比以后即环节(4)才能识别。对于已经建好投入使用的射电望远镜，后相关

处理尤显关键和重要，本节主要针对该环节研究 RFI 消除方法。

后相关消除方法的基本思想基于：通过统计分析数据来准确标记 RFI，即，通过时频二维平面上 RFI 信号与天文信号形态特性差异标记并消除 RFI。天文信号通常呈现宽带、平滑且时间跨度大的特点，而 RFI 信号经常在时频平面上呈现为高强度像素。目前的 RFI 消除方法可分为两类。第一类方法采用基于阈值的方法，例如累计和方法[49]与阈值求和方法[50]。这类方法把 RFI 定义为在时频平面上超过某些阈值的像素。算法简单、高效，被广泛应用于射电数据处理[51]。但此类方法最大的问题在于：如何根据 RFI 源及观测天体确定阈值？尤其是在针对脉冲星等时变天体信号，阈值选择尤为关键。在对 LOFAR 的恒星撕裂时间天体 Swift J1644+57 进行研究时，并未检测到预期的源，分析原因可能是其微弱瞬时信号被认定为 RFI 而删除[52]。第二类方法采用基于机器学习的方法。近年来，机器学习尤其是深度学习技术在众多领域取得了令人瞩目的研究成果，已有研究将机器学习中的有监督学习以及深度学习的方法应用于 RFI 消除，取得一定研究进展。例如文献[53]基于 K 近邻（k-Nearest Neighbor）和混合高斯模型（Gaussian Mixture Models）对 RFI 信号进行聚类，从而实现 RFI 标记；文献[54]对基于 ANN（Artificial Neural Network）、Adaboost、GBC（Gradient Boosting Classifier）和 XGBoost（eXtreme Gradient Boosting）实现的四种有监督学习 RFI 分类方法的效果进行了分析和比较。但该类基于有监督学习方法的关键问题是：RFI 分类准确度对特征选取非常敏感。为了减少对特征的依赖，Akeret 等将深度学习的方法应用于 RFI 消除[55]，对模拟 RFI 信号取得了非常好的效果。但这种采用模拟数据对深度网络进行训练的方式，很难防止过拟合。

独立成分分析（Independent Component Analysis，ICA）起源于盲源信号分离（Blind Signal Separation，BSS）。BSS 是信号处理中一个传统而又极具挑战性的问题，指仅从若干观测的混合信号中恢复出无法直接观测的各个原始信号的过程[56]。这里的“盲”，既指源信号不可测，又指混合系统特性事先未知。所谓“鸡尾酒会问题”就是 BSS 的典型例子。ICA 是研究 BSS 的一个重要方法，基于信号高阶统计特性已成为阵列信号处理和数据分析的有力工具。显然，ICA 所涉及的问题，在数学模型上本身是欠定的，但附加上原始信号间统计独立及原始信号非高斯分布两个条件后，各原始信号可完美复原[57]。本文中，我们将射电望远镜

观测到的脉冲星信号看作观测信号，而包含其中的各 RFI 和脉冲星信号视为原始信号。各 RFI 信号和脉冲星信号间统计上相互独立且各信号符合非高斯分布，满足 ICA 假设条件。相比已有方法，首先，无需人为选择或构造 RFI 结构特征，不存在阈值选择的困惑；其次，不存在训练或学习过程，因此无需考虑构建训练样本的问题。使用该方法对云南天文台 40 米射电望远镜接收到的脉冲星观测信号进行独立成分分析，分解出独立的 RFI 信号和脉冲星信号，进而实现 RFI 消除，取得良好效果。

4.5.1　独立成分分析

ICA 是近年来发展起来的一种统计方法。该方法目的是将观察到的数据进行某种线性分解使其分解成统计独立的成分。为严格定义 ICA 模型，使用统计隐变量模型表示 n 个同一时刻接收到的射电信号 x_1，…，x_n（混合信号）：

$$\boldsymbol{x_i = a_{i1}s_1 + a_{i2}s_2 + \cdots + a_{in}s_n, i = 1 \cdots n} \tag{4.3}$$

其中，s_i 表示包含在接收信号 x_i 中的源信号（独立成分），即，RFI 信号或脉冲星信号。这里混合信号 x_i 和独立成分 s_i 均可视为随机变量。不失一般性，设 x_i 和 s_i 为零均值（混合信号 x_i 总是可以通过减去样本均值实现零均值化。式 4.3 的矩阵形式为：

$$\boldsymbol{x = As} \tag{4.4}$$

其中，**x** 和 **s** 分别为由 x_1，…，x_n 和 s_1，…，s_n 组成的随机向量；矩阵 **A** 由系数元素 a_{ij} 组成，称为混合矩阵。ICA 模型，描述通过独立成分 s_i 得到混合信号 x_i 的过程。目标是求解独立成分 **s**，式 4.4 可调整为：

$$\boldsymbol{s = Wx} \tag{4.5}$$

其中，矩阵 $\mathbf{W}=\mathbf{A}^{-1}$，称为解混矩阵。独立成分 **s** 为隐随机向量，意味着 **s** 不能被直接观测到，需要根据仅有信息——随机向量 **x** 估计出混合矩阵 **A** 和独立成分 **s**。显然，该问题在数学模型上欠定。但在附加两个假设情况时问题变得可解：（1）独立成分 s_i 间相互统计独立；（2）独立成分 s_i 拥有非高斯分布。RFI 产生自人类活动，而天文射电信号产生自宇宙天体，二者之间相互统计独立，且分布上也不会呈现高斯分布，因此电天文 RFI 消除满足 ICA 模型假设条件。

4.5.1.1 ICA 估计

由概率论中心极限定理可知，多个独立随机变量的混合信号趋近于高斯分布。因此，在 ICA 模型中，若干个独立成分 s_i 组成的混合信号 x_i 比任何独立成分 s_i 更接近高斯分布。于是可使用分离信号的非高斯性作为分离信号之间独立性的度量。

求解基本 ICA 问题的通用步骤包括三步[57]：（1）数据(混合信号)的预处理，包括中心化、白化。（2）选择或定义非高斯性(独立性)度量，建立目标函数。该函数取极值时，估计出的独立成分之间非高斯性最大。目标函数代表一种分离准则，根据不同分离准则推导出不同 ICA 估计算法。（3）用某种最优化方法最大(小)化目标函数，实现 ICA 估计。依据非高斯性度量方法的不同，ICA 估计方法可分为基于峰度（Kurtosis）和负熵（Negentropy）。由于基于峰态的 ICA 估计方法在实际应用中对边缘样本过于敏感，导致其鲁棒性较差[58]，因此本文采用基于负熵的方法对射电天文中的 RFI 进行估计。

负熵基于信息论中熵的概念。随机变量的熵可视为其所表示信息的自由度，即，越随机，熵越大。信息论的一个重要结果为：相同方差时，高斯随机变量熵最大[59]，因此可用熵衡量随机变量的非高斯性。随机变量 y 的熵定义为：

$$\boldsymbol{H(y) = -\sum_{i} P(y = a_i)\, log\, P\,(y = a_i)} \tag{4.6}$$

其中，a_i 是 y 的可能取值。可知熵为负值。为方便描述随机变量的非高斯性，负熵定义为：

$$\boldsymbol{J(y) = H(y_{gauss}) - H(y)} \tag{4.7}$$

其中，y_{gauss} 为与 y 具有相同方差的高斯随机变量。式 4.6 表明：对于高斯随机变量负熵为 0，而其他情况则非负。从统计理论出发，负熵是对非高斯性估计的最优方法[60]。但使用式 4.7 计算负熵时，涉及到估计信号的概率密度函数，在实际应用中这往往非常困难，因此通常采用更简单的方式近似计算负熵。

4.5.1.2 负熵的近似

实际计算中，可用高阶矩近似负熵的方法获得有效 ICA 估计[61]：

$$\boldsymbol{J(y) \approx \frac{1}{12} E\{y^3\}^2 + \frac{1}{48} kurt(y)^2 G} \tag{4.8}$$

其中，kurt(*y*)定义为随机变量 *y* 的峰度函数：

$$\boldsymbol{kurt(y) = E\{y^4\} - 3(E\{y^2\})^2} \tag{4.9}$$

然而，当随机变量 *y* 的峰度较大时，该近似的有效性非常有限[60]。基于最大熵原理，Hyvärinen 提出了对负熵新的近似方法[62]：

$$\boldsymbol{J(y) \approx \sum_{i=1}^{p} k_i [E\{G_i(y)\} - E\{G_i(v)\}]^2} \tag{4.9}$$

其中，k_i 为正常数，*v* 为零均值和单位方差的高斯随机变量，G_i 为非二次函数。需要注意的是式(4.9)非负，且当随机变量 *y* 呈现高斯分布时，其值为零。

当仅使用一个非二次函数时，对任意非二次函数 G，式(4.9)近似为：

$$\boldsymbol{J(y) \propto [E\{G(y)\} - E\{G(v)\}]^2} \tag{4.10}$$

实际上，式 4.10 是基于矩对式 4.8 的近似。当 $G(y)=y^4$ 时，可得式 4.8。关键之处在于函数 *G* 的选择。一般情况下，选择非快速增长函数 *G*，通过式 4.9 可得到更鲁棒的负熵估计。例如：下列 *G* 函数被证明在负熵估计中非常有效[62]：

$$\boldsymbol{G_1(u) = \frac{1}{a_1} logcosh\, a_1 u, \qquad G_2(u) = -exp\left(-\frac{u^2}{2}\right)} \tag{4.11}$$

其中，a_1 为介于 1 与 2 间的常数，通常取 $a_1 = 1$。

4.5.1.3　FastICA 算法

FastICA 算法[63]利用近似于牛顿迭代的固定点迭代方案寻找式 4.10 所表示的非高斯性最大值，即 $J(\mathbf{w}^T\mathbf{x})$的最大值。其中 $\mathbf{w}^T$ 为权重行向量，即解混矩阵 **W** 的一行。每次迭代从混合信号中恢复出一个独立成分。FastICA 算法具有适应数据类型广、收敛速度快、并行、分布、计算简单和内存需求少等优点。

FastICA 算法性能通过一个合适的非线性函数 *g* 达到最优，*g* 为式 4.11 中非二次函数 *G* 的导数：

$$\begin{aligned} \boldsymbol{g_1(u)} &= \boldsymbol{tanh(a_1 u)}, \\ \boldsymbol{g_2(u)} &= \boldsymbol{u\, exp\left(-\frac{u^2}{2}\right)} \end{aligned} \tag{4.12}$$

FastICA 算法的基本形式如下：

1. 初始化的权重向量 **w**；
2. 令 $\mathbf{w}^{+}=E\left\{xg\left(\mathbf{w}^{T}\mathbf{x}\right)\right\}-E\left\{g^{'}\left(\mathbf{w}^{T}\mathbf{x}\right)\right\}\mathbf{w}$ ；
3. 令 $\mathbf{w}=\mathbf{w}^{+}/\left\|\mathbf{w}^{+}\right\|$ ；
4. 如果没有收敛，返回第二步。

收敛意味着向量 **w** 的新值和旧值方向一致，即，其点乘近似为 1。

4.5.2 基于 ICA 的 RFI 消除的实现

上面介绍了 ICA 模型、负熵近似以及快速迭代求解对比函数的方法。为使算法更为高效，将 ICA 运用到射电天文 RFI 消除前，需考虑数据初始化问题。此外，如何从 ICA 分解得到的独立成分中选择出脉冲星信号，也是实现的关键。

4.5.2.1 射电数据预处理

实现中，数据预处理包括中心化和白化。

所谓中心化，即，将观测到的射电信号处理为零均值。可通过观测向量 **x** 减去均值向量 **m**=E{**x**}实现。此时，对式两边取期望可知，独立成分 **s** 也为零均值。中心化的目的在于简化 ICA 估计。使用中心化后的射电观测数据估计出混合矩阵 **A** 后，可将 **s** 的均值向量 $\mathbf{A}^{-1}\mathbf{m}$ 加到零均值独立成分 **s** 上，从而完成独立成分估计。

白化可减少 ICA 估计中的参数。所谓白化，即，将观测到的射电信号 **x** 转换为非相关且具有单位协方差的新向量 $\tilde{\mathbf{x}}$，使得 $\mathrm{E}\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\mathrm{T}}\}=\mathbf{I}$，可通过对 **x** 协方差矩阵特征值分解实现白化：

$$\boldsymbol{E}\{\boldsymbol{x}\boldsymbol{x}^{\boldsymbol{T}}\}=\boldsymbol{E}\boldsymbol{D}\boldsymbol{E}^{\boldsymbol{T}} \tag{4.13}$$

其中，**E** 为 $E\{\mathbf{x}\mathbf{x}^{\mathrm{T}}\}$特征向量的正交矩阵，**D** 为其特征值构成的对角矩阵：**D**=diag(d_1, …, d_n)。注意，式 4.13 的值可通过不同时刻射电观测信号 **x**(1)，…，**x**(t) 得到，因此，白化可记为：

$$\tilde{\boldsymbol{x}}=\boldsymbol{E}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{E}^{T}\boldsymbol{x} \tag{4.14}$$

其中，$\mathbf{D}^{-1/2}=\mathrm{diag}(d_1^{-1/2},\cdots,d_n^{-1/2})$。可以验证：$E\{\tilde{\mathbf{x}}\tilde{\mathbf{x}}^{\mathrm{T}}\}=\mathbf{I}$。将白化后的观测向量 $\tilde{\mathbf{x}}$ 带入可得：

$$\tilde{\boldsymbol{x}} = \boldsymbol{E}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{E}^{T}\boldsymbol{x}\boldsymbol{A}\boldsymbol{s} = \tilde{\boldsymbol{A}}\boldsymbol{s} \tag{4.15}$$

其中 $\tilde{\mathbf{A}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^{\mathrm{T}}\mathbf{x}\mathbf{A}$，为新的混合矩阵。由

$$\boldsymbol{E}\{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^{T}\} = \tilde{\boldsymbol{A}}\,\boldsymbol{E}\{\boldsymbol{s}\boldsymbol{s}^{T}\}\tilde{\boldsymbol{A}}^{T} = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{A}}^{T} = \boldsymbol{I} \tag{4.16}$$

可看出，白化后，新的混合矩阵 $\tilde{\mathbf{A}}$ 为正交矩阵，因此只有 $n(n\text{-}1)/2$ 个自由度。若不进行白化，则需估计混合矩阵 **A** 的 n^2 个参数。

为便于表述，下面所描述的射电观测信号 **x** 为中心化和白化后的数据，而混合矩阵统一表述为 **A**。

4.5.2.2　ICA 的含混性

Comon Pierre 证明了 ICA 的含混性[64]——ICA 不能确定独立成分的顺序，即，通过 ICA 模型，可由射电观测信号 **x** 估计出混合矩阵 **A**（或解混矩阵 **W**）进而得到源信号 **s**，但无法确定 **s** 的组成成分 s_1，…，s_n 中，哪一个是脉冲星信号，哪些是 RFI。

观察图 4.8 所示脉冲星观测数据可发现：（1）数据列在时间上对齐。因此脉冲星观测信号（图像）**x** 的每一行可视为一个时变信号：$x_1(t)$，…，$x_n(t)$，由此估计出的源信号向量 **s** 也由时序信号 $s_1(t)$，…，$s_m(t)$组成，为保证 **A** 是满秩矩阵，要求 $n \geq m$；（2）相对于 RFI 信号，脉冲星信号呈现出强宽带特性，即，对于脉冲星信号，数据行与行之间有更强相关性。此外，由式可知矩阵 **A** 第 i 行元素实质上是将源信号 $s_1(t)$，…，$s_m(t)$混合成第 i 个频率通道观测信号 $x_i(t)$的权重；而矩阵 **A** 第 j 列各元素对应于将第 j 个源信号 $s_j(t)$混合到各频率通道观测信号 $x_1(t)$，…，$x_n(t)$的权重。由混合矩阵 **A** 各元素含义及上述发现（2）可知，若 $s_j(t)$对应脉冲星信号，则矩阵 **A** 第 j 列将呈现均匀分布的特点，可用方差衡量随机变量均匀分布程度。此外，实际中 RFI 源数目不可预知，且一般有 n 大于 RFI 源数目，由此会造成矩阵 A 某些列方差很小且中值也很小，此时这些列不对应脉冲星信号。根据以上分析，可用方差向量 **std** 与中值向量 **med** 点除得到判别向量：

$$\mathbf{d} = \frac{\mathbf{med}}{\mathbf{std}} \tag{4.17}$$

其中，方差向量和中值向量分别由矩阵A各列方差std1, …, stdm和中值med1, …, medm组成，即

$$\boldsymbol{std_j = \frac{\sum_{i=1}^{n} (a_{ij}-\mu_j)^2}{N_j}},$$

$$\boldsymbol{med_j = \left|median(a_{ij})\right|, i = 1, \cdots n.} \quad (4.18)$$

其中，μ_j 和 N_j，分别表示矩阵 **A** 第 j 列均值和元素个数，median(·)表示中值函数。若第 j_p 个源信号 $s_{j_p}(t)$ 对应脉冲星信号，而其余 $s_j(t)$（$j \neq j_p$）对应 RFI 信号。则 j_p 为

$$\boldsymbol{j_p = arg\, max_j (d_j)},$$

$$\boldsymbol{j = 1, \cdots m} \quad (4.19)$$

其中，$d_1, \cdots, d_m$ 组成判别向量 **d**。需注意，此时 $s_{j_p}(t)$ 包含了各频率通道脉冲星信号信息。由混合矩阵 A 各元素含义可知，通过以下变换可得到以图 4.8 形式呈现的去除 RFI 后脉冲星射电信号：

$$\boldsymbol{x_p = A_p s} \quad (4.20)$$

其中，矩阵 $\mathbf{A}_p$ 通过保留混合矩阵 **A** 第 j_p 列元素，并置其余列元素为 0 得到。

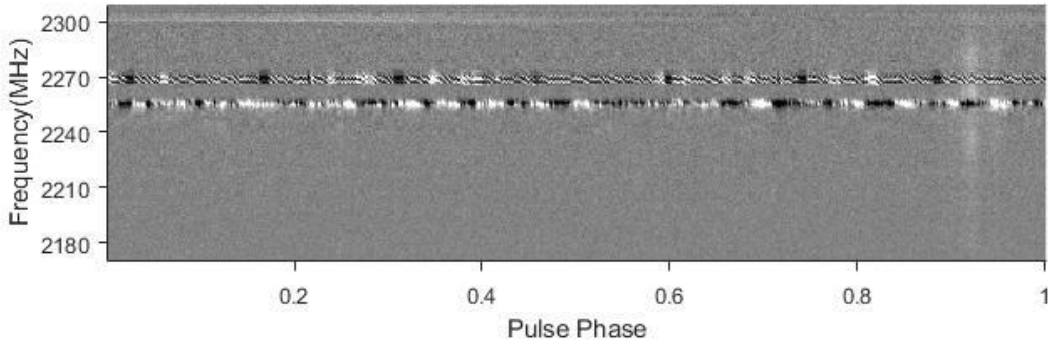
4.5.3 实验

4.5.3.1 实验数据

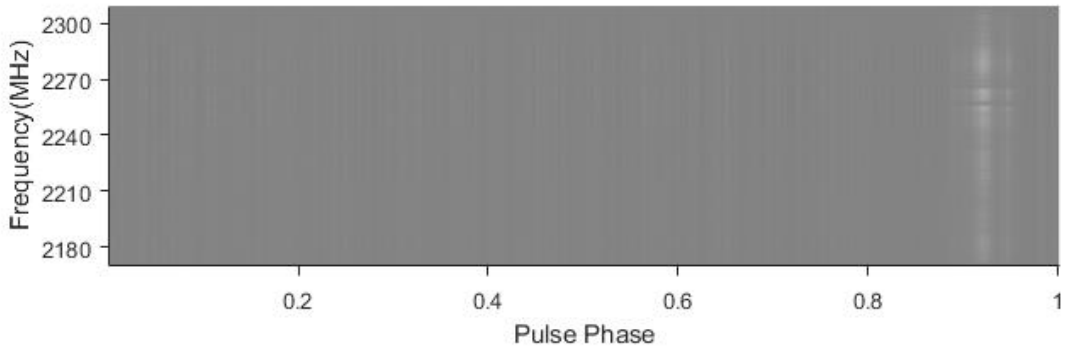
实验数据来自云南天文台 40 米射电望远镜 S 波段（2150MHz~2450MHz）脉冲星日常观测结果，观测终端为从澳大利亚 CISRO 引进的脉冲星观测终端 PDFB4（Pulsar Digital FilterBank 4）。PDFB4 对观测数据处理过程包括：非相干消色散和周期轮廓折叠。数据采样实现对脉冲星信号数据采集（采样率为 64us）。观测中心频率 2256MHz，脉冲星的观测方式为非相干消色散，观测配置为 512MHz-512Bin-512Chan，30 秒的子积分，数据存储格式为 PSRFITS 格式。由于 40 米射电望远镜距离昆明市区较近，存在较强 RFI，如 2G、3G、4G 手机信号频段和 WIFI 频段。为避免引起观测设备系统饱和，观测时需利用滤波器设备在射频波段直接将上述信号剔除，剔除上述干扰后，S 波段仅留下 60MHz 至 140MHz 相对干净的观测带宽。实际观测频段为 2170 MHz~2310MHz。在该带宽内仍存在各种类型的未引起系统饱和的干扰。

4.5.3.2　实验结果

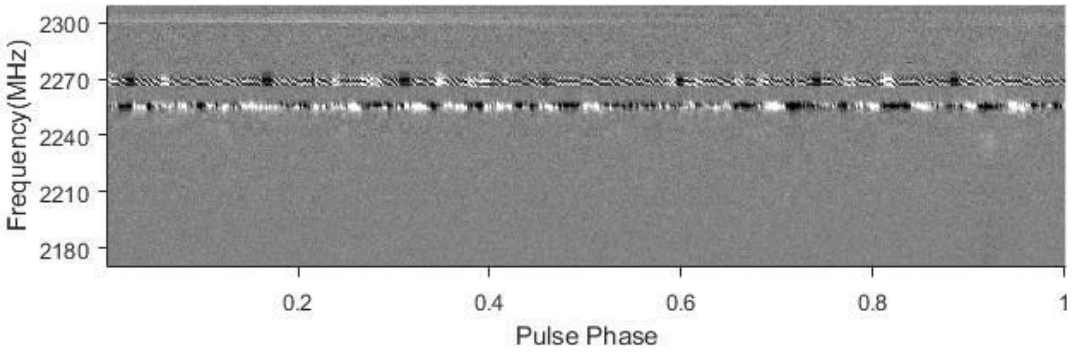
为验证本文方法的效果和可行性，利用该方法对 J0332+5434 脉冲星观测数据进行处理。对此我们选取该脉冲星在 2017 年 S 波段日常观测中比较有代表性的观测数据。图 4.9 (a)非常清晰地显示出观测干扰频点分别为 2231 MHz ~2241MHz， 2252MHz~2258MHz，2267MHz~2273MHz，2300MHz~2308MHz。其中第 2、3 个干扰频点呈现出干扰信号强、持续时间长的特点，而第 1、4 个干扰频点稍弱且在时间上显现出不确定性。实验结果如图 4.9(b)所示。从图中可看出 RFI 基本消除，仅保留下了 PSR J0332+5434 脉冲星信号。图 4.9 (c)则显示了图 4.9(a)与(c)的差值信号，即，FRI 信号。



(a)



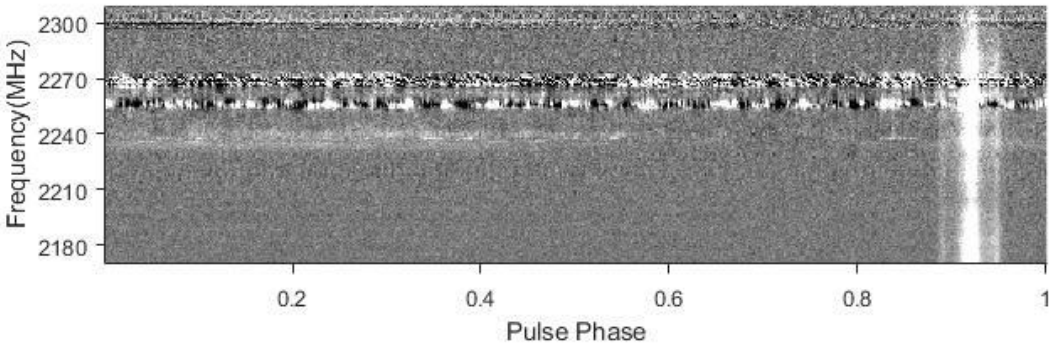
(b)



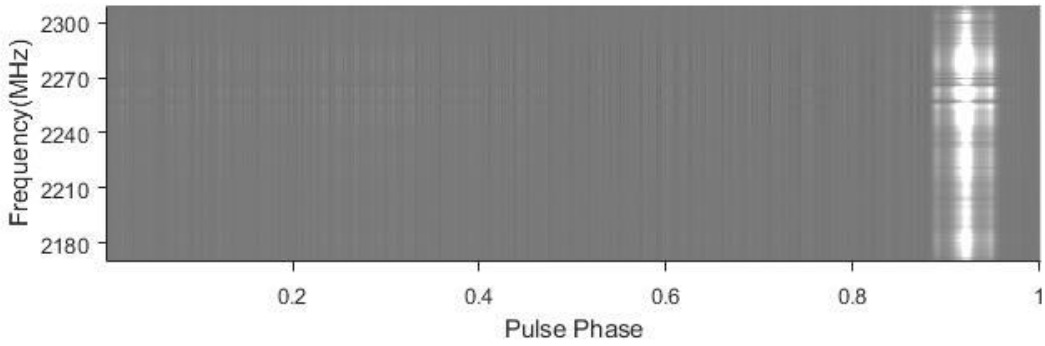
(c)

图 4.9 J0332+5434 RFI 消除效果；(a)观测数据；(b)RFI 消除结果；(c)差值信号

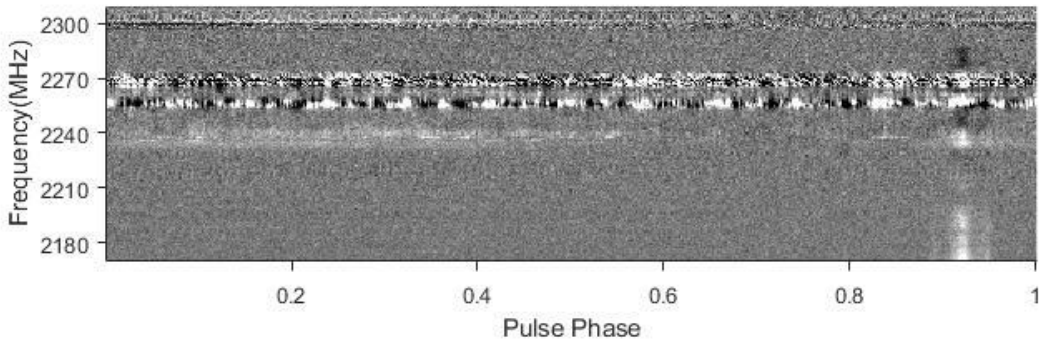
在现实应用中，对脉冲星连续观测信号取均值是一种有效且常用的增强脉冲星观测信号的手段。图 4.10 (a)显示了对 PSR J0332+5434 脉冲星连续 96 个子积分观测信号求取均值的结果。图中可明显看到，在脉冲星信号增强的同时，RFI 信号也被强化。图 4.10 (b)为采用本文方法对 96 个子积分观测信号分别提取脉冲星信号，然后求取均值的结果，图中可看出，在较好保留脉冲星信号同时，RFI 信号基本得到消除。图 4.10 (c)为图 4.101(a)与(b)的差值信号，即，96 个子积分观测信号求取均值后的 RFI 信号。



(a)



(b)



(c)

图 4.10 J0332+5434 积分均值信号 RFI 消除效果：(a) 观测信号积分均值；(b) RFI 消除结果；(c) 差值信号

为更直观反映本文方法的有效性，图 4.11 显示了对图 4.10 (a)、(b)沿频率域积分后脉冲星轮廓图。其中，蓝色曲线为观测信号中脉冲星信号轮廓图；橙色曲线为采用本文方法消除 RFI 后的脉冲星信号轮廓图。两幅轮廓图中均能清晰看到 PSR J0332+5434 脉冲星的三峰结构，但比较观测信号脉冲轮廓图可看出，本文方法有效消除了观测信号中的 RFI 信号，且脉冲星信号得到较完整保留。

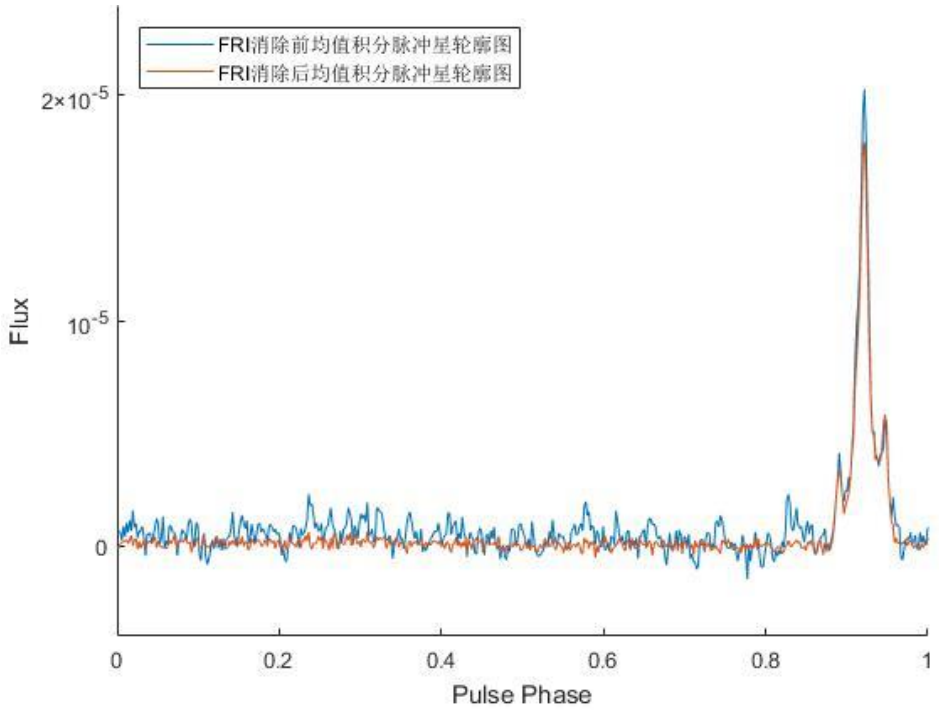


图 4.11 J0332+5434 积分均值脉冲星轮廓对比

RFI 广泛存在于射电天文观测中，而脉冲星射电观测信号可视为 RFI 信号与脉冲星信号的混合信号。

基于各 RFI 信号和脉冲星信号间统计上相互独立且各信号符合非高斯分布的特性，本文提出基于独立成分分析的 RFI 消除方法。相比已有方法，首先，无需人为选择或构造 RFI 结构特征，不存在阈值选择的困惑；其次，不存在训练或学习过程，因此无需考虑构建训练样本的问题。

4.6 小结

高时间分辨率和宽带宽采样导致的海量数据给脉冲星观测系统的采集和处理带来了巨大的挑战，一直是脉冲星观测系统研发中的核心瓶颈。

为了解决新一代脉冲星观测每秒高达几吉位甚至几十吉位采样以及实时处理的需求，本章提出了多节点多 GPU 并行消色散处理方法，设计并实现了分布式脉冲星相干消色散实时处理系统，充分保证了脉冲星射电观测中长时间、高速度的无丢包数据采集以及实时相干消色散处理。

本章对云南天文台 40 米射电望远镜进行脉冲星观测时面临射频干扰的问题也进行了相关研究，通过对观测信号进行独立成分分析，分解出独立的 RFI 信号和脉冲星信号，进而实现 RFI 消除，取得较好效果，为进一步提高脉冲星观测数

据利用率奠定良好的理论基础。

第5章　基于 OpenCL 的实时数据异构并行

在射电天文实时数据处理中，基于 CUDA 通用计算架构，利用 NVIDIA 的并行计算引擎来对数据处理进行加速，是当前主流的技术路线之一。本文相关的研究内容，包括在云台 40 米射电望远镜的相干消色散处理和明安图射电频谱日像仪的成像处理，都采用了 CUDA 通用计算架构进行加速。在两个射电望远镜的数据处理系统的实际开发测试的过程中，使用 CUDA 还是存在着一些限制和不足。以脉冲星的数据处理为例，目前的实现只能在 NVIDIA 的 GPU 硬件上实现，不能使用其他厂商的硬件。脉冲星数据处理的相关算法，需要在 CUDA 环境和普通的 CPU 环境下分别实现，增加了开发的工作量和难度。明安图射电频谱日像仪的成像处理也存在着同样的问题，受限于 NVIDIA 的硬件环境，依赖于 NVIDIA 的 GPU，未能充分利用计算资源，不仅提高了软件的开发成本，也影响了数据处理系统的部署和使用。

所以在射电数据实时计算的关键技术中，利用异构计算扩展数据的处理能力，不能仅仅依赖于 NVIDIA 的图像处理单元 GPU 和其 CUDA 开发环境。除了 NVIDIA 的 CUDA 环境外，OpenCL、DirectCompute 以及 Steam Computing 等技术，都可以用于使用图像处理单元 GPU 或者其他多核架构用于通用计算。本章主要对使用 OpenCL 技术进行射电数据实时处理加速进行研究，首先在对射电干涉成像过程进行分析的基础上，尝试采用 OpenCL 并行计算技术对其中的洁化算法进行并行优化，研究其可用性以及性能。使用 OpenCL 实现的 MUSER 数据处理程序，不仅可以运行在 NVIDIA 的 CUDA 环境下，也可以充分利用传统的 SMP 或者 NUMA 架构的通用处理器计算资源、或者是可编程门阵列等其他设备上进行数据的加速处理。可以实现一次编码，在各种多核设备上并行执行，不仅利于数据处理软件的兼容性和普适性，也有利于后期进一步在异构环境下利用各种不同的硬件加速技术来实现射电数据的实时处理。

5.1 MUSER 射电干涉阵成像算法分析

5.1.1 UVW 计算

MUSER-I 有 40 面天线，共有 40*(40-1)/2=780 个基线。可根据国际地球参考框架 ITRF（International Terrestrial Reference Frame，ITRF），可以以下公式计算计算得到空间频率 UVW 的值。

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} \sin H & \cos H & 0 \\ -\sin\delta\cos H & \sin\delta\sin H & \cos\delta \\ \cos\delta\cos H & -\cos\delta\sin H & \sin\delta \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \end{bmatrix} \tag{5.1}$$

公式（5.1）中，λ 是波长，H 是时角，δ 是参考相位点的赤纬。在计算过程中，需要使用 MUSER 观测站的经纬度和海拔进行计算（longitude=115°.2505, latitude=42°.211833333, altitude=1365.0 m）。

5.1.2 脏束和脏图

由于 MUSER 的天线数目有限，基线数目有限，UV 平面的覆盖不完整，采样的复可见度有限，加上信号干扰等因素，通过有限个采样值进行成像的原理如下。

天体辐射的亮度分布 $I(x,y)$ 与可见度函数 $V(u,v)$ 的关系为：

$$V(u,v) = F\ \left[I(x,y)\right] \tag{5.2}$$

空间频域 (u,v) 点的采样函数为：

$$S(u,v) = \sum \sigma(u-u_k, v-v_k) \tag{5.3}$$

采样函数进行 IFFT 是点扩散函数(Point Spread Function，PSF)或脏束（Dirty Beam）。脏束与采样函数之间的关系为：

$$S(u,v) = F\ \left[B(x,y)\right] \tag{5.4}$$

对可见度函数采样值进行 IFFT 得到的图像称为脏图（Dirty Image）：

$$I_D(x,y) = \frac{1}{(2\pi)^2} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} S(u,v) \cdot V(u,v) \cdot \mathrm{e}^{i2\pi(ux+vy)}\, dudv \tag{5.5}$$

简记为：

$$I_D(x,y) = F^{-1}[S(u,v)\cdot V(u,v)] \tag{5.6}$$

根据卷积定理，有：

$$I_D(x,y) = B(x,y) * I(x,y) \tag{5.7}$$

所以，脏图是脏束与亮度分布的卷积，可以通过反卷积（Deconvolution）的过程恢复出真实的天体辐射的亮度分布。

5.1.3　网格化与加权

由于可见度函数数据是不规则发布的，在进行逆傅里叶变换之前，需要将采样点分配到笛卡尔网格上，通过插值将非均匀的采样数据分布到笛卡尔网格上的过程称为网格化（Gridding）[65]。内插值法[66]和卷积核法[67]是常见的两种方法。常见的网格化卷积函数有 pillbox 函数、截断指数函数（a truncated exponential）、方波函数、截断 sinc 函数（truncated sinc function）、指数函数与正弦函数相乘、截断球面波函数（a truncated spheroidal function）等[68]。MUSER 的网格化卷积函数采用球面波函数。

MUSER 使用加权（Weighting）处理来控制点扩散函数 PSF 的形状[69]。钝化（tapering）和密度加权（density weighting）是常见的两种加权方法。密度加权方法又分为自然加权（natural weighting）、均匀加权(uniform weighting)和稳健加权（robust weighting）。本章的测试将采用自然加权操作，其对应的脏束和脏束截面图如图 5.1 所示。

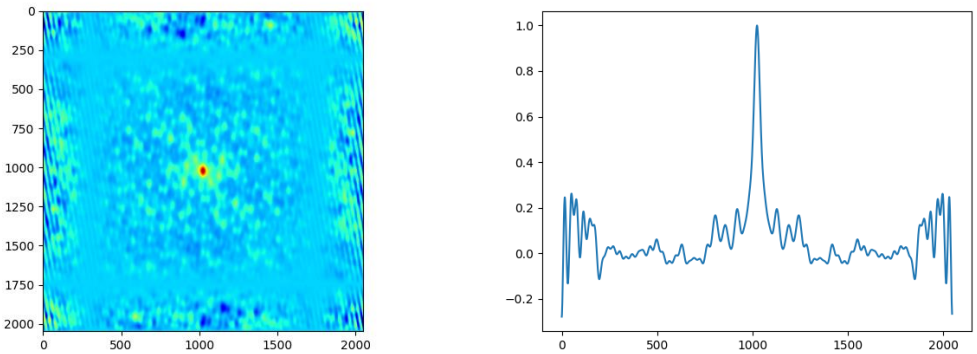


图 5.1 自然加权方法对应的脏束（左）及脏束截面图（右）

5.1.4 CLEAN 洁化

最常见的去卷积主要有两类方法：CLEAN算法[70]和MEM（the Maximum Entropy Method）算法[71]。

最经典的洁化退卷积算法由Högbom于1974年提出，此后，相继出现了许多改进版本，常见的改进算法有：Clark（1980年）[72]，Cotton-Schwab（1984年）[73]，Steer（1984年）[74]以及现在的多尺度洁化（Multi-scale Clean）算法[75]及其变型算法多频多尺度洁化算法[76]等等。本章中主要使用Högbom洁化算法展开研究。

Högbom洁化算法步骤如下：

1） 在脏图中找到最大峰值及其坐标；

2） 将脏束与最大峰值相乘，并从脏图中减去此乘积；记录峰值大小和坐标；

3） 重复步骤1和2，直到满足预设的阈值退出条件；

4） 将最大峰值的累积与洁束进行卷积；

5） 步骤三迭代退出剩余的残余图像与步骤4的结果相加（可选）；

在MUSER数据处理中最先实现的是Högbom洁化算法。

5.2 洁化(Clean)算法的并行实现

5.2.1 洁化(clean)算法并行优化研究

由于众所周知的 UV 覆盖不全问题，对可见度函数进行了采样再进行傅里叶逆变换得到的图像为脏图，是真实的太阳亮度分布与脏束的卷积，可以通过校准，权重网格化过程尽可能得到较好质量的脏图，此外利用图像恢复方法对脏图洁化来得到符合要求的图像，需要移除脏图中的旁瓣（CLEAN）。脏图可以看作是天空图像与脏束的卷积，需要进行退卷积（deconvolution），从而恢复出质量较好的称洁图（Clean Image）。去卷积的常见方法有两类：洁化（CLEAN）算法和最大熵 MEM（the Maximum Entropy Method）算法。本文使用 CLEAN 算法进行退卷积。

Clean 算法是由 Högbom 于 1974 年提出的，该算法的基本思想是：假设待

处理的图像是由点源构成，首先构造一个“脏束”，该“脏束”是 UV 平面采样分布的傅立叶逆变换，同时在原始测量值所对应的“脏图”上设置一定大小的窗口，找到其内绝对值最大点的强度 max|B|和位置，将发现的极值乘以一个因子，下一步将所得的乘积乘以脏束，再将所得结果从脏图上减去，并记录被除去的点的幅度和位置信息。然后在剩下的脏图上继续寻找下一个最大值点并重复以上过程，一直到剩下的脏图满足某个终止条件为止。迭代结束后，所有被除去的点的幅度和位置构成一张表，这个表实际上是一系列函数，最后再与一个“洁束”（Clean beam）作卷积得到最终重建的图像[77]。虽然后面不断进行改进，出现了各种变体，但基本的 clean 算法仍然被广泛使用。

通过对 Högbom clean 算法的分析，洁化算法是一个串行的迭代过程，计算效率不高，无法对整个洁化算法进行并行实现，只能分析其中可以并行的操作步骤，进行局部的程序加速。对 Högbom 的 Clean 算法进行分析，找出算法中运算量较大的若干个模块，如可见度数据网格化、脏图生成、脏束生成、直方图计算、脏图最大峰值计算等，再分析这些模块并行计算的可行性。目前对查找峰值、从脏图中减去脏束、以及脏图的残余图像加回等步骤进行了并行化分析。

查找最大峰值操作，是在二维数组中查找最大绝对值，OpenCL 内置函数中提供了 clmath.fabs()和 cl_array.max()用于计算绝对值和查找最大峰值。最大峰值在图像中的位置，可以通过数据并行，查找返回最大峰值的位置。

从脏图中减去脏图中峰值与脏束相乘的计算结果，以及迭代终止后，将脏图的残图和洁图相加，都是对二维数据进行基本运算操作，数据操作不存在依赖性，可以进行并行化处理。

5.2.2　洁化算法的 OpenCL 并行实现

本文的洁化算法的 OpenCL 并行实现包括寻找最大峰值亮度及其位置、从脏图中减去峰值亮度值与脏束相乘的计算结果、加上脏图的残余图像三个计算过程。随后将算法与 CPU 和 CUDA 上的实现进行比较，评价算法图像重建的速度。

基本算法流程如图 5.所示。

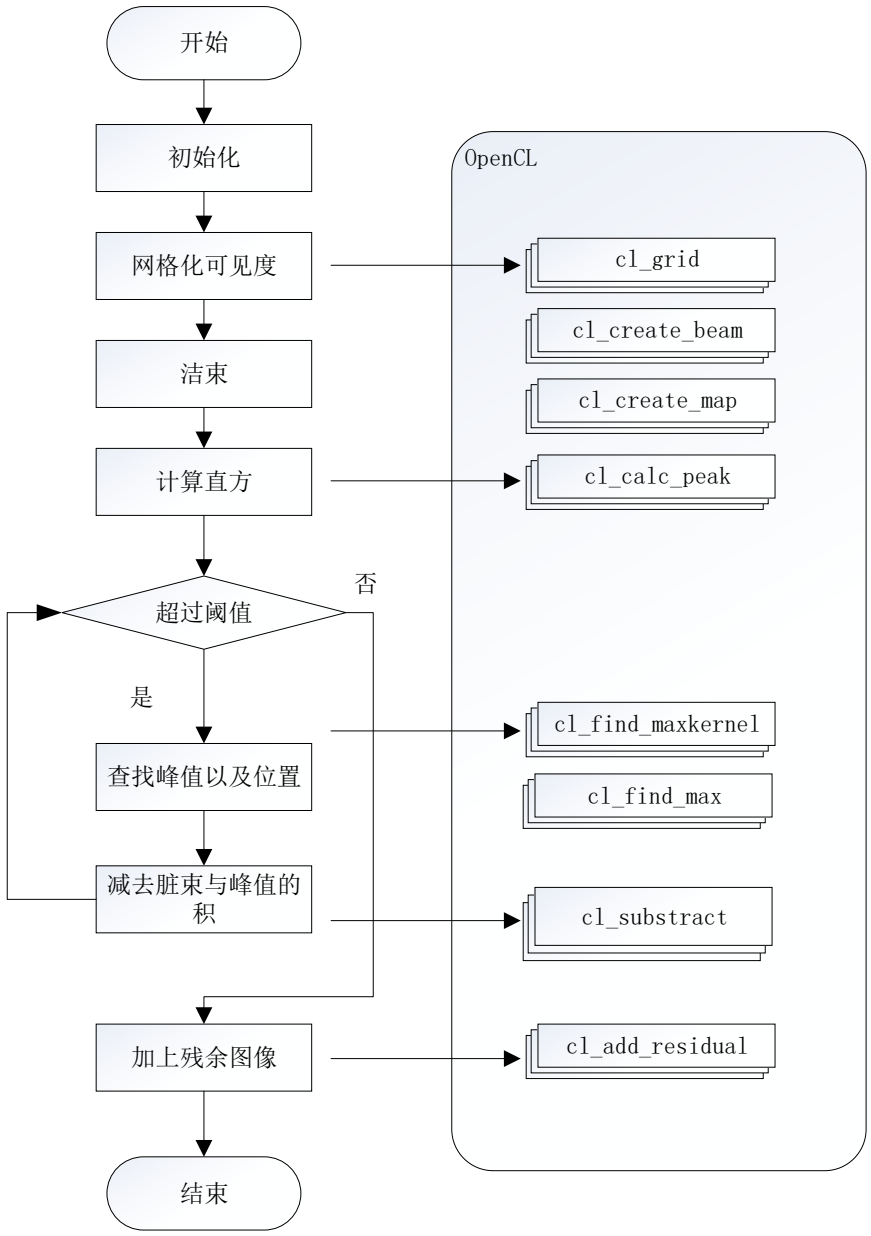


图 5.2 洁化算法并行示意图

以上算法流程图中，右边的函数是采用 OpenCL 将各个关键步骤或功能以内核函数的方式实现，从而实现洁化算法的 OpenCL 的并行。其中 cl_find_maxkernel()和 cl_find_max()用于查找峰值和其对应位置；cl_substract()用于在脏图上减去该峰值对应点源的响应；cl_add_residual 用于将残留强度加回到洁束图像。洁化算法并行优化主要是对以上四个函数进行。

查找峰值和其对应位置的主要算法步骤如下，处理的数据存放在二维数组中，首先利用 clmath.fabs 和 cl_array_max 查找脏图中的峰值。在获得脏图中当前的最大峰值后，计算该峰值在二维数组中的坐标。通过与数组中的元素进行比较，

返回二维数组的对应坐标，这里使用同步操作 barrier 和原子操作 atomic_xchg 保证不会发生竞态条件，以及保证多个线程能够同步最大值查找操作。

在计算出峰值及其坐标后，接下来将峰值与脏束相乘，并从脏图减去脏束，累加峰值与洁束进行卷积操作的结果。该算法步骤在 cl_substract 函数中并行实现。该函数的输入参数包括洁图的二维数组，脏图的二维数组，脏束的点扩散函数，洁束的点扩散函数也是保存在二维数组中。峰值位置由 cl_find_max 计算得出。图像的高度 Height 和宽度 Weight 用于算法并行时判断索引是否越界。输出的数据为洁束图像和剩余的残留强度图。

当 Högbom 算法满足退出条件后，也就是图像中显著的射电源结构被移除后，此时得到了洁束图像和剩余的残留强度图，将两个图相加，做为最终处理结果的输出。操作本身是对两个二维数组进行加法运算。此内核函数命名为 cl_add_residual()。

该函数的输入参数为两个二维数组，一个用于存放洁束图像，一个用于存放残留强度图。图像的高度 Height 和宽度 Weight 用于算法并行时判断索引是否越界。在获取二维数组的索引之后，转换为一维索引，通过多个线程并行操作数组进行求和运算，最后输出结果也是一个二维数据，存放着加回残图的洁束图像。

5.2.3　实验与性能分析

实验数据采用明安图射电频谱日像仪 MUSER 在 2017 年 2 月 15 日 12 时 1 分 26 秒 429 毫秒的观测数据。观测数据为 MUSER-I 低频阵产生的数据。MUSER 数据处理系统预处理生成的观测数据文件名为 20170215-120126_429068960.uvfits。uvfits 文件是 fits 文件的扩展，主要存放着可见度函数数据，可见度函数数据为复数，由实部和虚部构成。除此之外，在其数据首部中还存放着观测信息的元数据，如观测频率、空间频率 UVW 等。

洁化并行程序通过 pyfits 的第三方包读取 uvfits 文件，并进行可见度数据网格化，生成相应的“脏图”、“脏束”和“洁束”，然后进行洁化处理，生成长宽均为 1024 像素的图像。处理后的脏图和洁图如图 5.3 所示。

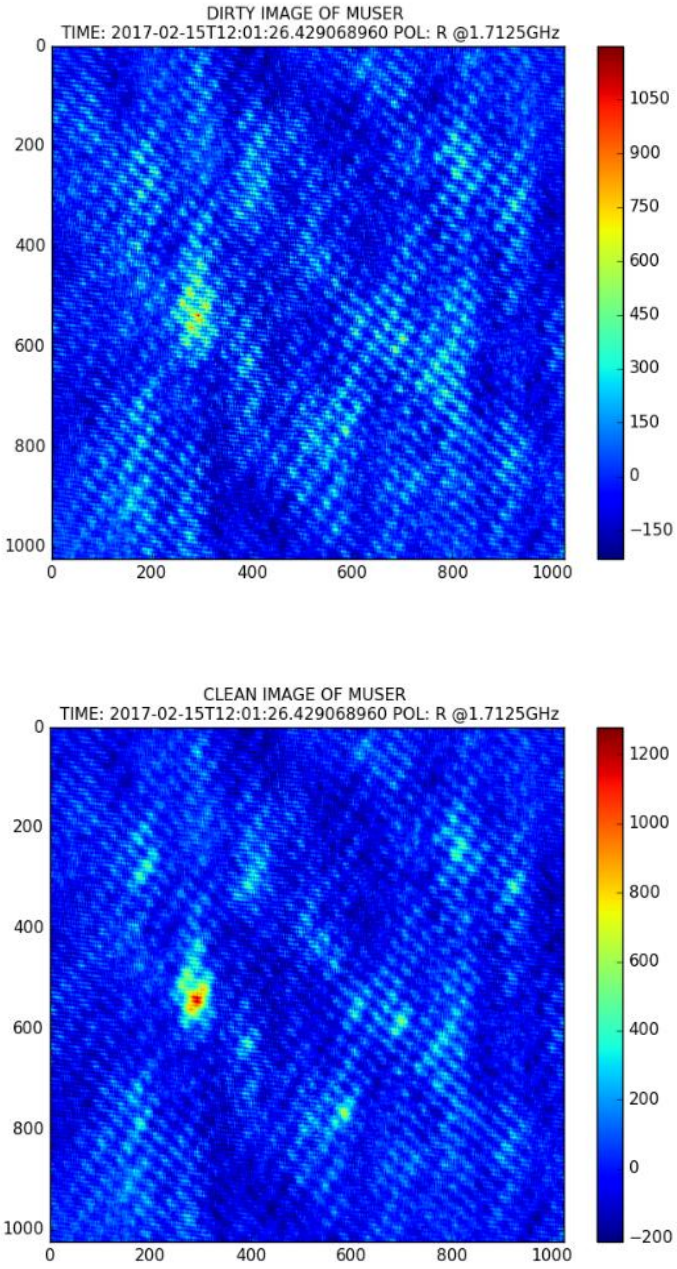


图 5.3 脏图和洁图

性能分析实验由两部分构成。首先分析 CPU 环境下，OpenCL 并行和传统的 CPU 串行模式的性能差异，主要获取 OpenCL 并行后的加速比。然后分析在使用图像处理单元 GPU 的情况下，基于 CUDA 实现和 OpenCL 实现的性能差异。

首先是在 CPU 环境下进行测试，在相同的硬件配置情况下，分别采用 OpenCL 并行模式和传统的 CPU 串行模式对 uvfits 文件进行洁化处理，测试次数为 10 次，记录每一次的处理时间。两种模式的洁化过程执行时间如图 5.4 所示。

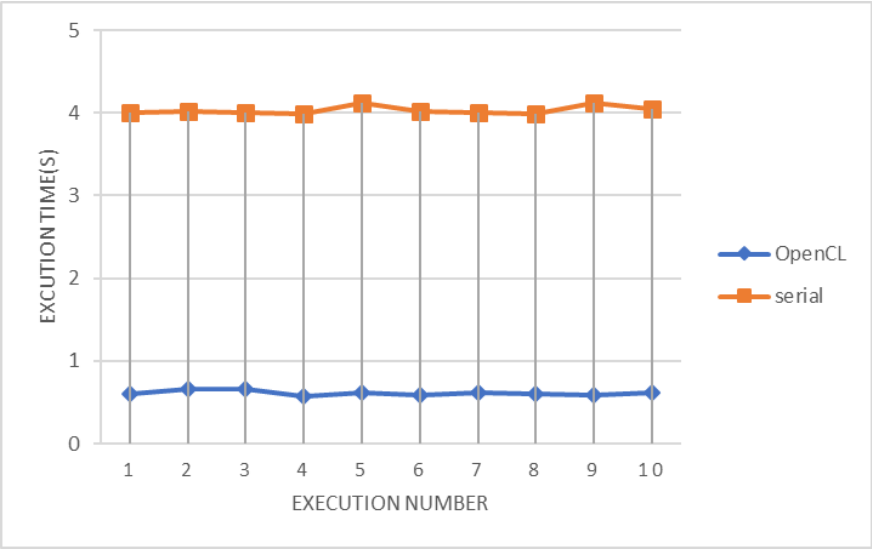


图 5.4 CPU 环境下洁化过程执行时间比较

从图 5.4 可以看出，使用传统的 CPU 串行模式进行洁化处理执行时间约为 4 秒左右，而使用 OpenCL 并行加速模式执行时间约为 0.6-0.7 秒左右。使用 OpenCL 并行加速获得的加速比为 7 倍，说明 OpenCL 并行较传统的串行模式，可以有效的利用多核架构进行通用计算的加速。

其次是在 GPU 环境下进行测试，在相同的硬件配置情况下，分别采用 OpenCL 并行模式和 CUDA 并行模式对 uvfits 文件进行洁化处理，测试次数为 10 次，记录每一次的处理时间。两种模式的洁化过程执行时间如图 5.所示。

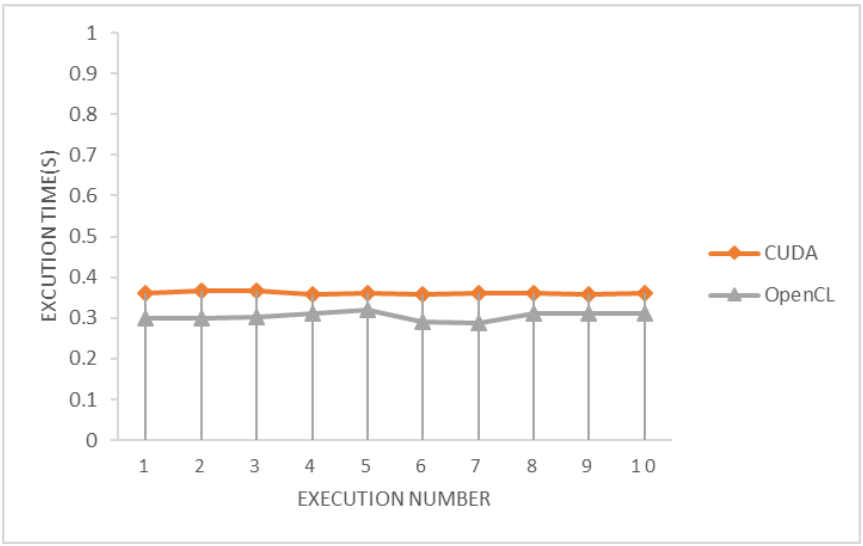


图 5.5 GPU 环境下洁化过程执行时间比较

从图 5.5 可以看出，使用 CUDA 并行进行洁化处理执行时间约为 0.3 秒左右，而使用 OpenCL 并行加速模式执行时间约为 0.35 秒左右。使用 OpenCL

OpenCL 并行较 CUDA 并行性能略慢。考虑到使用 OpenCL 会带来较好的硬件兼容性，能够跨硬件平台执行，可以支持在多核 CPU 或者多个厂商的 GPU 硬件上运行的优势，一定程度上的性能损失在很多场景下也有其应用价值。

由于实验条件限制，只在多核 CPU 和 GPU 环境下测试了洁化程序的性能。实验结果表明，利用 OpenCL 可以方便的使程序获得跨硬件平台执行的特性，一次开发，多个平台运行的优势使其具有很好的移植性，即可以利用多核 CPU 的运算能力进行加速，在图像处理单元上也有较好的性能表现。对射电天文数据的实时处理来说，对底层的异构硬件有了更广泛的支持，使得在设计和实现实时处理平台时，也有了更多的选择。对于 MUSER 数据处理来说，不依赖于 GPU 硬件，可以重新利用多核 CPU 的计算资源，对 MUSER 数据处理程序的开发和测试，对 MUSER 数据处理程序的单机运行离线处理，都提供了较大的便利。

5.3 小结

在射电天文实时数据处理中，基于 CUDA 通用计算架构，利用 NVIDIA 的并行计算引擎来对数据处理进行加速，是当前主流的技术路线之一。本文相关的研究内容，包括在云台 40 米射电望远镜的相干消色散处理和明安图射电频谱日像仪的成像处理，都采用了 CUDA 通用计算架构进行加速。在两个射电望远镜的数据处理系统的实际开发测试的过程中，使用 CUDA 还是存在着一些限制和不足。以脉冲星的数据处理为例，目前的实现只能在 NVIDIA 的 GPU 硬件上实现，不能使用其他厂商的硬件。脉冲星数据处理的相关算法，需要在 CUDA 环境和普通的 CPU 环境下分别实现，增加了开发的工作量和难度。明安图射电频谱日像仪的成像处理也存在着同样的问题，受限于 NVIDIA 的硬件环境，依赖于 NVIDIA 的 GPU，未能充分利用计算资源，不仅提高了软件的开发成本，也影响了数据处理系统的部署和使用。

所以在射电数据实时计算的关键技术中，利用异构计算扩展数据的处理能力，不能仅仅依赖于 NVIDIA 的图像处理单元 GPU 和其 CUDA 开发环境。除了 NVIDIA 的 CUDA 环境外，OpenCL、DirectCompute 以及 Steam Computing 等技术，都可以用于使用图像处理单元 GPU 或者其他多核架构用于通用计算。本章主要对使用 OpenCL 技术进行射电数据实时处理加速进行研究，首先在对射电干涉成像过程进行分析的基础上，尝试采用 OpenCL 并行计算技术对其中的洁化算

法进行并行优化，研究其可用性以及性能。使用 OpenCL 实现的 MUSER 数据处理程序，不仅可以运行在 NVIDIA 的 CUDA 环境下，也可以充分利用传统的 SMP 或者 NUMA 架构的通用处理器计算资源、或者是可编程门阵列等其他设备上进行数据的加速处理。可以实现一次编码，在各种多核设备上并行执行，不仅利于数据处理软件的兼容性和普适性，也有利于后期进一步在异构环境下利用各种不同的硬件加速技术来实现射电数据的实时处理。

MUSER 的数据处理软件目前有三个版本。早期的设计参考了天文学常用软件 MIRIAD(Multichannel Image Reconstruction, Image Analysis and Display）和 CASA 等，其数据处理在 CPU 上完成，主要用于早期的原理探索和算法验证，其速度远远不能满足实时数据处理的要求。MUSER-I 包括 40 面 4.5 米天线，观测频率 0.4G-2.0GHZ，共 64 个通道，每 3ms 产生一帧原始观测数据（100KBytes），按 10 小时观测估算，每天产生 1.125TB 观测数据。MUSER-II 包括 60 面 2 米天线，观测频率 2G-15GHZ，共 520 个通道，每 3ms 产生一帧原始观测数据（204.KBytes），按 10 小时观测估算，每天产生 2.36TB 观测数据。每帧数据对应 16 个通道，经过相应的数据处理，需要构建图像 16 张。显然，只是通过 CPU 的串行实现的算法是无法应对 MUSER 数据的实时处理。

MUSER 数据处理的第二个版本是利用了 NVIDIA 的 CUDA 计算环境，在高性能的图像处理单元上实现了数据处理过程的并行化，在单个 GPU 的计算节点上实现了加速。和基于 CPU 的实现相比，gridding 的速度提高了约 5 倍，cleaning 的速度提高了约 50 倍，然而还是不能达到实时数据处理的要求。

MUSER 数据处理的第三个版本是一个分布式架构的数据处理管线 OpenCluster，可以在多个计算节点上进行计算任务的调度、监控和分配资源、故障和任务恢复，使用 Python 语言开发，融合了工厂、组长和工人的调度模式，通过水平扩展计算节点的数量，或者垂直扩展每个计算节点的计算能力，进一步满足了 MUSER 对射电数据成像处理的要求。

虽然 MUSER 已经有了不同版本的数据处理程序，但是还存在着以下问题。首先，MUSER 的工作人员有在移动环境下处理数据的需求，基于 CPU 的版本速度不能满足要求，也不可能在移动环境下部署分布式的 OpenCluster 软件，使用 GPU 的版本要求移动计算环境必须具备 NVIDIA 的显卡，这对用户的使用增加

了限制。其次，未来的实时计算模式不局限于一个厂商的解决方案，如果将关键的算法通过一个更通用具有更好硬件兼容性的并行计算框架实现，可以更加充分利用计算集群的各个节点以及各种不同硬件的计算资源。

本章研究了基于 OpenCL 的实时数据异构并行的方法，异构并行包括多核 CPU、图像处理单元等组成的异构环境。本文结合 MUSER 的成像过程，以 MUSER 数据处理为例，在分析了射电干涉成像原理的基础上，对数据处理过程中的洁化算法展开研究，采用 OpenCL 对成像算法中的洁化算法进行并行实现。性能测试使用了实际观测数据，分别和 CPU 串行实现和 CUDA 的并行实现比较了性能。

实验结果表明，采用 OpenCL 技术进行射电数据的并行化处理，可以在提高算法对硬件平台的适用性的同时，降低开发成本，可以做到一次开发，在多个平台上编译运行。在 GPU 环境下，OpenCL 并行实现比 CUDA 并行实现效率略低在 CPU 环境下时，OpenCL 并行实现比基于 CPU 的串行实现有较大提升，使用 OpenCL 并行环境可以避免用户直接面对底层的多线程或多进程以及相关的线程和进程间同步和通讯的复杂机制。

基于以上结论，基于 OpenCL 的射电数据的实时处理，不仅可以利用 NVIDIA 的 GPU，也能利用于其它厂商的 GPU 或处理器，无论是单机环境还是分布式环境，都能够在保证数据处理效率不受影响的同时，提高了算法对硬件平台的适应性。未来将考虑使用 OpenCL 技术，对云台 40 米的脉冲星相干消色散的并行算法进行移植，使其可以充分利用计算节点的多核 CPU 的计算资源，将从 CPU/CUDA 的模式扩展到 CPU/OpenCL 的模式。

第6章 基于 Docker 的敏捷封装与部署

明安图射电频谱日像仪大部分装置已建设完成并投入观测，观测数据从望远镜采集，由相应的数据处理软件系统[78]进行数据处理。云台 40 米脉冲星的相干消色散终端也即将进入试观测。明安图射电频谱日像仪目前面临着环境部署成本较高，数据处理操作繁琐，计算需求要求较高，计算资源动态扩展等需求。而云台 40 米脉冲星的相干消色散也面临着以下问题。

(1)由于用户的软硬件环境的差异，如 Linux 操作系统的不同发行版本 CentOS，Ubuntu 等，以及不同硬件的图形处理器驱动程序、不同版本的 CUDA 运行环境的搭建，DPDK 环境等，增加了系统部署安装的复杂性；

(2)软件系统需要较多的第三方软件支持，如 psrchive，tempo, tempo2, psrcat 等，增加了不少的工作量和难度；

(3)软件系统在实际运行中还面临着软件版本和支持库的更新等问题，导致运行和维护成本较高。

针对天文软件的部署问题，需要有一种更为简单，更容易部署的技术，帮助他们轻松地部署像明安图射电频谱日像仪、云台 40 米脉冲星数据处理这样的软件系统，以便系统快速方便的部署，能够让这些的分布式的计算软件的计算能力和处理能力得到灵活的扩展，其计算资源能够合理的调度，为观测数据的实时处理提供一个虚拟化的架构的支撑[79]。

6.1 MUSER 系统的封装与部署

明安图射电频谱日像仪数据处理系统主要处理望远镜观测数据，由于观测数据量比较大，在处理数据时，将数据分为在线实时处理的数据和离线事后处理数据。系统整体功能结构主要包括除了核心数据处理，还需要有用户交互界面以及后台数据库。

通常软件系统有单机和集群两种工作方式，分别用于基于单机的少量数据处理和基于集群的实时大批量数据处理。由于观测数据不但单个文件尺寸大，而且由于自动采集频繁，集群方式采用了 OpenCluster 框架[80]，该框架用于对集群内

的节点在并行任务中进行分布式任务调度。本软件系统的数据处理程序部分采用Python 编写中央处理器和图像处理单元两种计算模式均会涉及。

6.1.1 软件封装

Docker 作为一种特殊的开源的应用容器化引擎，要实现软件环境的封装，只需在前期工作中按需求构建相应的软件镜像，并应用以及相关联依赖包打包到一个可移植的容器中,然后发布到任何流行的 Linux 机器上即可。由于它可以方便的实现虚拟化,容器是完全使用沙箱机制,相互之间不会有任何接口，对非计算机专业的天文工作着来说，后期的移植工作非常方便。目前支持两种方式构建镜像文件：

(1) 按需求配置 Dockerfile 文件，根据此文件构建新的镜像文件；

(2) 构建基础镜像启动容器，并在该容器内完打包和部署软件应用环境，然后通过 Docker 命令将部署的容器提交保存为新的镜像文件。

通过两种方法的构建的容器做成镜像文件最后将镜像文件上传至镜像仓库供天文工作者下载。

首先来介绍按需求配置 Dockerfile 文件，根据此文件构建新的镜像文件的步骤。

用构建文件构建 Docker 镜像的优点是只需要在部署时下载相应的 Dockerfile 文件，再用 docker 命令来自动的构建镜像，镜像内容则是完全根据 Dockerfile 内容制作，可以在一定程度上减少干预因素。不足的是构建过程需要掌握相关的 DockerFile 文件的构建机制。本文采用了第二种方法来部署 MUSEROS 主要过程为以下几步：

1）拉取基础系统镜像

由于 Docker 官方镜像服务器不在国内，下载官方镜像会比较缓慢，在这里我们使用国内镜像仓库拉取镜像。

```
$ sudo docker pull hub.c.163.com/public/muser-cpu:latest
```

2）编写 DockerFile 文件

其中搭建基于 CPU 或 GPU 的环境需要不同运行时和驱动文件安装，Docker 文件内容主要如下：

```
FROM hub.c.163.com/public/centos:7.2-tools

MAINTAINER chengrong <543232413@qq.com>

#Setting up the environment
RUN yum -y install git make cmake gcc gcc-c++ swig python-devel
RUN cd /root
ADD https://heasarc.gsfc.nasa.gov/FTP/software/fitsio/c/
cfitsio3410.tar.gz .
RUN tar -xvf cfitsio3410.tar.gz && cd cfitsio && mkdir -p /root/cf
&& ./
configure --prefix=/root/cf && make && make install
RUN mkdir -p /usr/include/cfitsio
RUN cp /root/cf/include/* /usr/include/cfitsio/
RUN cp /root/cf/lib/libcfitsio.a /usr/lib64/

# install CPU/GPU environment
…
# install MUSER
…
CMD ["/bin/bash"]
```

3）根据 DockerFile 文件构建镜像

完成 DockerFile 文件构建后，可使用如下命令建立镜像，并可测试 DockerFile 文件正确性。

```
$ sudo docker build -t chengrong/muser:v1
```

其次来介绍构建基础镜像启动容器的步骤。

构建基础镜像可以保证创建的镜像的稳定性，因为在容器所有的操作是和真实操作主机完全一致。只需要保证特定需挂载在容器内的主机驱动与容器环境不会发生兼容冲突即可。唯一的不足是在构建的完整镜像一般都比较大，在部署时下载该镜像可能会花费较长的时间。

采用提交现成容器构建的方法比较简单，可以直接在拉取基础镜像后进入容器内，之后就可以像在物理机上一样安装软件，最后提交容器为镜像。其中提交操作如下：

```
$ sudo docker commit 容器ID museros:1.0
 //容器提交为镜像
$ sudo docker tag Museros:1.0 192.168.1.101:5000/museros
 //将新镜像更改标签
$ sudo docker push 192.168.1.101:5000/museros
 //将新镜像提交到镜像中心
```

在完成镜像构建后，镜像可被上传到本地或云镜像仓库，这样客户机就可以用 PULL 命令将镜像下载到本机部署，即在本机启动镜像即可。同时多个镜像可以同时运行在不同的操作系统的主机中。

6.1.2 软件部署

镜像制作完成后需要下载至本地容器内启动。系统部署时，由于软件的环境不同，则可分基于中央处理器或图像处理单元不同计算设备环境下运行，下面分别对中央处理器和图像处理单元两种模式启动镜像的方法予以说明。

6.1.2.1 中央处理器模式下启动镜像

在中央处理器模式下的镜像是封装完整的镜像文件环境，启用命令直接启动镜像：

```
$ sudo docker run -ti --rm museros：1.0/bin/bash
```

其中，docker run 启动镜像命令；-ti 表示以交互的方式启动并进入某个容器；museros: 1.0 是镜像名称；--rm 表示退出时自动删除容器；/bin/bash 表示从 Shell 终端启动。

6.1.2.2 图像处理单元模式下启动镜像

与图像处理单元运行环境的镜像与中央处理器模式不同，中央处理器模式属于集成模式。在 Docker 容器启用图像处理单元环境，--device 命令可以将主机的图像处理单元设备直接挂载到容器中，然后启动容器，以下问图像处理单元的启动模式，具体指令为:

```
$ sudo docker run -ti
--privileged=true
--device/dev/nvidia0：/dev/nvidia0
--device/dev/nvidiactl：/dev/nvidiactl
--device/dev/nvidia-uvm：/dev/nvidia-uvm
--rm hub.c.163.com/chengrong/muser-gpu：1.0
/bin/bash
```

通过 Docker 技术对软件系统封装部署的总体框架如图 6.1 所示。

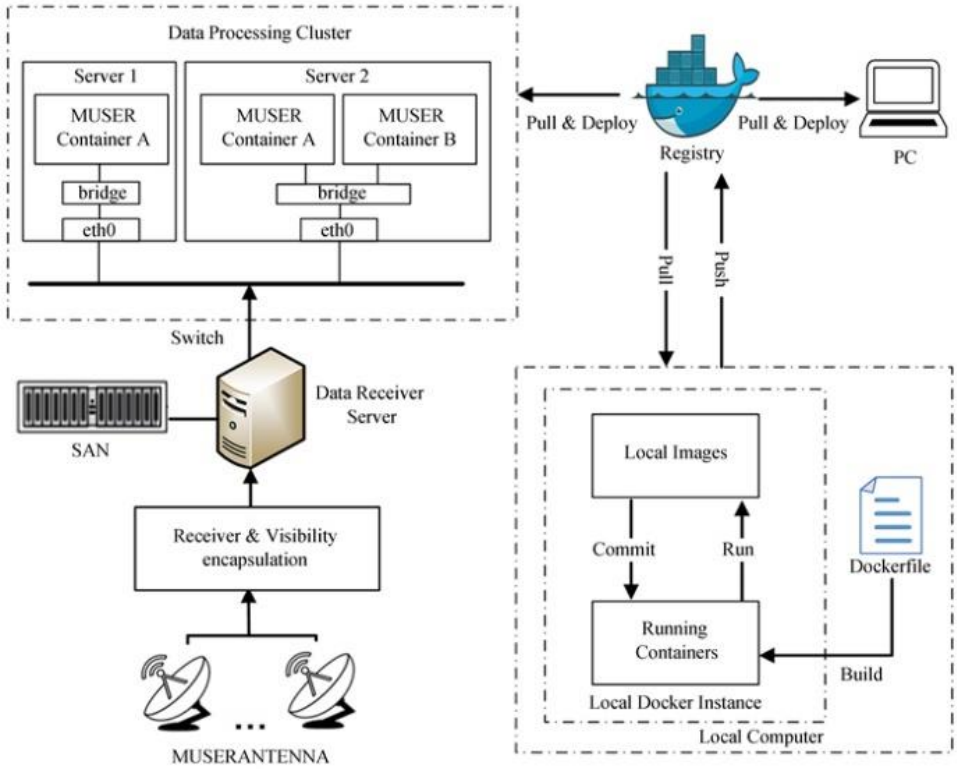


图 6.1 Docker 封装部署的总体框架

6.2 功能与性能测试

按照 Docker 封装的部署方式，完成对明安图射电频谱日像仪软件系统的 Docker 容器上封装与部署设计流程。敏捷性和可用性较前期的方法有很大的不同。本部分通过设计实验，测试并比较软件系统分别在基于不同平台上的性能表现。

6.2.1 敏捷性

明安图射电频谱日像仪数据处理系统两年内更新了数个版本。利用 Docker 进行升级，可直接在原始版本的镜像启动的容器中进行软件更新和环境升级操作，再将测试好的容器提交为升级版本镜像发布，从而实现对软件版本的敏捷化管理。

6.2.2 可用性

首先测试在主机负载不高的情况下镜像文件启动。镜像文件可以在数秒内完成启动。通过容器默认开放 TCP 的 22 号端口，远程主机与容器完成通讯，用户

也可根据需求指定为其他的端口。Docker 启动时在主机默认生成 docker0 虚拟网络适配器。经 docker0 网络适配器为多个容器提供数据收发服务。

6.2.3 性能测试

为了得到不同平台下对明安图射电频谱日像仪软件系统部署影响，本测试分别在不同平台下进行性能测试，并进行比较。日像仪软件系统可对观测到的可见度数据进行预处理，整理并导出 UVFITS 观测数据文件，需要元数据通过洁化算法进行洁化，生成脏图和洁图，由于洁化处理运算在并行系统中完成，运算量较大，本文以洁化处理的时间效率来测试 Docker 容器性能。考虑到 Docker 本身是一种虚拟化技术，以及公有云在虚拟机中使用 Docker 的用法现状，因此除物理主机外，我们将加入虚拟机和基于虚拟机上的 Docker 容器作为实验平台，分别与物理主机上的 Docker 容器进行性能对比。

基于以上讨论，实验中把软件系统分别搭建于物理机、Docker 容器、虚拟机以及虚拟机上的 Docker 4 种平台上，然后分别进行洁化处理，比较处理时间。目前洁化算法已经分别在 CUDA 和 OpenCL 环境下实现，实验在两种计算模式下分别进行。

实验中使用 KVM 虚拟机，它是基于 Linux 内核(Kernel-based)的虚拟机。KVM 支持通过透传(passthrough)方式在虚拟机内使用物理图像处理单元设备，且相较其他虚拟机，KVM 通过透传方式使用图像处理单元设备的性能表现更好[81]。

a) 实验环境：

实验环境如表 6.1。实验采用日像仪在 2015 年 11 月 1 日 12 时 8 分 49 秒 354 毫秒的观测数据，原始数据经过处理保存为 UVFITS 标准天文格式文件，文件名为 120849_354161240.uvfits。

表 6.1 实验环境

硬件环境	32 G 内存，硬盘容量 800 GB
GPU 设备	NVIDIA Corporation GM200 [GeForce GTX TITAN X] 12 GB
CPU 设备	Intel(R) Xeon(R) CPU E5-2630 v2 @ 2.60 GHz
软件环境	MUSER 1.0.0-REL (r1), docker 17.03.1-ce
CUDA 版本	NVIDIA CUDA release 7.5, V7.5.17

OpenCL 版本	OpenCL 1.2 LINUX
KVM 版本	qemu-kvm-1.5.3-141.el7_4.2

b) 实验方案：

(1) 首先实验按软件系统的计算模式分为两组，第 1 组为基于 OpenCL＋CPU 的洁化处理，第 2 组为基于 CUDA＋GPU 的洁化处理；

(2) 在每个计算模式组中再按主机平台分为两个组，分别为物理主机组和虚拟主机组。物理主机组使用物理主机和 Docker 容器作为实验平台，虚拟主机组使用虚拟主机和虚拟机上的 Docker 容器作为实验平台；

(3) 在两个平台组中分别进行基于洁化算法生成 512 × 512 像素和 1024 × 1024 像素的洁图的实验，并分别记录执行 10 次的时间；

(4) 相同计算模式下，首先比较各平台组内基于 Docker 容器和裸机下生成相同像素洁图的平均时间，然后比较物理主机组和虚拟主机组生成相同像素洁图的平均时间。

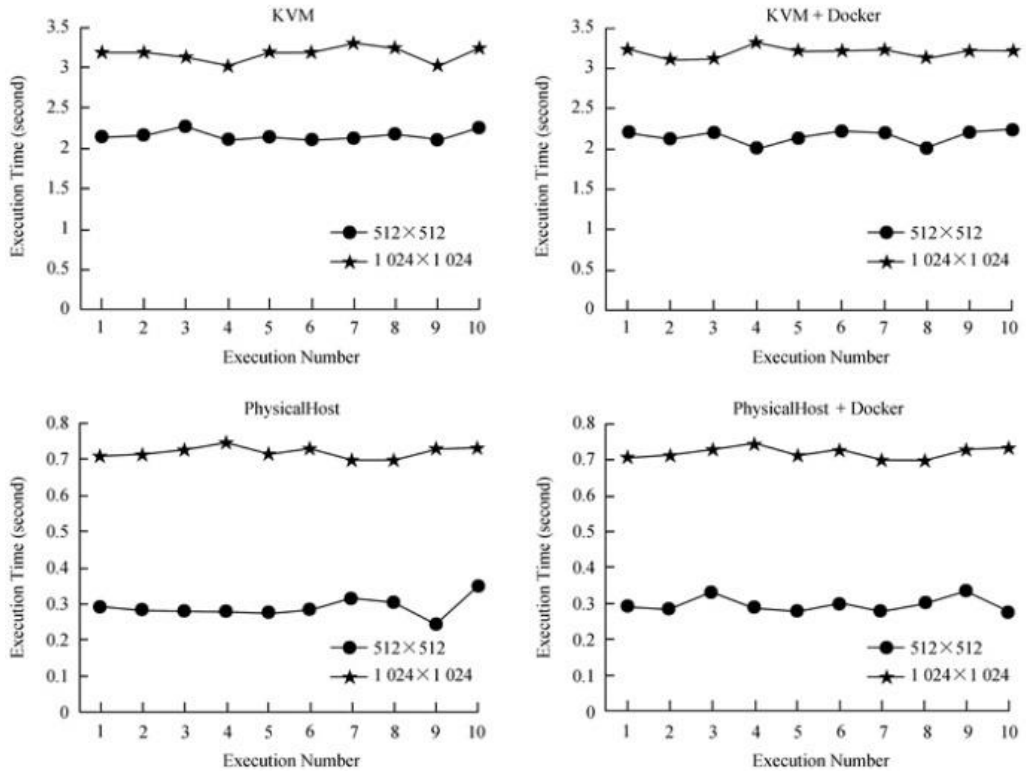


图 6.2 OpenCL + CPU 下不同主机的洁化时间对比

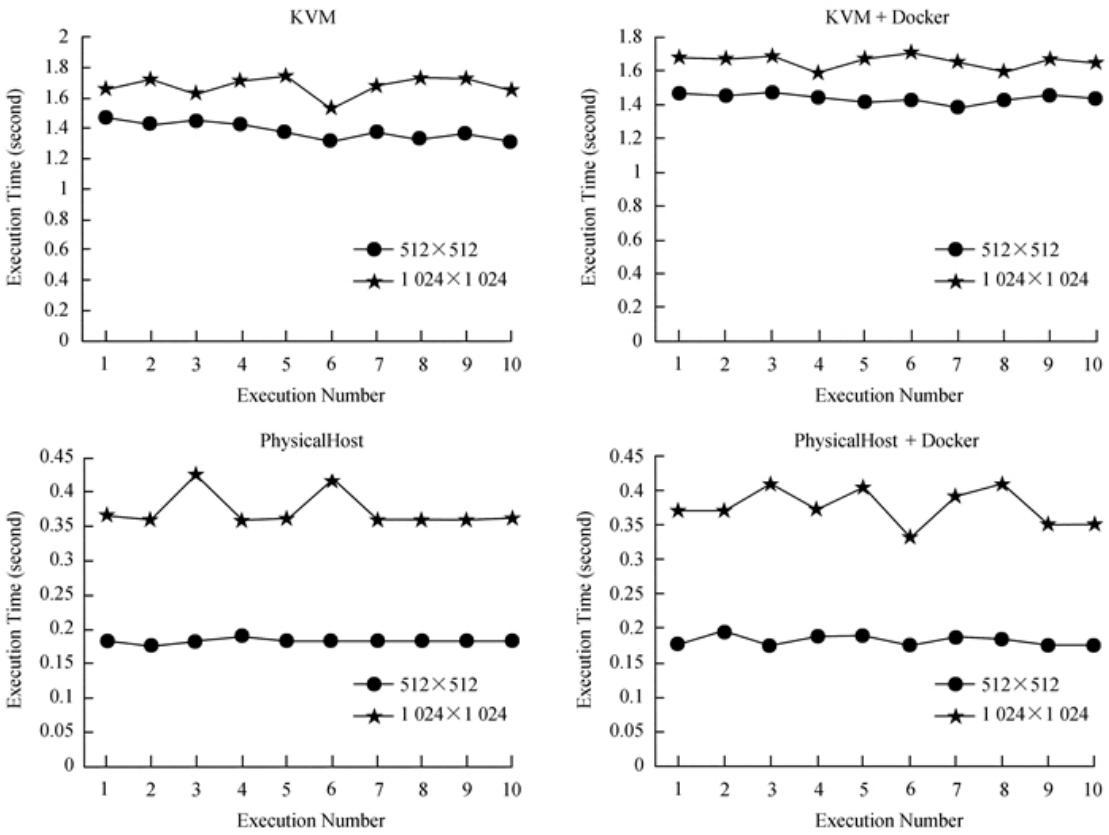


图 6.3 基于 CUDA + GPU 在不同主机上洁化处理的时间

以上实验结果可归纳为表 6.2。

表 6.2 实验结果

组名	平台	生成洁图像素 /pixel	执行次数/次	两种模式下平均时间/s	
				OpenCL + CPU	CUDA + GPU
物理主机组	Physical Host	512 × 512	10	0.292	0.182
		1 024 × 1 024	10	0.719	0.367
	Docker	512 × 512	10	0.296	0.183
		1 024 × 1 024	10	0.720	0.376
虚拟主机组	KVM	512 × 512	10	2.172	1.382
		1 024 × 1 024	10	3.161	1.671
	KVM+Docker	512 × 512	10	2.159	1.431
		1 024 × 1 024	10	3.200	1.665

(1)在基于 OpenCL + CPU 计算模式中，物理主机组中物理主机和 Docker 生

成两种像素洁图的平均时间十分接近，各约为 0.29 s 和 0.72 s；虚拟主机组的 KVM 虚拟机以及 KVM + Docker 生成两种像素洁图的平均时间也十分接近，各约为 2.16 s 和 3.20 s。相比两个平台组在相同像素下的平均时间，虚拟主机组皆高于物理主机组 2~3 s 左右；

(2) 在基于 CUDA + GPU 的计算模式中，物理主机组的物理主机和 Docker 生成两种像素洁图的平均时间十分接近，各约为 0.18 s 和 0.37 s，虚拟主机组的 KVM 虚拟机以及 KVM + Docker 的平均时间也十分接近，各约为 1.40 s 和 1.66 s。相比两组相同像素下的平均时间，虚拟主机组皆高于物理主机组 1~2 s 左右。

综上可知，无论哪种计算模式，虚拟主机组的处理时间均高于物理主机的 Docker 容器约 1~2 s，可见 Docker 容器相较于虚拟机有较明显的性能优势；而物理主机上的 Docker 容器与物理主机之间处理时间相差几乎在毫秒级别，虚拟主机上的 Docker 容器与虚拟主机之间处理时间也十分接近，可见 Docker 容器可获得与当前主机几乎一致的处理性能。相较于虚拟主机而言，Docker 容器可在物理主机上取得更好的性能，且与物理主机几乎一致。

6.3 小结

明安图频谱射电日像仪已经使用国内自主研发的数据处理系统进行观测数据的处理。数据处理系统作为一个开源的天文数据处理软件，支持在不同的环境下安装部署，支持在高性能服务器集群以及单机环境下离线的数据处理，其功能正在不断地升级和完善。

本文针对科研工作人员在安装和使用数据处理系统过程中面临的实际困难，以 Docker 容器技术为基础，实现了对系统的敏捷化构建和部署，并进行了相应的功能和性能测试。利用容器封装 MUSER 系统，可以提高开发和部署环境可重用性、可维护性以及快速持续集成的能力，改进软件开发者和使用者之间的协作，加快软件的调试升级过程。

在中央处理器和图像处理单元计算模式下分别在容器、物理主机、虚拟机和虚拟机的容器上进行洁化的性能测试表明，运行在容器内的系统与在硬件虚拟机内相比具有较低的开销和更好的性能优势，和物理主机内运行的性能非常接近。使用容器技术封装软件系统，有效减轻了科研工作者的工作负担，保障了软件系统的开发和推广使用。可以预见，基于 Docker 容器技术进行封装与部署将是射

电数据实时处理软件未来进行虚拟化或者云化的主要趋势。

第7章 总结与展望

随着新一代望远镜的建设，天文数据已经成为这个大数据时代里最具有代表性的科学领域的数据。如何对海量的天文观测数据进行实时的高性能处理，快速得到科研结果，对天文学家的科研工作提供有效的支持，具有重要的意义。本文研究了天文实时处理面临的若干问题，包括：

a) 天文数据流的实时采集方法的研究；
b) 使用 GPU 技术进行数据处理加速的研究，并实现了相干消色散的数据处理流水线在多 GPU 上的并行；
c) 使用 OpenCL 在其他异构环境进行数据处理加速的研究，实现了 MUSER 的网格化和洁化的 OpenCL 算法；
d) 基于 Docker 技术来实现分布式实时处理集群的虚拟化，实现了 MUSER 的软件容器化管理和部署。

这些方面即涵盖了底层网络传输技术、数据处理的硬件加速、实时数据处理流水线的框架以及如何在分布式环境下利用虚拟化技术进行扩展和管理。

7.1 工作总结

本文研究的数据采集方法具备一定的通用性，目前天文的数字前端后后端普遍采用工业互联技术而非专用硬件进行连接。以 CASPER 和 UniBoard 进行射电天文数字信号处理，都普遍面临着 FPGA 前端和通用计算集群后端的数据采集问题。我们讨论了超高 IO 的数据流为现在的数据采集系统和随后的数据处理系统的设计带来的难度和压力，通过对当前流行的用户态空间网络加速技术的分析和比较，探讨了 DPDK 技术作为新的数据采集技术的优势，并在此基础上结合云台 40 米射电望远镜脉冲星终端的高速实时数据流采集的需求，设计并实现了数据采集系统的框架原型，在用户模式下实现 TCP/IP 精简协议栈，直接对原始以太数据帧的解封装和封装，处理 ARP、IP 和 UDP 等网络层和传输层协议细节，在支持协议传输的同时，实现了高速的数据采集性能和较低的计算资源使用。性能测试结果表明，基于 DPDK 的数据采集不需要额外的内核和网络参数调优，在高带宽下性能稳定，能够满足当前长时间无丢包的射电观测数据采集。

数据采集问题是天文数据实时处理流水线的第一个环节，在解决了数据采集

问题后，本文研究了多核 CPU、GPU 等异构环境的计算资源，并行化数据处理算法，提高数据吞吐能力。首先以云台 40 米脉冲星数据处理为研究对象，实现了解码、消色散、合成滤波、偏振计算、折叠在 GPU 上的并行，使用环形缓冲和任务调度实现了多节点和多 GPU 的负载均衡调度，获得线性的加速比，构建了可用的脉冲星相干消色散实时数据处理管线，单节点处理 4Gb/s 观测数据流，并有较大的提升空间。另外，我们以射电干涉阵数据处理中的成像过程进行研究，采用 OpenCL 技术对射电干涉阵成像算法进行并行优化研究与实现，旨在保证算法执行效率的同时，提升算法对硬件平台的适应性，让系统的运行环境不再局限于 NVIDIA 的 GPU，进一步利用了异构环境下计算资源。

脉冲星相干消色散的实时数据处理与射电干涉成像 MUSER 的实时数据处理，都是在分布式计算集群下，利用计算框架下实现的数据处理的调度，在单节点上研究了数据采集、基于 GPU 和 OpenCL 的数据处理算法并行和优化的基础上，我们研究了分布式计算在轻量级虚拟化环境下的应用、部署和运维。基于 Docker 技术，为实时分布式计算集群提供弹性的云服务，在保证数据处理性能不变的前提下，提高集群的业务部署和升级演进的效率。结合 MUSER 当前的数据处理软件包，研究了镜像构建、软件封装和部署，以及功能与性能的测试。

7.2 展望

天文数据实时处理是目前天文观测装置普遍面临的挑战性的工作。未来，随着天文观测设备的发展，观测数据量的带宽和总量也会迅速的增长。本文以云台 40 米脉冲星的相干消色散以及 MUSER 的射电干涉成像为具体的研究对象，研究了高性能的实时数据采集和实时处理的加速方法，这些技术是未来天文实时数据处理的重要工具。

基于用户态空间的网络加速技术，不仅在性能上可以满足万兆以太网上线速的数据采集，而且可以应用于未来广泛使用的 40G 以及 100G 以太网络环境。如 MUSER 数字相关器升级，升级后的数据输出带宽会达到几十个 GB/s，但需要进一步完善协议栈支持，如支持组播和 TCP 协议，并能提供 socket 兼容的接口，可以直接移植或者支持现有的数据采集软件。

目前多数射电望远镜都是使用或者计划使用 GPU 来进行相关、干涉或者谱线等数据处理，如何在 GPU 服务器和 GPU 集群上，进行多设备和多机的并行，除了在算法上进行优化之外，如果有高级语言接口如 Python wrapping 进行封装，

用户开发效率会更高，这些迫切需要异构环境多核环境的完整的框架技术的支持，在目前的框架原型上还有更多的工作要开展。

参考文献

[1] 吴鑫基. 现代天文学与诺贝尔物理学奖 [J]. 中国国家天文, 2012, 11): 14-23.

[2] 崔辰州, 于策, 肖健, et al. 大数据时代的天文学研究 [J]. 科学通报, 2015, 60(Z1): 445-449.

[3] Wiley K, Connolly A, Krughoff S, et al. Astronomical image processing with hadoop; proceedings of the Astronomical data analysis software and systems XX, F, 2011 [C].

[4] Schaaf K, Broekema C, Diepen G, et al. The Lofar Central Processing Facility Architecture [J]. Experimental Astronomy, 2004, 17(1-3): 43-58.

[5] Varbanescu A L, Van Amesfoort A S, Cornwell T, et al. Building high-resolution sky images using the Cell/BE [J]. Scientific Programming, 2009, 17(1): 113-134.

[6] Berriman G B, Groom S L. How will astronomy archives survive the data tsunami? [J]. Commun ACM, 2011, 54(12): 52-56.

[7] Lorimer D R, Kramer M. Handbook of pulsar astronomy [M]. Cambridge University Press, 2005.

[8] Lyne A, Graham-Smith F. Pulsar astronomy [M]. Cambridge University Press, 2012.

[9] 吴鑫基, 乔国俊, 徐仁新. 脉冲星物理 [M]. 北京大学出版社, 2018.

[10] Joshi B, Lyne A, Kramer M. Next Generation Software Based Instruments for Pulsar Astronomy [J]. 2003,

[11] Kramer M, Lyne A, Joshi B, et al. COBRA, a digital receiver at Jodrell Bank; proceedings of the Proc IUCAF RFI Mitigation Workshop, F, 2001 [C].

[12] Hotan A. High-Precision baseband timing at Parkes; proceedings of the Binary Radio Pulsars, F, 2005 [C].

[13] 徐永华, 罗近涛, 李志玄, et al. 基于 MARK5B+ DSPSR 的基带数字脉冲星观测系统 [J]. 天文研究与技术: 国家天文台台刊, 2013, 10(004): 352-358.

[14] 蔡佳慧. 光学综合孔径望远镜中电感式位移传感器的研究 [D]; 南京航空航天大学, 2010.

[15] 李莉. 日像仪图像重建算法研究 [D]; 西安电子科技大学, 2013.

[16] Dewdney P, Hall P, Schillizzi R, et al. The square kilometre array [J]. Proceedings of the Institute of Electrical and Electronics Engineers IEEE, 2009, 97(8): 1482-1496.

[17] Weinreb S. Digital radiometer [J]. Proceedings of the Institute of Radio Engineers, 1961, 49(6):

1099-&.

[18] Escoffier R, Comoretto G, Webber J, et al. The ALMA correlator [J]. Astronomy & Astrophysics, 2007, 462(2): 801-810.

[19] Perley R, Chandler C, Butler B, et al. The Expanded Very Large Array: A new telescope for new science [J]. The Astrophysical Journal Letters, 2011, 739(1): L1.

[20] Hampson G, Brown A, Bunton J, et al. ASKAP Redback-3—An agile digital signal processing platform; proceedings of the 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS), F, 2014 [C]. IEEE.

[21] Salminen T. Flexible and transparent buffering of radio astronomy measurements: VLBI-streamer and Flexbuff [J]. 2015,

[22] Manley J R. A scalable packetised radio astronomy imager [D]; University of Cape Town, 2015.

[23] Duplain R, Ransom S, Demorest P, et al. Launching guppi: the green bank ultimate pulsar processing instrument; proceedings of the SPIE Astronomical Telescopes+ Instrumentation, F, 2008 [C]. International Society for Optics and Photonics.

[24] 汪敏, 韩金林, 白金明, et al. 建设云南景东 90 米脉冲星射电望远镜 [J]. 云南科技管理, 2016, 29(5): 90-90.

[25] 韩发新. 基于 GPU 的射电脉冲星的实时并行数据处理; proceedings of the 2013 中国天文学会学术年会文集, F, 2013 [C].

[26] 李佳功. 基于 GPU 的脉冲星相干消色散技术的研究和应用 [D]; 昆明理工大学, 2015.

[27] Magro A. A Real-Time, GPU-Based, Non-Imaging Back-End for Radio Telescopes [J]. arXiv preprint arXiv:14018258, 2014,

[28] Ammendola R, Biagioni A, Frezza O, et al. NaNet: a flexible and configurable low-latency NIC for real-time trigger systems based on GPUs [J]. Journal of Instrumentation, 2014, 9(02): C02023.

[29] Luebke D, Harris M, Govindaraju N, et al. GPGPU: general-purpose computation on graphics hardware; proceedings of the Proceedings of the 2006 ACM/IEEE conference on Supercomputing, F, 2006 [C]. ACM.

[30] Munshi A, Gaster B, Mattson T G, et al. OpenCL programming guide [M]. Pearson Education, 2011.

[31] Du P, Weber R, Luszczek P, et al. From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming [J]. Parallel Computing, 2012, 38(8): 391-407.

[32] Dua R, Raja A R, Kakadia D. Virtualization vs containerization to support paas; proceedings of the 2014 IEEE International Conference on Cloud Engineering, F, 2014 [C]. IEEE.

[33] Nguyen N, Bein D. Distributed MPI cluster with Docker Swarm mode; proceedings of the Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual, F, 2017 [C]. IEEE.

[34] 徐永华, 李纪云, 张颖倩, et al. 相干消色散脉冲星观测系统的研究 [J]. 天文研究与技术, 2015, 12(4): 480-486.

[35] SIGPROC [M].

[36] The DSPSR Project [M].

[37] Sarkissian J M, Carretti E, Van Straten W. The Parkes Pulsar Backends; proceedings of the AIP Conference Proceedings, F, 2011 [C]. AIP.

[38] Pipeline for Extensible Lightweight Imaging and CAlibratioN [M].

[39] SKA Non Imaging Processing Concept description: GPU processing for real-time isolated radio pulse detection [M].

[40] Baffa C. SKA pulsar search: technological challenges and best algorithms development; proceedings of the Ground-based and Airborne Instrumentation for Astronomy V, F, 2014 [C]. International Society for Optics and Photonics.

[41] HTGS [M].

[42] The PSRDADA Project [M].

[43] Kocz J, Greenhill L, Barsdell B, et al. Digital Signal Processing Using Stream High Performance Computing: A 512-Input Broadband Correlator for Radio Astronomy [J]. Journal of Astronomical Instrumentation, 2015, 4(01n02): 1550003.

[44] Recnik A, Bandura K, Denman N, et al. An efficient real-time data pipeline for the CHIME Pathfinder radio telescope X-engine; proceedings of the Application-specific Systems, Architectures and Processors (ASAP), 2015 IEEE 26th International Conference on, F, 2015 [C]. IEEE.

[45] David H E M, Danny C P, Matthew L, et al. The Breakthrough Listen Search for Intelligent Life: A Wideband Data Recorder System for the Robert C. Byrd Green Bank Telescope [J]. Publications of the Astronomical Society of the Pacific, 2018, 130(986): 044502.

[46] Edwards R T, Hobbs G, Manchester R. TEMPO2, a new pulsar timing package–II. The timing model and precision estimates [J]. Monthly Notices of the Royal Astronomical Society, 2006, 372(4):

1549-1574.

[47] 戴伟，尚振宏，徐永华，刘辉，杨亚光，强振平. 基于独立成分分析的射频干扰信号消除方法 [J]. 天文研究与技术, 2019,

[48] 安涛, 陈骁, Mohan, et al. 射电频率干扰的消减 [J]. 天文学报, 2017, 5): 18-39.

[49] Baan W A, Fridman P A, Millenaar R P. Radio frequency interference mitigation at the Westerbork Synthesis Radio Telescope: Algorithms, test observations, and system implementation [J]. Astronomical Journal, 2004, 128(2): 933-949.

[50] Offringa A R, De Bruyn A G, Biehl M, et al. Post-correlation radio frequency interference classification methods [J]. Monthly Notices of the Royal Astronomical Society, 2010, 405(1): 155-167.

[51] Akeret J, Seehars S, Chang C, et al. HIDE & SEEK: End-to-end packages to simulate and process radio survey data [J]. Astronomy and computing, 2017, 18(8-17.

[52] Cendes Y, Wijers R a M J, Swinbank J D, et al. LOFAR Observations of Swift J1644+57 and Implications for Short-Duration Transients [J]. Physics, 2014,

[53] Wolfaardt C J. Machine learning approach to radio frequency interference (RFI) classification in radio astronomy [J]. 2016,

[54] Bethapudi S, Desai S. Separation of pulsar signals from noise using supervised machine learning algorithms [J]. Astronomy & Computing, 2018,

[55] Akeret J, Chang C, Lucchi A, et al. Radio frequency interference mitigation using deep convolutional neural networks [J]. Astronomy and computing, 2017, 18(35-39.

[56] Cardoso J F. Blind signal separation: statistical principles [J]. Proc of IEEE, 2009, 86(10): 2009-2025.

[57] Hyvärinen A, Karhunen J, Oja E. What is Independent Component Analysis? [M]. John Wiley & Sons, Inc., 2003.

[58] Huber P J. Projection Pursuit [J]. Annals of Statistics, 1985, 13(2): 435-475.

[59] Cover T M, Thomas J A. Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing) [M]. Wiley-Interscience, 2017.

[60] Hyvärinen A, Oja E. Independent component analysis: algorithms and applications [J]. Neural Networks, 2000, 13(4): 411-430.

[61] Jones M C, Sibson R. What Is Projection Pursuit [J]. Journal of the Royal Statistical Society, 1987, 150(1): 1-37.

[62] Hyvärinen A. New Approximations of Differential Entropy for Independent Component Analysis and Projection Pursuit [J]. Advances in Neural Information Processing Systems, 1997, 10(273-279.

[63] Hyvarinen A. Fast and Robust Fixed-Point Algorithms for Independent Component Analysis [J]. IEEE Transactions on Neural Networks, 1999, 10(3): 626.

[64] Comon P. Independent component analysis, a new concept? [M]. Elsevier North-Holland, Inc., 1994.

[65] Schomberg H, Timmer J. The gridding method for image reconstruction by Fourier transformation [J]. IEEE transactions on medical imaging, 1995, 14(3): 596-607.

[66] Thompson A, Bracewell R. Interpolation and Fourier transformation of fringe visibilities [J]. The Astronomical Journal, 1974, 79(11-24.

[67] Jackson J I, Meyer C H, Nishimura D G, et al. Selection of a convolution function for Fourier inversion using gridding (computerised tomography application) [J]. IEEE transactions on medical imaging, 1991, 10(3): 473-478.

[68] Sedarat H, Nishimura D G. On the optimality of the gridding reconstruction algorithm [J]. IEEE transactions on medical imaging, 2000, 19(4): 306-317.

[69] Boone F. Weighting interferometric data for direct imaging [J]. Experimental Astronomy, 2013, 36(1-2): 77-104.

[70] Högbom J. Aperture synthesis with a non-regular distribution of interferometer baselines [J]. Astronomy and Astrophysics Supplement Series, 1974, 15(417.

[71] Narayan R, Nityananda R. Maximum entropy image restoration in astronomy [J]. Annual review of astronomy and astrophysics, 1986, 24(1): 127-170.

[72] Clark B. An efficient implementation of the algorithm'CLEAN' [J]. Astronomy and Astrophysics, 1980, 89(377.

[73] Schwab F. Relaxing the isoplanatism assumption in self-calibration; applications to low-frequency radio interferometry [J]. The Astronomical Journal, 1984, 89(1076-1081.

[74] Steer D, Dewdney P, Ito M. Enhancements to the deconvolution algorithm'CLEAN' [J]. Astronomy and Astrophysics, 1984, 137(159-165.

[75] Cornwell T J. Multiscale CLEAN deconvolution of radio synthesis images [J]. IEEE Journal of Selected Topics in Signal Processing, 2008, 2(5): 793-801.

[76] Rau U, Cornwell T J. A multi-scale multi-frequency deconvolution algorithm for synthesis

imaging in radio interferometry [J]. Astronomy & Astrophysics, 2011, 532(A71.

[77] 冯勇, 陈坤, 邓辉, et al. 基于 OpenCL 的 MUSER CLEAN 算法研究与实现 [J]. 天文学报, 2017, 58(2): 55-64.

[78] Wang F, Mei Y, Deng H, et al. Distributed data-processing pipeline for mingantu ultrawide spectral radioheliograph [J]. Publications of the Astronomical Society of the Pacific, 2015, 127(950): 383.

[79] 余程嵘 王, 戴伟, 邓辉, 王锋, 卫守林. 基于 Docker 的射电干涉阵软件系统敏捷封装与部署 [J]. 天文研究与技术, 2019, 16(1): 123-130.

[80] Wei S, Wang F, Deng H, et al. OpenCluster: A Flexible Distributed Computing Framework for Astronomical Data Processing [J]. Publications of the Astronomical Society of the Pacific, 2016, 129(972): 024001.

[81] Walters J P, Younge A J, Kang D I, et al. GPU passthrough performance: A comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL applications; proceedings of the 2014 IEEE 7th international conference on cloud computing, F, 2014 [C]. IEEE.

致　谢

光阴似箭，博士的学生生涯马上就要结束了，成为我的人生中一段特别难忘的经历。几年的学习生活，使我不仅在专业学术能力上得到了提升，也使我学会了很多做事做人的道理。然而，在这感觉无比漫长的读博期间，如果没有周围的老师和同学、同事和朋友、亲人的鼓励和支持，也许未必有勇气能够孤独而茫然走到最后完成学业，在此，向他们表示最真诚的感谢。

感谢我的博士研究生导师王锋教授。本文研究工作是在王锋教授的悉心指导完成的，从论文选题、课题研究、实验测试到论文的撰写与修改，王老师倾注的大量的时间和心血并给予了巨大的帮助。感谢王老师多年来在科研上提供的良好实验环境、工程实践机会，工作和生活中给予的无微不至的关怀和帮助。值此论文完成之际，向王老师表示由衷的敬意!

感谢实验室邓辉老师、段晓虹老师、梁波老师、卫守林老师、彭玮老师、张晓丽老师等多位老师多年来的帮助和支持，实验室是一个积极热情，团结友爱、互相帮助和积极向上的实验室，总是给我一种温暖如家的感觉。

感谢实验室一起努力工作的同学们，感谢一起参与云台 40 米脉冲星和 MUSER 数据处理研究工作的梅盈同学、秦天骏同学、姚坤同学、朱彦飞同学、冯勇同学、余程嵘同学。

感谢云南天文台汪敏老师、郝龙飞老师、徐永华老师、李志玄同学对我研究课题的支持和帮助，感谢季凯帆老师对我的鼓励和指导。

感谢中国科学院国家天文台，明安图观测基地的颜毅华老师、王威老师等 MUSER 项目团队的老师在项目期间给予的支持和帮助。

感谢我的父亲母亲，感谢你们对我的养育之恩，父母渐渐老去，而读书的我没能好好陪伴父母，世界那么大，想带父母去看看的愿望一定要去实现。感谢我的妻子和可爱的女儿，读书期间，我自身工作和学习的压力，也有形无形的影响到了你们的生活，在这里表示歉意，感谢在此期间给予我的包容、关爱和鼓励!

最后，衷心感谢论文的评审老师和答辩老师们的辛勤工作！

作者简历及攻读学位期间发表的学术论文与研究成果

作者简历

姓名：戴伟　　性别：男　　出生日期：1976.10.12　　籍贯：山东莱州

1994 年 09 月— 1998 年 06 月，在天津大学（计算机系）获得学士学位
2001 年 09 月— 2004 年 06 月，在昆明理工大学（计算机系）获得硕士学位
2013 年 09 月— 2019 年 06 月，在中国科学院云南天文台攻读博士学位

已发表(或正式接受)的学术论文

[1]戴伟，尚振宏，徐永华，等. 基于独立成分分析的射频干扰信号消除方法[J]. 天文研究与技术，2019

[2]Wei D，Feng W. Study On Processing Performance Of A DPDK And GPU Combined Pulsar Data Reduction System [J]. International Journal of Mechatronics and Applied Mechanics, 2019

[3]余程嵘，王威，**戴伟**(*)，邓辉，王锋，卫守林. 基于 Docker 的射电干涉阵软件系统敏捷封装与部署.天文研究与技术，2019.1.15，16(1): 123~130

[4]赖铖，梅盈，邓辉，王锋，**戴伟**. MUSER 可见度数据积分方法与实现[J]. 天文研究与技术，2018，15(01):78-86.

[5]冯勇，陈坤，邓辉，王锋，梅盈，卫守林，**戴伟**，杨秋萍，刘应波，吴静平. 基于 OpenCL 的 MUSER CLEAN 算法研究与实现[J]. 天文学报，2017，58(02):57-66.

[6]Wei Shoulin ，Wang Feng，Deng Hui，Liu Cuiyin， Liang Bo，Mei Ying，**Dai Wei**，Shi Congming，Liu Yingbo，Wu Jingping， OpenCluster: A Flexible Distributed Computing Framework for Astronomical Data Processing， Publications of the Astronomical Society of the Pacific， 2016.01.01， 129（972）：024001

[7]Wei Shoulin ，Yu Konglin，Liang Bo，Zhang Xiaoli，**Dai Wei**. A Study of Internet of Things Real-time Data Updating Based on WebSocket，6th International Conference

on Electronics and Information Engineering (ICEIE)，大连， 2015.01.01-2015.01.01

[8]Ying-bo Liu ，Feng Wang，Hui Deng，Kai-fan Ji， ，Shou-lin Wei， **Dai Wei**，Bo Liang，Xiao-li Zhang， Low-cost high performance distributed data storage for multi-channel observations， New Astronomy， 2015.10.01， 40：78~86

[9] Wang F, Mei Y, Deng H, Wang W, Liu CY, Liu DH, Wei SL,**Dai W**, Liang B, Liu YB. Distributed Data-Processing Pipeline for Mingantu Ultrawide Spectral Radioheliograph， Publications of the Astronomical Society of the Pacific，2015.01.01， 127（950）：383~396

[10]袁智，梁波，邓辉，王锋，**戴伟**，季凯帆， 一种面向选址的低功耗远程电源控制和视频监控系统设计与实现， 天文研究与技术， 2015.01.01， （01）：70~77

[11]Yingbo ，Wang Feng，Ji Kaifan，Deng Hui， ，Liang Bo，**Dai Wei**. NVST data archiving system based on fastbit nosql database， Journal of the Korean Astronomical Society， 2014.01.01， 47（3）：115~122

[12]王锋， 邓辉，**戴伟**， 卫守林， 梁波， 季凯帆. CASA 混合编程技术分析与功能扩展研究.天文研究与技术， 01 期， pp 46-53， 2014.

[13]Liu Ying-bo，Wang Feng，Ji Kai-fan，Deng Hui，**Dai Wei**，Liang Bo. Nvst data archiving system based on fastbit nosql database Journal of the Korean Astronomical Society， 47(3)， pp 115-122， 2014/6/30.

[14]卫守林，曹子皇，王锋，邓辉，梁波， **戴伟**， 基于 WebSocket 的 RTS2 Web 控制研究， 天文研究与技术， 2014.3.01， （04）：404~409

[15]刘应波，王锋，季凯帆，邓辉， **戴伟**，梁波， 基于 NoSQL 的 FITS 文件头元数据存储和查询研究， 计算机应用研究， 2014.8.01

[16]陈亚杰，梁波，王锋，邓辉， **戴伟**，曹子皇， RTS2 中新 CCD 类型扩展方法， 天文研究与技术， 2014.01.01， （03）：281~286

[17]王锋，邓辉， **戴伟**，卫守林，梁波，季凯帆， CASA 混合编程技术分析与功能扩展研究， 天文研究与技术， 2014.01.01， （01）：46~53

[18]梅盈，刘东浩，王锋，邓辉， **戴伟**，季凯帆， 中国频谱射电日像仪 FITS-IDI 文件格式研究， 天文研究与技术， 2014.10.15， （04）：388~395

[19]刘应波，王锋，季凯帆，邓辉， **戴伟**，梁波， 基于压缩-字对齐位图的天文海量数据实时索引， 计算机工程与应用， 2014.5.01， （01） ：37~41+140

受理或已获得的专利：

戴伟,冯涛,季凯帆,杨云飞,冯松,王锋,邓辉. 一种用于太阳高分辨率图像中图像配准的方法. 中国专利, 授权, 2019.2.5-2020.2.5, 2016108084339

戴伟,朱彦飞,王锋,张晓丽,罗静. 一种基于 DPDK 的天文数据采集和实时处理方法. 中国专利, 受理，申请号 201810632883 .6

攻读博士学位期间参加的科研项目

1.国家自然科学基金联合项目，U1631129，NVST 观测调度系统关键技术研究，2017.01-2019.12，50 万元，在研，参加；

2.国家自然科学基金青年项目，11403009，CCD 集群分布式采集控制虚拟化技术研究，2015.01-2017.12，26 万元，已结题，参加；

3.国家自然科学基金地区项目，11263004，基于 NoSQL 的海量太阳观测数据分布式存储技术的研究，2013.01-2016.12，64 万元，已结题，参加。

4.国家自然科学基金青年项目，11203011，面向天文选址的自主观测与数据传输技术研究，2013.01-2015.12，27 万元，已结题，参加;

5.国家自然科学基金青年项目，11103005，基于云计算的虚拟天文台关键技术研究，2012.01-2014.12，24 万元，已结题，主持；