第 44 卷 第 1 期
2026 年 3 月

天 文 学 进 展
PROGRESS IN ASTRONOMY

Vol. 44, No. 1
Mar., 2026

doi: 10.3969/j.issn.1000-8349.2026.01.10

基于 GPU 和 Spark 框架的 VLBI 相关处理架构研究

谢科屹[1,2]，张 娟[1,3,4]，童锋贤[1,4]，郑为民[1,3,4]，童 力[1,4]，刘 磊[1,4]

(1. 中国科学院 上海天文台，上海 200030； 2. 中国科学院大学 天文与空间科学学院，北京 100049； 3. 射电天文与技术全国重点实验室，北京 100101； 4. 上海市空间导航与定位技术重点实验室，上海 200030)

摘要： 甚长基线干涉测量技术（VLBI）正逐渐向高灵敏度、高时空分辨率方向发展，观测台站数量和观测带宽成倍增加，以至 VLBI 观测数据量急剧增长，给现有数据处理系统带来了严峻挑战。为满足大规模 VLBI 数据相关处理的需求，提出并实现了一种基于 GPU 与 Spark 框架的 VLBI 相关处理架构。测试结果表明，该架构具备高可扩展性与高可靠性，加速比随计算资源扩展呈近线性提升，能够高效处理大规模 VLBI 数据。这为应对未来 VLBI 观测任务中的海量数据处理需求奠定了技术基础，也为脉冲星测时阵列中信号合成所需的高速相关处理技术提供了有力支撑。

关键词： VLBI；相关处理机；GPU；分布式计算；Spark

中图分类号： P228.6 **文献标识码：** A

1 引 言

甚长基线干涉测量 (very long baseline interferometry，VLBI) 技术起源于 20 世纪 60 年代。该技术通过将不同地点的望远镜所接收到的射电信号传输至数据处理中心进行 VLBI 相关处理，可形成等效口径达到基线长度（两个望远镜间的距离）的虚拟望远镜，极大地提升了角分辨率，被广泛应用于天体物理、天体测量、航天器定位与大地测量等领域[1]。

近年来，随着脉冲星测时对微弱信号探测需求的增长，研究者通过天线信号合成技术将数台望远镜的数据进行互相关处理并叠加，可以提高脉冲星测时的探测灵敏度[2]。在天线信号合成过程中需要 VLBI 相关处理机将信号从时域转换到频域，并开展交叉相关和积分以获得合成所需的校正数据。上述操作涉及大量复杂计算，需要高速相关处理技术提供支撑。

VLBI 系统的灵敏度与天线口径及信号记录带宽等有关。为提升 VLBI 观测的灵敏度，记录设备的带宽不断提升[3]。下一代事件视界望远镜（Next-Generation Event Horizon Telescope，

收稿日期： 2025-03-20； **修回日期：** 2025-04-28

资助项目： 国家重点研发计划（2022YFC2205203）；对外技术合作科研（ZA06）；上海市空间导航与定位技术重点实验室 (22DZ2229017)；射电天文与技术全国重点实验室专项基金

通讯作者： 张娟，zhangjuan@shao.ac.cn

ngEHT）计划部署 20 个台站，单站数据记录速率预计突破 128 Gb·s^{-1}[4]；全球测地观测系统（VLBI Global Observing System，VGOS）计划扩展至 40 个台站，单站数据率达 32 Gb·s^{-1}，整个观测网络的数据记录速率也将达 1.28 Gb·s^{-1}[5]。在高灵敏度与高宽带观测需求下，ngEHT 等 VLBI 阵列的数据量预计可达每月 PB 级[4]。这对现有的 VLBI 相关处理机架构提出了考验。

VLBI 相关处理机的核心功能是对接收到的各个台站信号进行互相关运算，输出可见度数据。目前使用最广泛的软件相关处理机 DiFX（Distributed FX Software Correlator）采用 MPI（Message Passing Interface）框架构建并行计算架构实现高速数据处理[6]。其他 VLBI 软件处理机，如 SFXC（Super FX Correlator）[7]与 CVNScorr（Chinese VLBI Network Software Correlator）[8]，也均使用 MPI 并行框架。基于 MPI 框架开发的并行计算软件能够支持高性能计算，但在大规模集群计算中面临负载均衡、容错机制等复杂问题，在可扩展性方面存在局限性。

随着大数据并行编程与计算方法的快速发展，以 Apache Hadoop（以下简称 Hadoop）与 Apache Spark（以下简称 Spark）为代表的大数据处理框架，凭借良好的可扩展性、可靠性以及大规模集群上的易用性，成为了大数据的重要解决方案。其中，Spark 通过基于内存的计算模式优化了读写开销，使计算速度相较 Hadoop 得到大幅提升，被广泛应用于基因组学、机器学习与图计算中[9]。在此背景下，美国 Haystack 天文台基于 Hadoop 与 DiFX 等开源软件，开发了基于 Hadoop 框架的相关处理机 CorrelX[10]。为提高性能，CorrelX 被迁移至 Spark 框架，并重命名为 CXS [11]。作为使用分布式大数据处理框架开发相关处理机的探索，CorrelX 与 CXS 重视架构研究，为高可扩展性的相关处理机架构设计提供了新的范式，但性能相较其他相关处理机存在差距，CXS 在测试中计算速度仅为 DiFX 的 $\frac{1}{4}$[12]。

拥有高性能并行计算能力的 GPU（Graphics Processing Unit）已在部分 VLBI 相关处理机中得到应用。中国科学院上海天文台研究了 GPU 集群并行软件相关处理架构[13]。俄罗斯科学院应用天文研究所开发了相关处理机 RASFX，通过使用 GPU 实现了对 6 个台站数据的并行处理，单站处理速率达 16 Gb·s^{-1}[14]。Zhang 等人针对深空探测任务研发了一套使用 GPU 的 VLBI 相关处理机，在 NVIDIA RTX 2070 平台上的基准测试显示，其数据处理速率比 Intel Xeon E5-2683 v3（20 Threads）处理器组成的双节点集群提升 4.7 倍[15]。DiFX 开发社区也正在推进 GPU 版本的研发工作，初期测试表明：在 NVIDIA P100 平台上，与运行于 Intel Xeon Gold 6140（18 Cores）处理器的 CPU 版本相比，当前 GPU 版本在数据处理速度上仍存在 0%~20% 的差距[16]。

现有处理大数据的相关处理机速度较慢，为此，本文基于 GPU 和 Spark 框架开展了相关处理技术的研究，并基于 CXS 研发了 GPU 版的相关处理原型机。主要研究内容包括：（1）在 CXS 架构的基础上设计了使用 GPU 的 Spark 框架相关处理机架构；（2）使用 GPU 并行实现了相关处理中的算法；（3）探究并实现了 Spark 框架下的多 GPU并行计算策略。

2 基于 Spark 框架的相关处理机系统设计

Spark 框架依托分布式架构和内存计算能力，可将大规模数据集分割至海量节点并行处理。其编程模型提供了 Java、Python 等多语言的 API，便于敏捷开发和迭代。Spark 还通过记录数据转换过程的容错机制，在节点故障时可自动恢复计算，确保分布式环境下的稳定性。因此，Spark 为 VLBI 相关处理机系统设计提供了一个兼具可扩展性、可靠性和易用性的分布式计算解决方案。

2.1 VLBI 相关处理主要步骤

VLBI 相关处理的主要计算步骤包括：数据解码、整数时延补偿、条纹旋转、快速傅里叶变换（fast Fourier fransform, FFT）、分数时延补偿以及共轭相乘与积分。

各步骤分别将 VLBI 记录设备所采集的原始数据解码；使用预报时延模型补偿整数位时延；对数据进行逐点的相位补偿，以消除对基频信号进行时延补偿而引入的相位误差；将补偿后的数据从时域变换为频域；在频域中矫正残余的非整数时延；对来自不同望远镜的数据，开展两两共轭相乘并积分，得到目标源的可见度函数[17]。

2.2 Spark 框架下的相关处理架构

基于 VLBI 相关处理的主要步骤，CXS 中将相关处理分为数据读取、预处理、数据分发、相关处理与结果存储五个阶段。本文沿用这五个阶段的设计，并优化了阶段内的任务设计及实现。其中，预处理与相关处理阶段涵盖了相关处理的主要计算任务。预处理阶段的计算复杂度为 $O(1)$，使用 CPU 执行；相关处理阶段为计算密集型任务，本文利用 GPU 并行计算该阶段任务。数据读取、数据分发与结果存储阶段依托 Spark 提供的 I/O 接口实现。系统流程如图 1 所示。

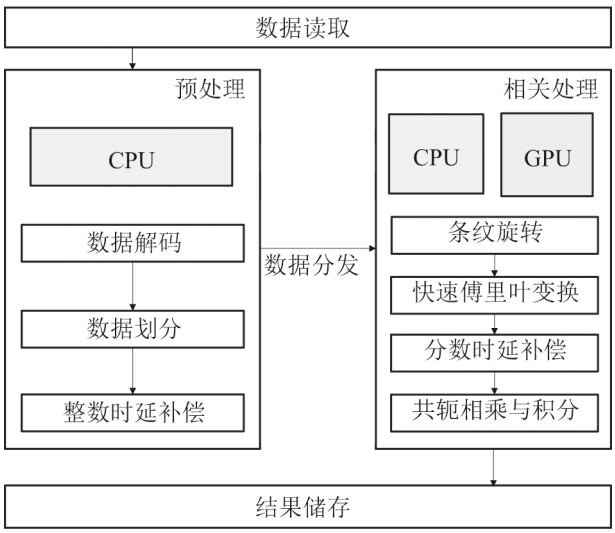


图 1 使用 Spark 框架的 VLBI 相关处理流程图

数据读取阶段使用 Spark 提供的 I/O 接口。在读入数据时即对每个数据进行分区及并行读取，各数据分区数量与数据块数量相同。

在预处理阶段，系统利用 CPU 集群对数据块进行并行处理，完成数据解码、数据划分以及整数时延补偿。在执行完数据解码后，数据被以时间和通道为基本单位，细分为粒度更小的数据块；在此基础上，使用预报时延模型对这些数据块执行整数时延补偿。在 CXS 中，进行整数时延补偿后还需将数据重新映射为比特流，本文通过在预处理阶段使用比特位运算，从而避免了重新映射时的资源消耗。

在数据分发阶段，补偿完成的数据被以时间和通道为特征分组并分发给相应计算节点。同一积分时间与同一通道的各台站数据被分配至同一数据块内，再将数据块分发至不同计算节点（如图 2 所示），以确保相关处理阶段中各节点仅使用局部数据运行，避免跨节点的数据依赖。

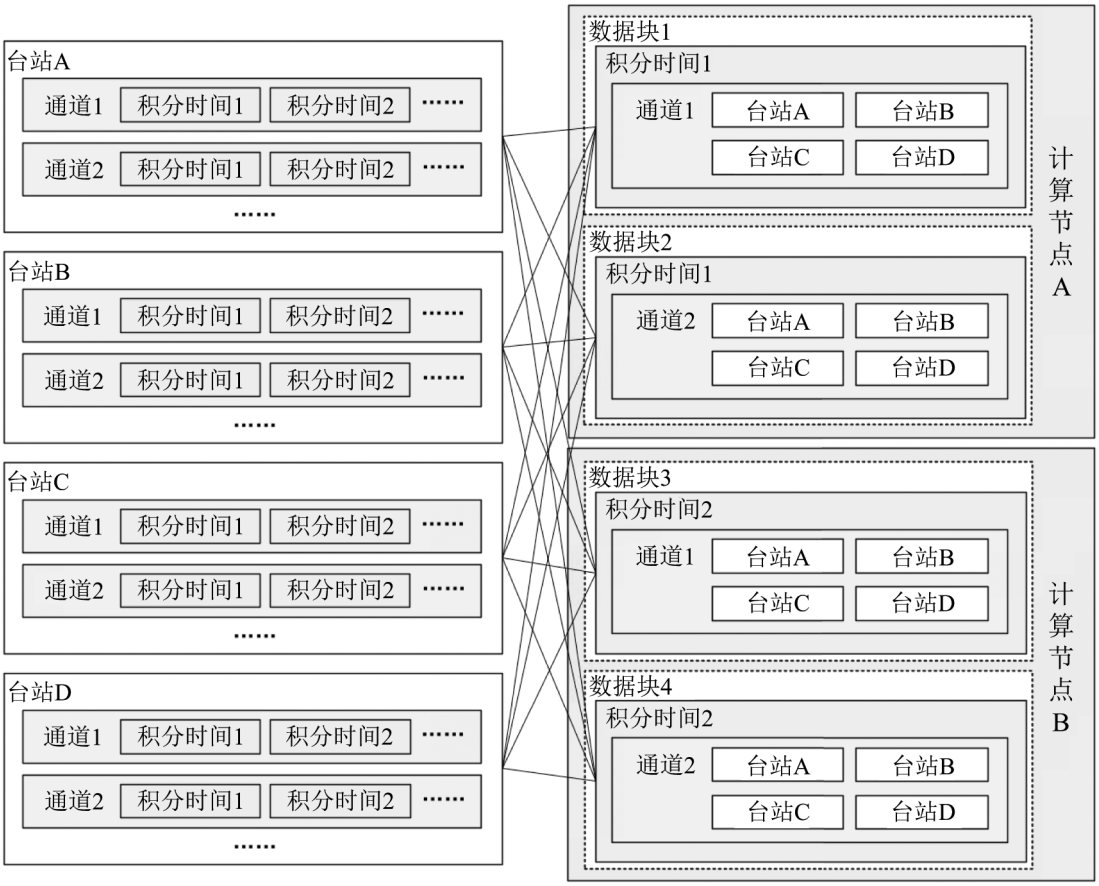


图 2　预处理与相关处理阶段间的数据分发示意图

相关处理阶段中的条纹旋转、快速傅里叶变换、分数时延补偿以及共轭相乘与积分等计算密集型任务利用 GPU 的高并行计算能力执行，以提升处理效率。在各计算节点完成任务

后，通过 Spark 内置的聚合函数将分散的计算结果汇总至单一节点，并将最终的可见度函数写入磁盘保存，得到完整的相关处理结果。

3 基于 GPU 的相关处理

VLBI 相关处理中的计算在不同积分时间相互独立，易于实现并行化。GPU 因配备大量浮点计算单元，具备出色的浮点并行计算能力，特别适用于 VLBI 相关处理中涉及的大量浮点运算。但 GPU 的单个计算单元规模较小，其指令控制模块大幅精简，难以胜任逻辑复杂的任务。基于 CPU 与 GPU 的计算特性及 VLBI 相关处理中计算的特点，本文设计了一套基于 GPU 的 VLBI 数据相关处理方案。该方案使用 CuPy 库与 cuFFT 库，将相关处理中条纹旋转、快速傅里叶变换、分数时延补偿及共轭相乘等计算密集型任务迁移至 GPU 上进行并行加速，以实现 VLBI 数据的高效处理（如图 3 所示）。

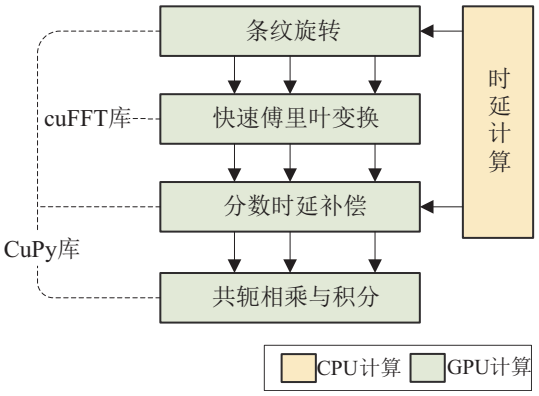


图 3 GPU 与 CPU 执行相关处理步骤示意图

3.1 GPU 并行算法的实现

对于 VLBI 相关处理算法在 GPU 上的实现，首先需解决的是 CPU 与 GPU 间的数据传输问题。GPU 的计算作业依赖其高速专用内存（下称显存）来存储和访问数据。因此，处理 VLBI 数据时，需先将数据从内存传输至显存，以确保 GPU 能够高效执行运算任务。通常相关处理机在预处理后，数据被映射为浮点数。浮点数会增加内存至显存的传输开销，本文将数据传输至显存后，再于 GPU 上完成原始数据至浮点数的映射。

在 VLBI 数据传输至显存且完成浮点数的映射后，依次执行以下相关处理步骤。

（1）条纹旋转

在 CPU 中条纹旋转步骤通常采取逐个数据点独立计算，可利用 GPU 的并行特性同时计算各点数据。时延补偿值由 CPU 计算，再将结果传输至显存。考虑到条纹旋转中各数据点的时延补偿值变化通常较小，可仅计算部分数据点的时延补偿值并求取变化率，通过传递时延模型周期内首位时延与变化率的方式减少内存至显存间的数值传递，降低 I/O 开销。

（2）快速傅里叶变换

数据经条纹旋转处理后，利用 cuFFT 库执行快速傅里叶变换，将其从时域转换至频域。

（3）分数时延补偿

数据被转换至频域后可进行分数时延补偿。分数时延补偿中所需时延补偿值由 CPU 计算并传输至显存中，利用 GPU 并行计算生成复指数形式的分数时延补偿因子，再与频域数据相乘得到将分数时延补偿后的数据。

（4）共轭相乘与积分

对来自各测站的观测数据进行共轭相乘与积分，计算台站的自功率谱和台站间的互功率谱。由于共轭相乘时各数据点间相互独立，因而适应 GPU 上的并行计算。

上述步骤中，条纹旋转、分数时延补偿中相位补偿值的计算、分数时延补偿操作及共轭相乘与积分，共通过四个 GPU 核函数执行，需调用多次 GPU 计算以完成完整的相关处理流程。

3.2 Spark 框架下的多 GPU 调度策略

Spark 框架具备可扩展性与可靠性，但未提供对 GPU 的原生支持，未具备对 GPU 资源的调度策略。若要在 Spark 框架下使用多 GPU 并保持备高可扩展性，需要设计对多 GPU 资源的调度策略，以保障并行计算的负载均衡。

Spark 框架的并行计算采用 Worker-Executor-Task 三层架构（如图 4 所示）。其中，Worker 为物理工作节点；Executor 是资源分配的基本单位，运行于 Worker 节点，拥有线程数（Core）与内存等资源；Task 是任务执行的最小单元，每个 Task 对应一个数据块的计算。Core 指 Executor 内可并行执行任务的线程，通常直接映射至物理 CPU 线程上。本文中每个 Task 使用一个 Core。系统整体的并发度由 Executor 数量与单个 Executor 所含 Core 数量（CPU 线程数）的乘积决定。通常 Spark 框架在启动后只接受使用一种计算资源的调度策略。但鉴于 CPU 与 GPU 的异构特性，在使用 GPU 计算时系统并发度无法以不同于 GPU 数量的 CPU 线程数决定，系统需在数据分发阶段加入 GPU 资源调整 Task 的计算并发度，以适配不同的计算资源。

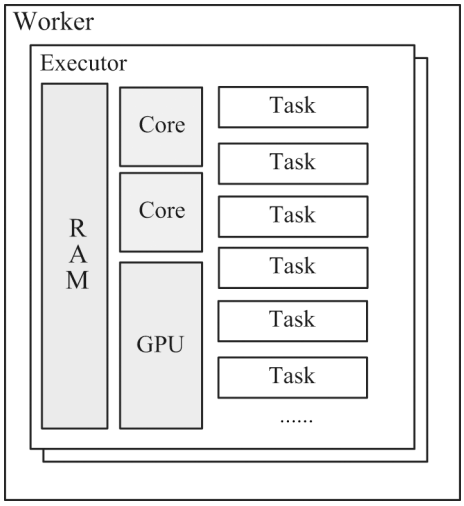


图 4　Spark 框架的并行计算层级示意图

针对 GPU 间数据传输效率低、I/O 开销大的问题，本方案采用 Executor 与 GPU 绑定的设计：每个 Executor 均独占物理 GPU，且不同 Executor 内 Task 执行保持相互独立，从而避免 GPU 间的数据传输。图 5 展示了预处理与相关处理阶段内 Task 所使用的计算资源差异。

在预处理阶段，系统根据应用程序参数和平台配置，每个 Worker 启动多个 Executor，各 Executor 利用含有的 Core 资源并行执行多个 Task。

在相关处理阶段，每个 Executor 绑定独立物理 GPU，Worker 节点通过运行多个 Executor 实现对多 GPU 的并行利用。此时可通过设定每个 Task 所需的 GPU 资源决定 Executor 内 Task 运行的并行度，而非由预处理阶段中的 Core 数量决定。这种动态调整机制使得系统整体的并发度与可用 GPU 数量间相匹配。

不同计算阶段内 Task 的差异化分配策略可在 Spark 框架下实现跨节点、跨 GPU 的并行处理。在预处理阶段基于 Core 数量并发执行，在相关处理阶段根据 GPU 数量与 Task 需求调

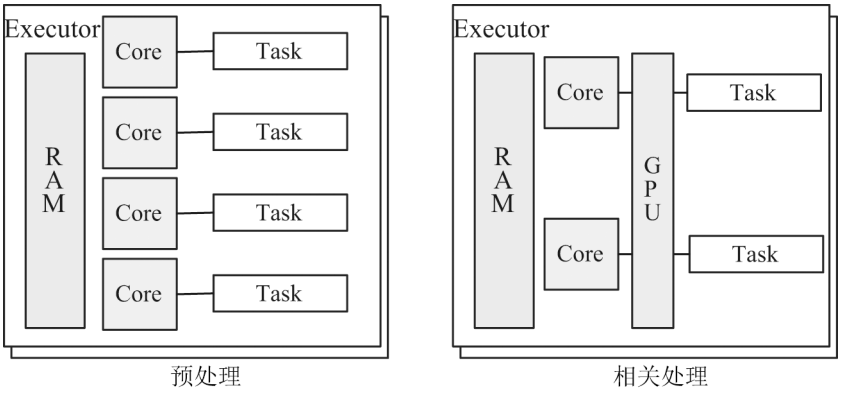


图 5 预处理与相关处理阶段计算资源与 Task 的映射关系

整调度策略，最终达成异构计算资源的高效协同处理。

4 测试结果与分析

基于本文提出的 VLBI 相关处理系统，我们设计研制了相关处理原型机，并开展了对各步骤计算速度、结果正确性、完整流程计算速度以及不同计算资源下加速比的测试。

4.1 测试样例介绍

测试数据集 1 为 VLBI 宽带观测仿真数据，使用与他人工作相同的数据集参数[3]，数据量为 80 GB 。测试数据集 2 为 EVN（European VLBI Network）于 2019 年对脉冲星的一次观测（EL060）中的部分数据[18]，数据量达 600 GB。数据集 1 与数据集 2 的参数见表 1。

表 1 测试数据集参数

参数	数据集 1	数据集 2
台站数量	2	6
观测时长	20 s	900 s
采样率	4 GHz	16 MHz
量化精度	2 bit	2 bit
数据通道数	2（双极化）	16（双极化）

4.2 测试平台

测试平台为商用集群，该集群含 2 个计算节点，总计拥有 104 个 CPU 核心、4 个 GPU 与1004 GB 的内存。在该集群上部署了基于 GPU 和 Spark 框架的相关处理原型机与 CXS 的最新版本 CXS3311。集群单节点配置与环境参数见表 2。

4.3 模块化测试

由于 GPU 采用异步执行架构，其计算任务通过 CUDA 流（CUDAStream）进行调度，计算指令的提交与执行过程间相互独立。分析各模块的计算开销时，在 CUDA 流中设置同步事件，以确保准确记录相关处理过程中不同模块的时间消耗。分别使用单 GPU 与单 CPU 核执行测试，并比较 CXS 在相同计算步骤下的耗时，结果如图 6 所示，可见 GPU 在执行相关处理的各个子模块（条纹旋转、快速傅里叶变换、分数时延补偿以及共轭相乘）时均表现出显著的性能优势。

表 2　计算节点参数

参数	值	数量
CPU	Intel Xeon Gold 5320 (Ice Lake, 2.2 GHz with 26 Cores)	2
GPU	NVIDIA A30 (Ampere, GA100 with 56 SM)	2
RAM	DDR 4	502 GB
OS	Linux 4.18.0 (Rocky 8.6)	—
CUDA Version	11.8	—
Spark Version	3.5.2	—

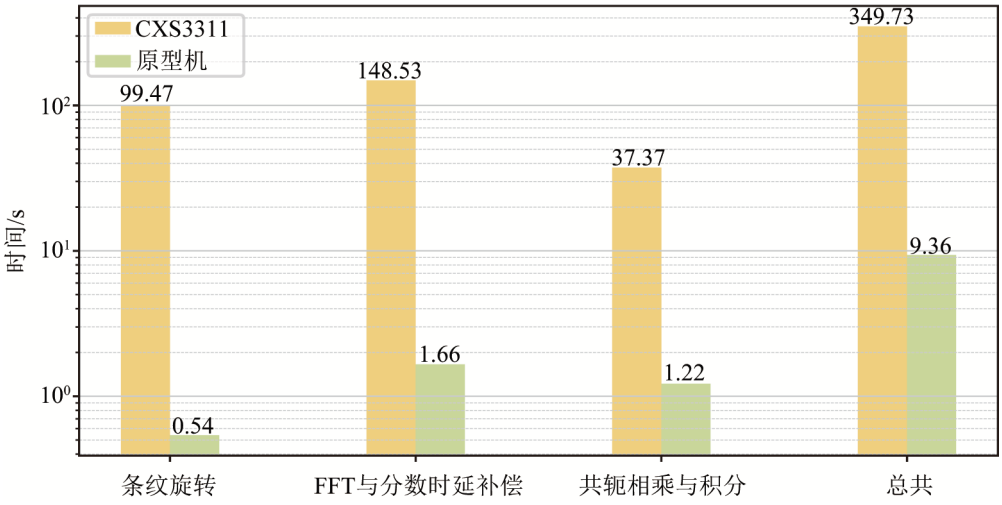


图 6　CXS 与原型机在相关处理阶段关键步骤上耗时对比

表 3 给出了各模块的计算速度比，结果显示 GPU 在执行相关处理各模块时展现出显著的性能提升，但完整相关处理过程的速度比未达到各子模块速度比的时间加权平均。测试中各模块的时间消耗占比如图 7 所示，可见大部分时间仍被内存与显存之间的数据传输导致的 I/O 开销占据。

表 3　原型机相较 CXS 的计算速度比

步骤	速度比
条纹旋转	约 190
FFT 与分数时延补偿	约 90
共轭相乘与积分	约 30
综合 ①	约 37

注：① 综合指完整相关处理流程，包括数据传输等非计算过程中的时间消耗

4.4 完整流程测试

4.4.1 正确性测试

使用数据集 2 进行测试，图 8 为本相关处理原型机与 CXS 处理同一数据的计算结果相位谱。比较两个计算结果，图 9 展示了各数据点间的差异，两组数据的均方根误差 (RMSE) 仅为 3.11×10^{-15} rad，原型机的计算结果与 CXS 基本一致。

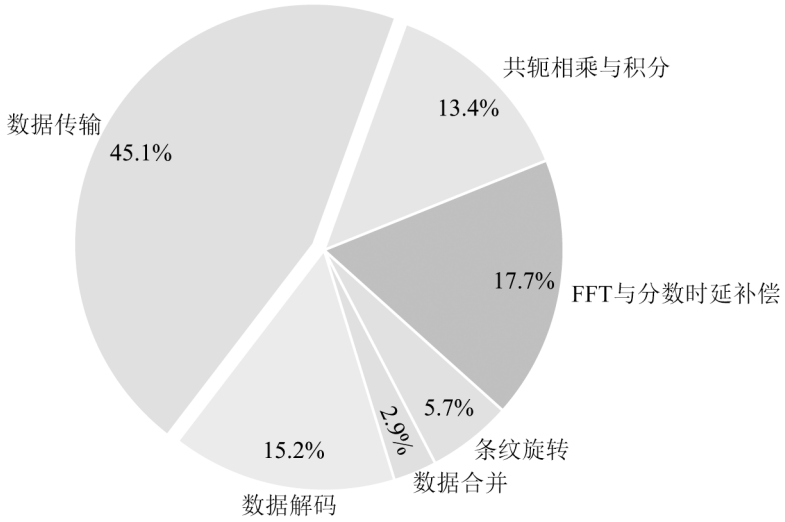
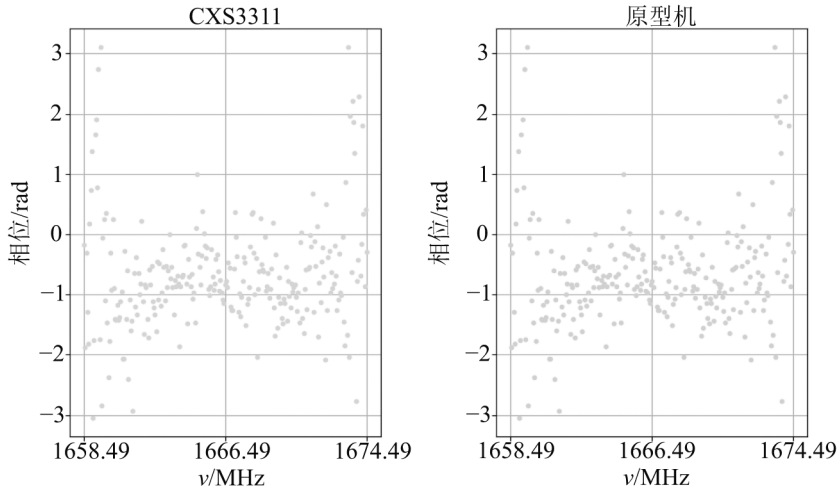


图 7　相关处理阶段中各步骤的耗时占比



注：使用数据集 2（EL060），针对射电源 J1800+3848 ，采用 LL 极化、Mc-Sr 基线和 10 s 积分时间。

图 8　a) 为 CXS 的计算结果相位谱；b) 为本原型机的计算结果相位谱

4.4.2 速度比对

在 GCP（Google Cloud Platform）服务器上部署的 DiFX[3] 与本测试平台部署的 CXS 作为速度基准，比较三者对数据集 1 的相关处理速度，结果可见表 4。如图 10 所示，配备 4 块 NVIDIA A30 GPU 的原型机相关处理计算速度是 40 核 Intel Xeon Gold 5320 处理器上运行的 CXS 的 4 倍左右，是 96 核 Intel Xeon 云服务器上运行的 DiFX 计算速度的 1.1 倍左右。

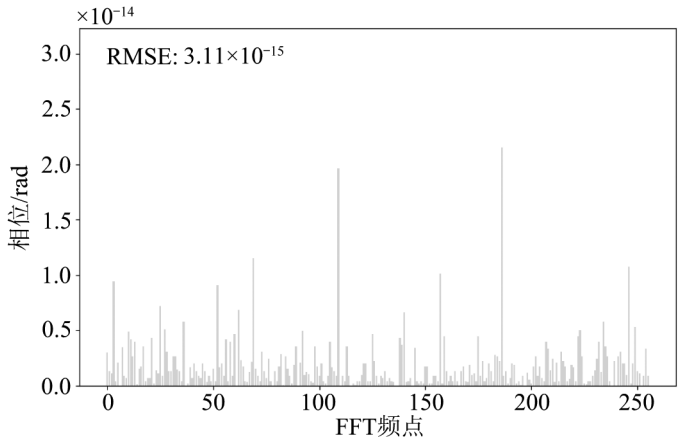
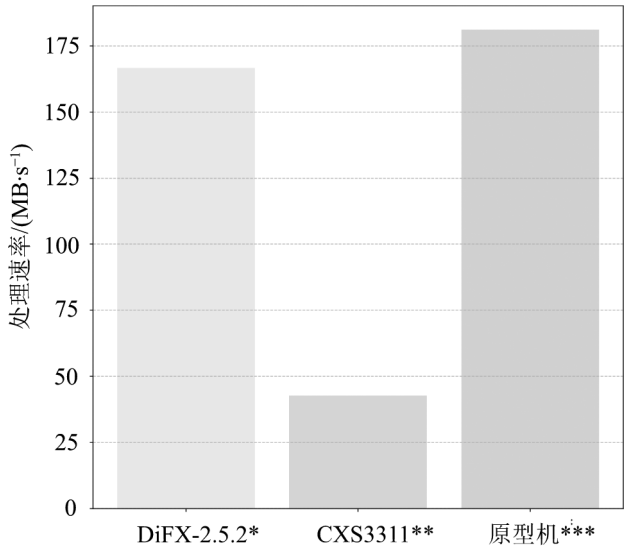


图 9　CXS 与原型机的计算结果相位谱差异

表 4　原型机与 DiFX 及 CXS 的计算速度

名称	主要计算资源	规模	处理速度
DiFX-2.5.2	Intel Xeon (n1-highmem-96)	96 vCPU	166.7 MB/s[3]
CXS3311	Intel Xeon Gold 5320	40 Core	42.4 MB/s
原型机	NVIDIA A30	4 GPU	181.2 MB/s



注：* 运行于 Google Cloud Platform (GCP) 上的 Intel Xeon 处理器 (共 96 核)；** 运行于 Intel Xeon Gold 5320 处理器 (共 40 核)；*** 运行于 NVIDIA A30 GPU × 4。

图 10　原型机与 DiFX 及 CXS 的计算速度对比

4.4.3 加速比测试

通过调整计算资源，在双节点集群中分别使用 10、20 和 40 个 CPU 核以及 1、2 和 4 个 GPU，利用数据集 1 进行了加速比测试，结果如图 11 所示。可见采用 Spark 框架的相关处理机在不同计算资源下均表现出了良好的近线性加速比，且相较于使用 CPU 的 CXS，采用 GPU 的原型机在加速比上也具有一定优势。

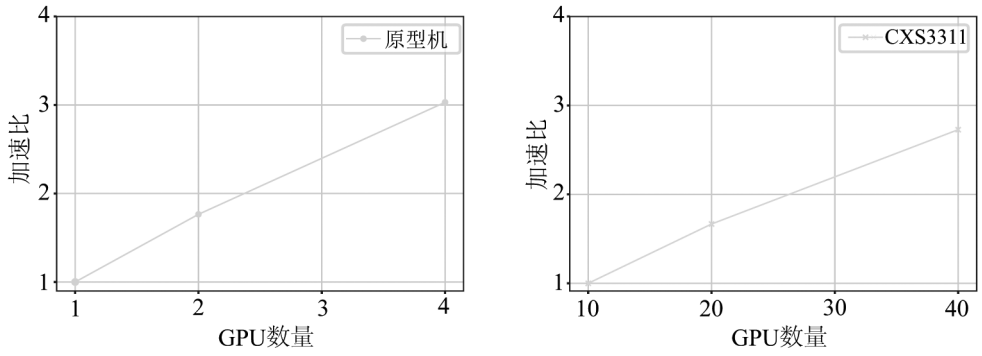


图 11 采用 GPU 的原型机及 CXS 的加速比

4.5 测试结果分析

如表 5 所示两张 NVIDIA A30 GPU 的性能与单张 NVIDIA A100 GPU 相当。对比表 4 测试结果，并参考 GCP 上的 A100 实例价格，可见 CXS 需要使用约 1.4 倍的成本才可达到本原型机相当的性能；而 DiFX 在使用成本约为 A100 实例 65% 的情况下即可实现本原型机测试性能的 90%。

综上所述，原型机在 GPU 支持下计算速度优于 CXS。受益于 Spark 框架的高可扩展性，在测试中该原型机呈现了近线性的加速比。通过成本分析，原型机在计算成本上优于 CXS，逊于发展成熟的 DiFX。通常而言，DiFX 具备完全优化后的计算速度和更通用的计算环境需求，能够适应大多数相关处理任务。另一方面，因为 Spark 框架具有良好的可扩展性以及包括故障恢复在内的容错机制，所以本原型机也具有近线性的加速比与高可靠性，更适用于大规模集群的部署或对可靠性要求较高的场景。

表 5 GPU 加速卡计算性能对比

名称	CUDA 数	单精度浮点性能 ①	显存大小/GB	显存带宽/GB·s^{-1}
NVIDIA A30[19]	3 584	10.3 TFLOPS	20	933
NVIDIA A100[20]	6 912	19.5 TFLOPS	40	1 555

注：① 为 GPU 每秒执行的单精度浮点运算数

5 结　论

本文提出并实现了一种基于 GPU 和 Spark 框架的 VLBI 相关处理架构，在 CXS 的基础上使用 GPU 并行计算技术，提高了 Spark 框架相关处理机的计算速度，并改进了 Spark 框架下的多 GPU 的调度策略，实现了对 VLBI 宽带观测数据的高速处理。测试表明，基于该架构的相关处理原型机在处理速度上比 CXS 大幅提升，加速比随计算资源扩展呈近线性增长，可满足高可扩展性与高可靠性的相关处理计算，以应对未来 VLBI 观测任务以及脉冲星测时阵列中信号合成的海量数据处理需求。

参考文献：

[1] 陈中, 郑为民. 天文学进展, 2015, 33: 489

[2] Bassa C G, Janssen G H, Karuppusamy R, et al. MNRAS, 2016, 456: 2196

[3] Gill A, Blackburn L, Roshanineshat A, et al. PASP, 2019, 131: 124501

[4] Doeleman S S, Barrett J, Blackburn L, et al. Galaxies, 2023, 11(5): 107

[5] Vázquez A J, Elosegui P, Lonsdale C J, et al. PASP, 2022, 134: 104501

[6] Deller A T, Tingay S J, Bailes M, et al. PASP, 2007, 119: 318

[7] Keimpema A, Kettenis M M, Pogrebenko S V, et al. Experimental Astronomy, 2015, 39: 259

[8] Zheng W, Zhang J, Wang G, et al. IVS 2018 General Meeting Proceedings. Longyearbyen: Norwegian Mapping Authority (Kartverket), 2019: 7

[9] 张登科. 硕士论文, 西安: 西安电子科技大学, 2022

[10] Pankratius V, Vazquez A J, Elosegui P. The 5th International VLBI Technology Workshop. Westford: MIT Haystack Observatory, 2016: 1

[11] Vázquez Álvarez A J. Master's thesis, Madrid: Universidad Nacional de Educación a Distancia, 2021: 1

[12] Álvarez A J V. https://github.com/ajvazquez/CXS, 2021

[13] 陈中. 博士论文, 北京: 中国科学院大学, 2015

[14] Surkis I, Ken V O, Kurdubova Y L, et al. The 13th EVN Symposium & Users Meeting Proceedings. St. Petersburg: IAA RAS, 2017: 123

[15] Zhang F, Zhao C, Han S, et al. Remote Sensing, 2021, 13: 1226

[16] Deller A. The 16th DiFX Users and Developers Meeting. Vienna: Vienna Center for VLBI, 2024

[17] 郑为民, 张娟, 徐志骏, 等. 深空探测学报 (中英文）, 2020, 7: 354

[18] Liu L, Xu Z, Yan Z, et al. The Astronomical Journal, 2021, 162: 159

[19] Corp. N. https://www.nvidia.com/en-us/data-center/products/a30-gpu/, 2021

[20] Corp. N. https://www.nvidia.com/en-us/data-center/a100/, 2020

Research on VLBI Correlator Architecture Based on GPU and Spark Framework

XIE Keyi[1,2], ZHANG Juan[1,3,4], TONG Fengxian[1,4], ZHENG Weimin[1,3,4], TONG Li[1,4], LIU Lei[1,4]

(1. Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030, China; 2. School of Astronomy and Space Science, University of Chinese Academy of Sciences, Beijing 100049, China; 3. State Key Laboratory of Radio Astronomy and Technology, Beijing 100101, China; 4. Shanghai Key Laboratory of Space Navigation and Positioning Techniques, Shanghai 200030, China)

Abstract: Very Long Baseline Interferometry (VLBI) is progressively advancing towards higher sensitivity and spatiotemporal resolution. This evolution has led to a substantial increase in the number of observing stations and bandwidth. This has resulted in a dramatic surge in the volume of VLBI observation data, posing significant challenges to existing data processing systems. To address the requirements for large-scale VLBI correlation processing, this paper proposes and implements a VLBI correlator architecture based on GPU and Spark framework. The test results show that the proposed architecture exhibits high scalability and reliability, with the speedup increasing near-linearly with the expansion of computational resources, thus enabling the efficient processing of large-scale VLBI data. This work provides a solid foundation for addressing the massive data processing demands in future VLBI observation missions and provides robust support for the high-speed correlation technology required in pulsar timing array signal synthesis.

Key words: VLBI; correlator; GPU; distributed computation; Spark