中国科学 : 物理学 力学 天文学 2023 年 第 53 卷 第 2 期 : 229503
SCIENTIA SINICA Physica, Mechanica & Astronomica physcn.scichina.com
《中国科学》杂志社 SCIENCE CHINA PRESS
论 文 SKA区域中心数据处理系统和科学应用专题

SKA低频成像管线并行优化

韦建文[1], 张晨飞[1], 劳保强[2,3*], 林新华[1], 安涛[2]

1. 上海交通大学网络信息中心, 上海 200240;
2. 中国科学院上海天文台, 上海 200030;
3. 云南大学物理与天文学院, 昆明 650500
*联系人, E-mail: lbq@shao.ac.cn

收稿日期: 2022-06-29; 接受日期: 2022-08-24; 网络出版日期: 2023-1-12
国家重点基础研究发展计划(编号: 2018YFA0404600, 2018YFA0404603)和中国科学院(编号: 114231KYSB20170003)资助项目

摘要 平方公里阵列(Square Kilometre Array, SKA)射电望远镜的数据处理是通过管线方式进行的, 管线的执行效率是SKA区域中心考虑的重要因素. 连续谱成像观测是SKA的主要观测模式之一, 也是许多科学工作的基础. 本文以SKA低频先导设备(Murchison Widefield Array, MWA)的成像管线为例, 在中国SKA区域中心原型机(China SKA Regional Centre Prototype, CSRC-P)上进行并行处理管线优化. 以往的优化方案都集中在少数性能热点, 缺乏对整体管线的系统优化, 导致整体加速比相对较低. 针对这一问题, 本文提出一种全局优化方案, 针对管线使用多种编程语言和图像数据可独立处理的特点, 综合使用C++多线程、Python多进程和Shell多任务并行等优化方法, 并验证了优化结果的准确性. 实验表明, 优化后的代码在CSRC-P的x86节点和ARM (Advanced RISC Machine)节点上分别获得了2.7和2.4倍加速, 运行时间分别从7479和9666 s, 降低为2759和4061 s. ARM计算节点展现出对SKA应用良好的适应性. 本文的优化策略和方法也适用于其他SKA科学应用, 对SKA先导望远镜的科学运行和未来的运行也有帮助.

关键词 平方公里阵列, 低频成像, 高性能计算, 并行优化

PACS: 07.05.-t, 07.05.Bx, 07.05.Kf, 95.85.-e, 95.85.Bh

1 引言

为了支持不同的科学目标, 平方公里阵列(Square Kilometre Array, SKA)射电望远镜天文台将以不同的观测模式进行, 即成像模式和非成像模式. 其中, 连续谱成像和谱线成像是主要的两种成像模式. 在连续谱成像中, 天空区域可以在很宽的频率带宽上成像. 以该模式进行的射电巡天(即连续谱巡天)观测是SKA实现星系演化、宇宙大尺度结构演化和宇宙磁场等关键科学目标的重要前提[1], 并将提供不同的科学数据产品, 包括连续谱和偏振图像[2–4]、多波段星表[5,6]、慢速瞬变体搜索数据[7]和电离层测量[8]等.

在过去的几十年中, 研究人员在射电望远镜上进行了许多大型的连续谱观测, 例如, 20 cm暗弱图像射

引用格式: 韦建文, 张晨飞, 劳保强, 等. SKA低频成像管线并行优化. 中国科学: 物理学 力学 天文学, 2023, 53: 229503
Wei J W, Zhang C F, Lao B Q, et al. Optimization of parallel processing of the Square Kilometre Array low-frequency imaging pipeline (in Chinese). Sci Sin-Phys Mech Astron, 2023, 53: 229503, doi: 10.1360/SSPMA-2022-0262

电巡天(Faint Images of the Radio Sky at Twenty Centimeters survey, FIRST)[9], 国家射电天文台甚大阵列巡天(NRAO VLA Sky Survey, NVSS)[10]和超大阵列巡天(Karl G. Jansky Very Large Array Sky Survey, VLASS)[11], 默奇森宽视场阵列(Murchison Wide-field Array, MWA)的银河系及河外全天巡天(GaLactic and Extragalactic All-sky MWA, GLEAM)[12]和MWA拓展阵的银河系及河外全天巡天观测(GaLactic and Extragalactic All-Sky MWA-eXtended, GLEAM-X; Natasha Hurley-Walker, 2022), 巨米波射电望远镜(Giant Metrewave Radio Telescope, GMRT) 150 MHz TIFR GMRT巡天(TIFR GMRT Sky Survey, TGSS)[13], 低频阵列(Low Frequency Array, LOFAR)的2 m巡天(LOFAR Two-metre Sky Survey, LoTSS)[14], MeerKAT国际GHz银河系外巡天(MeerKAT International GHz Tiered Extragalactic Exploration, MIGHTEE)观测[15], 澳大利亚平方公里阵列探路者(Australian Square Kilometre Array Pathfinder, ASKAP)的宇宙演化图谱巡天(Evolutionary Map of the Universe, EMU)[16]等. 这些大型射电望远镜观测极大地促进了对射电天空的深入了解, 为其他射电天文观测项目奠定了基础, 促进了射电天文学的蓬勃发展.

作为SKA先导望远镜之一, MWA开展了低频射电连续谱巡天观测, 即GLEAM项目. GLEAM是迄今为止低频射电波段的最大规模观测项目之一, 覆盖了赤纬+30°以南的整个南部天区. GLEAM的观测区域和频率范围与SKA低频阵列高度一致, 对未来SKA科学研究和数据处理有极高的参考意义. 除了建立观测星表[17], GLEAM还为广泛的科学研究提供基础观测数据, 包括射电星系与活动星系核[18]、星系团[19], 麦哲伦云[20]、漫射银河发射[21]、银河系的磁场[3,22]、银河系与银河系谱线[23]、脉冲星[24]和宇宙射线.

GLEAM第一年(2013年8月9日至2014年6月18日)和第二年(2014年8月4日至2015年7月6日)的观测数据已经发布. 第一年的数据包括: 在72–231 MHz的频率范围内, 对赤纬+30°以南和银河纬度10°以上的天空区域进行的观测, 其中不包含一些复杂辐射的区域, 如麦哲伦云. 第一年数据共覆盖约24800 deg^{2}的天区, 按赤纬划分为7个条带, 每个条带观测一晚, 循环完成, 直至预定的天区全部扫描完毕. 经校准等处理后, 第一年所有快照观测的可见度(Visibility)数据总量约为300 TB.

为协助SKA数据处理及相关科研成果输出, 上海天文台推动中国SKA区域中心(China SKA Regional Centre, CSRC)的筹建, 并完成了中国SRC原型机(CSRC Prototype, CSRC-P)[25,26]及其软件系统[27]的搭建和部署. 原型机前瞻地采用了x86和ARM (Advanced RISC Machine, ARM)两种指令集架构的中央处理器, 两种处理器计算能力各占一半, 其主要技术参数对比如表1所示. 在成熟的x86处理器之外加入ARM处理器的主要理由是: (1) ARM处理器起源于优化功耗的嵌入式场景, 具有更低的功耗, 内存带宽也优于同代x86产品; (2) 国内厂商有能力设计和制造ARM处理器, 基本具备构建自主可控计算集群的条件; (3) ARM具有仅次于x86的软件生态环境, 原有操作系统、计算软件和流程代码通常无需修改, 只需重新编译就能运行. 文献[28]在ARM超算系统上评估了计算流体和多体模拟等应用的性能, 结果接近甚至优于同代x86处理器. 如表1所示, x86节点在双精度浮点性能上占优, ARM节点在内存带宽上占优. 哪一种处理器在SKA成像管线应用上更有优势, 需要进一步分析测试.

SKA区域数据中心建成后, 将以MWA为基础, 同时开拓与ASKAP, MeerKAT和LOFAR的合作, 开展SKA先导项目的科学预研究和技术发展, 其中包括低频射电脉冲星搜索研究[29]以及本文所研究的低频射电连续谱巡天观测项目. 目前, 利用MWA GLEAM观测项目的原始数据, CSRC-P已用于开展数据传输、存储、管线处理等关键技术的研究.

SKA先导项目采用了大视场、宽频带、高分辨率和多波束技术, 且观测持续很长时间, 因此会产生海量观测数据, 对这些数据的处理将是一个巨大挑战[30]. 一个典型的观测任务完成后将生成PB级的原始数据和数百TB级的图像数据. 因此, 使用管线系统来自动化处理SKA和MWA的数据. 通过将流程分解为多个步骤, 将不同步骤的操作重叠, 管线可以实现并行处理,

表 1 CSRC-P中的x86和ARM计算节点

Table 1 x86 and ARM compute nodes in CSRC-P

特性	x86节点	ARM节点
处理器型号	2 × Xeon Gold 6132	2 × Kunpeng 920
厂商	Intel	华为
处理器核心	2×14核心@2.6 GHz	2×48核心@2.5 GHz
内存通道	2×6通道DDR4-2666	2×8通道DDR4-2666
内存带宽	119.2 GB/s	170.6 GB/s
浮点性能	1536 Gflops	384 Gflops

加快运行速度.

然而, 现有的成像管线效率低下, 无法充分利用超级计算机计算节点上的多核特性. 现有的优化方法只针对几个性能热点, 缺乏对整个系统的优化, 这导致相对较低的整体加速比. 我们需要在MWA现有管线的基础上探索更有效的优化方法, 以便高效地处理未来SKA的观测数据. 本文提出一种针对MWA成像管线在x86和ARM处理器上的全局优化方法, 并重点关注优化后源匹配的准确性和完整度, 主要贡献如下:

(1) 将成像管线作为整体分析, 综合采用C++多线程并行、Python多进程并行和Shell多任务并行, 进一步提高了当前并行代码的运行效率, 在x86和ARM平台上分别获得了2.7和2.4倍的加速.

(2) 在x86和ARM平台验证了并行优化的可行性, 并验证了成像结果的准确性. 结果表明, 本文的并行优化方法适用于x86和ARM两种指令集不同的计算架构, 优化前后的结果差异在实验允许范围内, 满足SKA数据分析的精度要求.

(3) 从实际加速比与理想加速比差距入手, 分析管线可并行任务数与实际加速比的关系, 指出下一步的优化方向, 即单个MWA管线仍然无法发挥现代多核处理器的能力, 需要增加工作流协调机制, 同时处理多条管线.

本文其余部分安排如下: 第2节在介绍MWA数据处理成像管线的基础上, 寻找程序热点, 有针对性地设计并行优化方法. 第3节讨论并行优化效果, 并对比优化前后的结果差异. 第4节介绍本领域的相关工作. 最后, 第5节概括我们的工作, 并展望了将来的优化方向.

2 管线并行优化方法设计

2.1 低频成像管线

SKA成像管线包含了对于MWA GLEAM数据处理的全过程[17,31]. 图1展示了SKA低频观测数据成像的流程图. 它包含5个阶段预处理(Pre-processing)、建模(Modeling)、校准(Calibration)、成像(Deep imaging)和后处理(Post-processing).

在预处理阶段, 管线调用Cotter软件[32]读入观测的快照原始数据, 搜索和去除数据中的射频干扰, 同时在时间和频率上对可见度数据做平均化处理, 使其时间分辨率为4 s, 频率分辨率为40 kHz. 最后产生通用天文软件应用程序(Common Astronomy Software Applications, CASA)的Measurement Set (MS)格式数据文件, 并使用Dysco[33]压缩数据.

在建模阶段, 管线比对星表数据库数据, 搜索在数据预处理阶段中已经去除射频干扰的图像的射电源, 制作出可被校准阶段处理的天空模型.

在校准阶段, 管线使用MitchCal算法[34], 根据建模阶段得到的天空模型, 校准来自预处理阶段的数据. 校准后的数据将会被作为新的数据列(通常为"CORRECTED_DATA"列)存储在MS文件中.

在成像阶段将调用大视场成像与反卷积软件WCSClean中的wsclean[35]命令对校准后的数据进行深度成像并获得快照图像, 主要参数见表2.

在后处理阶段, 管线运行fits_warp脚本[36], 对斯托克斯Stokes XX和YY子带图像以及Stokes I子带图像进行流量密度标度和天体测量修正. 最后在快照图像上运行射电源搜索程序Aegean[37], 将得到的源清单与GLEAM第1年公开的星表比对源的流量密度, 得到

表 2 成像阶段使用的参数

Table 2 Parameters in deep imaging phase

参数	取值
输出图像像素大小	4000×4000
最大洁化迭代次数	300000
迭代阈值	σ
迭代掩膜值	3σ
洁化操作增益值	0.85
像素分辨率	23.43 arcsec
权重模式	briggs ("robust" = −1.0)
偏振参数	计算4个偏振方向: XX, YY, XY, YX
频率通道参数	分成4个子通道进行频率综合

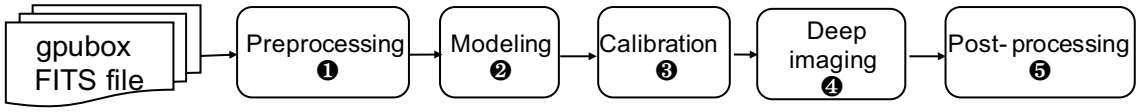

图 1 SKA低频阵列成像管线

Figure 1 The imaging pipeline for SKA low frequency array.

每个快照图像的流量密度标度校正系数. 在天体测量校正中, 校正步骤是通过对Stokes I 30.72 MHz子带图像中测得的源位置偏移进行径向基函数拟合, 在图像平面上比对MRC (Molonglo Reference Catalogue) [38]和NVSS [10]星表以确定参考源的位置并进行最终的位置修正.

2.2 管线热点分析

本节使用“串行运行”和“并行运行”两种方式测量低频成像管线的运行时间. 估算出理论加速比后, 将其与并行运行的实际加速比作比较, 综合考量优化潜力和优化难度, 设计最合适的优化策略.

串行运行 将流水线中所有阶段调用子程序的线程数或进程数设置为1, 并在运行过程中记录运行总时间$t_{\rm total}$和只能串行运行的时间$t_{\rm serial}$ (通常由开发人员阅读代码后估算), 最后通过阿姆达尔定律计算理论加速比$s_{\rm theory}$ (其中N为每个计算节点的处理器核心数)

$$s_{\rm theory} = \frac{t_{\rm total}}{t_{\rm serial} + \frac{t_{\rm total} - t_{\rm serial}}{N}}. \tag{1}$$

并行运行 将所有阶段具有并行能力的子程序的线程数或进程数调整为节点处理器核心数, 即在x86节点上以28核心运行, 在ARM节点上以96核心运行. 并行运行是在不修改流水线代码的前提下运行最快的方式, 也是当前流水线的默认运行方式. 由此可计算实际加速比$s_{\rm real}$, 其中$t_{\rm total}$是串行运行的总时间, $t_{\rm parallel}$是并行运行的总时间,

$$s_{\rm real} = \frac{t_{\rm total}}{t_{\rm parallel}}. \tag{2}$$

本文基于并行运行的总时间绘制程序热点图, 下文也以该时间为基线, 对比代码优化前后的运行时间. 选择运行时间长、与理论加速比差距大、未并行的函数进行优化. 应用程序在多机多核计算集群上, 可以利用任务的独立性, 使用不同的处理器核心并行处理任务, 从而缩短整体运行时间. 根据编写函数所用的计算机语言, 适用于SKA管线并行优化的并行编程方法包括C/C++多线程并行、Python多进程并行和Shell多任务并行.

成像管线在x86和ARM平台上的函数热点如图2和3所示, fits_warp, wsclean, shell_loops和calibrate是管线的函数热点. 其中fits_warp和shell_loop未做过并行优化且易于通过数据并行改写, 是首选优化目标; wsclean和calibrate本身已经做了多线程并行, 前者距离理想加速比仍有差距, 可进一步分析, 后者接近理想加速比, 难以进一步优化, 具体分析如下.

fits_warp是一个在❺后处理阶段用于校正图像的Python函数. 由于fits_warp的多个输入图像可以被独立处理, 且当前未做并行优化(加速比为1.0), 可采用Python多进程方法优化, 理论上可获得与节点核心数相等的加速比.

wsclean是一个在❹成像和❺后处理阶段用于成像的C++程序, 当前并行版本在x86和ARM上的加速比分别为3.6和8.4, 与估算的理想加速比18.3和56.5差距较大. 分析发现代码中数据同步机制采用了较保守的策略, 可尝试部分移除同步屏障, 并通过实验数据验证结果正确性.

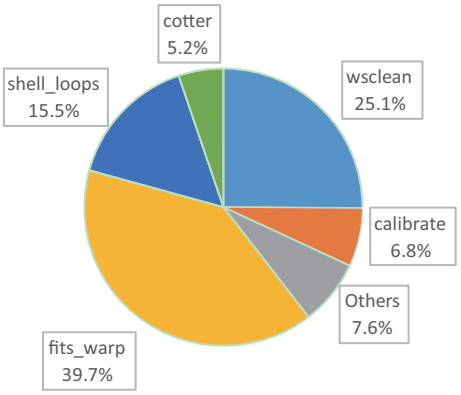

图 2 成像管线在x86节点上的热点

Figure 2 Hotspots of the imaging pipeline on a x86 node.

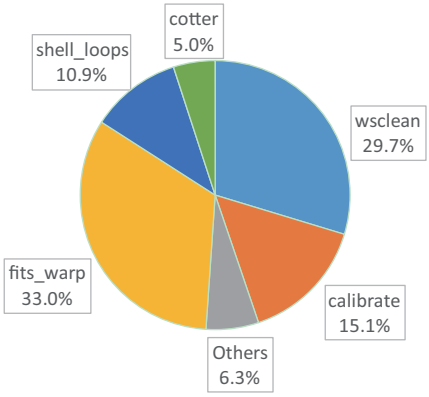

图 3 成像管线在ARM节点上的热点

Figure 3 Hotspots of the imaging pipeline on a ARM node.

calibrate在❷建模、❸校准和❹成像阶段被调用, 用于宽场干涉快速成像. 其运行情况如表3所示, 目前的加速比分别为16.5和28.6, 很接近理想加速比18.9和31.6. 进一步分析发现, calibrate代码中必须串行执行的部分, 集中在并行线程的启动和销毁上, 难以并行, 因此不再继续优化.

shell_loops是成像管线中的多层循环操作, 对不同天区的大量图像重复相似的处理流程. 原有Shell代码串行执行, 未做并行优化(实际加速比1.0), 可使用Shell多任务优化, 从而并行处理独立的图像来提高整体性能, 理论加速比与节点核心数相当.

关于fits_warp, wsclean和shell_loops的详细优化将在下面的小节中介绍(如表3所示).

2.3 通过python多进程优化fits_warp

fits_warp的执行流程及热点如图4所示. 其中, correct_images子函数是最耗时的部分, 该函数使用make_pix_model生成的校准模型来校准每幅图像. 而在correct_images中, CloughTocher2DInterpolatorw占了主要的运行时间. 我们发现, 该函数中待处理的图像之间没有依赖关系, 可以在不同的进程中并行处理. 使用Python multiprocessing改写后如代码1所示, 每个图像会在一个独立的进程中被校准, 由于每个进程会分配一个独立的CPU核心, 从而发挥多核节点的并行处理能力.

代码 1 使用Python多进程并行处理图像

```
def correct_images(fnames, dxmodel,
    dymodel, suffix):
    ...
    for fname in fnames:
        multiprocessing.Process(target=
            parallel_func, args=[fname, x,
             y, xy, suffix]).start()

def parallel_func(fname, x, y, xy, suffix)
    :
    fout = fname.replace(".fits", "_"+
        suffix+".fits")
    im = fits.open(fname)
    data = np.squeeze(im[0].data)
    ...
    return
```

2.4 通过C++多线程优化wsclean

wsclean在❷建模、❸校准和❹成像阶段被多次调用, 优化wsclean函数可以缩短这三个阶段的运行时间. 我们对wsclean进行了C++多线程优化和参数优化. 这些方法略微改变了原始wsclean代码中的算法, 因此第3.4节验证了优化代码的正确性. 本

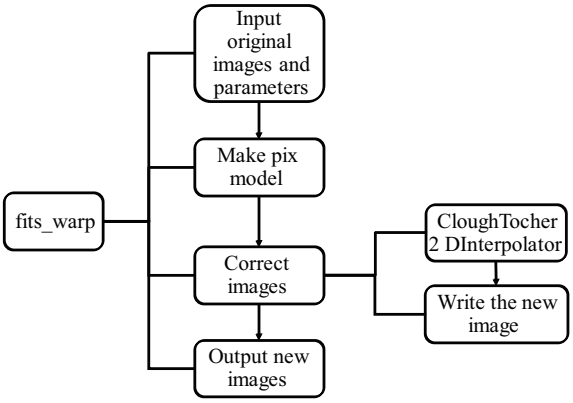

图 4 fits_warp执行流程及热点

Figure 4 Execution process and hotspots of fits_warp.

表 3 针对热点函数的并行优化策略

Table 3 Parallel optimization methods against hotspot functions

平台	热点函数	调用阶段	串行运行分析		当前并行效果评估		优化策略选择		
			串行耗时(s)	理论加速比	并行耗时(s)	实际加速比	是否已并行	优化难度	优化策略
x86	fits_warp	❸❹	2917	28.0	2917	1.0	否	低	Python多进程
ARM			3190	96.0	3190	1.0			
x86	wsclean	❹❺	5792	18.3	1609	3.6	是	中	C++多线程和参数搜索
ARM			23979	56.5	2871	8.4			
x86	shell_loops	❹❺	1103	28.0	1103	1.0	否	低	Shell多任务
ARM			1054	96.0	1504	1.0			
x86	caliberate	❷❸❹	8012	18.9	485	16.5	是	高	不做进一步优化
ARM			41699	31.6	1460	28.6			

文代码使用的是wgridder模式. 在源文件wsclean.cpp和wsmsgridder.cpp中, 对热点子函数runIndependentGroup进行C++多线程优化. 在wsclean.cpp中, 障碍函数griddingTaskManager->Finish()被移到了上层循环, 使得多个predict()任务的执行不需要等待上一任务的完成.

在wsmsgridder.cpp中, 用每个线程的写入函数取代了聚合函数. 每个工作线程(predictCalcThread)都会并行地写到它的目标内存区域, 从而提高整体的吞吐量. 同时, 将超参数优化应用于两个wsclean参数parallel-griddin和parallel-deconvolution. 前者控制imageMain和Predict子函数中的线程数量, 后者控制子图像的大小, 影响整体工作量和源匹配的准确性. 在不考虑使用额外程序辅助处理混叠效应的前提下, 本文将分别在x86和ARM节点上搜索这两个参数的最佳组合, 从而获得优化前后最高的源表匹配率.

2.5 通过数据并行优化shell_loops

MWA成像管线由Bash脚本语言组织执行. 在❹成像和❺后处理图像中的四个循环显示了数据并行优化的潜力. 这四个循环分别记为L1-img, L2-img, L1-cor和L2-cor, 分散于image.sh和snapshot_correct.sh函数中.

我们使用GNU Parallel[39]工具改写原先串行执行的Bash脚本, 使其使用多进程并行处理多个独立的任务. 为了使任务负载与节点的CPU核心数相匹配, 我们尝试进行搜索, 以获得GNU Parallel工作进程数和调用程序线程数的最优组合.

具体地说, L1-img在子通道、图像及偏振方向上进行迭代, 调用了子函数make_beam, pbcorrect和render来生成偏振图像. 由于任务之间没有数据依赖性, 我们先生成一个任务列表, 然后并行执行. 优化后的代码如代码2所示, 使用了与CPU核数相等的工作进程数(NCPU).

代码 2 L1-img中的Shell多任务优化

```
# Generate a job list for L1-img
for subchan in MFS 0000 0001 0002 0003; do
 for image in $images; do
 for pol in XXr XXi XYr XYi YXr YXi YYr
     YYi; do
 echo "$subchan-$img-$pol" >> jobs.txt
# Run jobs in parallel
cat jobs.txt | parallel -j $NCPU
    make$_$beam
cat jobs.txt | parallel -j $NCPU pbcorrect
cat jobs.txt | parallel -j $NCPU render
```

L2-img在偏振方向、图像和子通道上迭代, 调用子函数rms_measure, pyhead copy_metafitsheader来记录和重命名图像. 与L1-img类似, 我们展开这个3层循环, 使其并行运行, 并启动NCPU工作进程来并行处理独立数据, 在此省略代码示例.

L1-cor迭代make_beam和mv, 对图像进行流密度校正. 我们展开这个2层循环, 然后启动8个(4×2)工作进程来并行处理, 如代码3所示.

代码 3 L1-cor中的Shell多任务优化

```
# Generate a job list for L1-cor
for ((i=1; i<=(4); i++ )); do
 for pol in XX YY; do
 echo "$i-$pol" >> jobs.txt
# Run jobs in parallel
cat jobs.txt | parallel -j 8 make_beam
cat jobs.txt | parallel -j 8 mv
```

L2-cor迭代agean, BANE, pyhead和correct_gleam_flux函数, 通过将图像中检测到的源与GLEAM IDR6目录交叉匹配来校正快照图像的流量比例. 子函数BANE和aegean已经是多线程的, 为了配合CPU核心的总数, 其使用了较少的进程数, 如代码4所示.

代码 4 L2-cor中的Shell多任务优化

```
# Generate a job list for L2-cor
for ((i=1; i<=(4); i++ )); do
  for file in ${Xcorr[$i]} ${Ycorr[$i]};
      do
    echo "$i-$fil" >> jobs.txt
# Run jobs in parallel
cat jobs.txt | parallel -j $(NCPU/NBANE)
    --cores=$NBANE
cat jobs.txt | parallel -j $(NCPU/NAEGENE)
     aegene --cores=$NAEGENE
cat jobs.txt | parallel -j $NCPU pyhead
cat jobs.txt | parallel -j $NCPU
    correct_gleam_flux
```

在管线上采用的优化策略汇总在表3中. 对于fits_warp我们使用了Python多进程方法; 对于wsclean应用了C++多线程优化和参数搜索优化方法;

对于image.sh和snapshot_cotter.sh内的shell_loops, 使用了Shell多任务优化方法; 没有额外优化caliberate函数.

3 结果与讨论

实验是在CSRC-P[25]上完成的, 其单节点的软硬件配置如表4所示, 计算节点分别使用了x86和ARM指令集架构的处理器. 实验使用的算例来自GLEAM巡天的快照观测数据, 观测ID为1060179576, 观测时间为UTC时间2013-08-10 14:19:18至2013-08-10 14:21:10, 观测相位中心在(RA=290.51°, DEC=−26.73°).

实验对比的基线运行时间来自第2.2节介绍的"并行运行", 即在x86和ARM节点上分别指定28核心和96核心运行, 这是不修改代码的最快运行方式. 作为对比, 在保持参数不变的情况下运行优化后的代码.

为评估优化效果, 对比实际加速比$s_{\rm real}$与理论加速比$s_{\rm theory}$的差距, 其定义如(1)和(2)所示. 为了定量评估程序优化后利用并行资源的效率, 本文计算优化后的"并行效率", 其定义为实际加速比$s_{\rm real}$与理论加速比$s_{\rm theory}$的比值, 即程序使用单核心运行的时间$t_{\rm 1core}$与优化后运行时间$t_{\rm Ncore}$的比值再除以分配的核心数, 上限为100%, 计算公式如下:

$$E_{\rm parallel}=\frac{s_{\rm real}}{s_{\rm theory}}=\frac{t_{\rm 1core}}{t_{\rm Ncore}\times N}. \tag{3}$$

3.1 fits_warp优化结果

如图5所示, fits_warp的运行时间在优化后大幅下降, 与当前未并行优化的版本相比, 在x86和ARM平台上分别获得7.66和7.10的加速比, 相比28.0和96.0的理论加速比仍有提升空间. 分析发现受限于数据处理流程, fits_warp处理的输入文件数每个批次仅有12个, 不能充分发挥x86和ARM每节点28和96个核心计算能力. 未来的优化方向是在作业协调系统, 如DALiuGE[40]的协助下, 在一个计算节点上启动多条成像管线, 并行运行多个数据集管线.

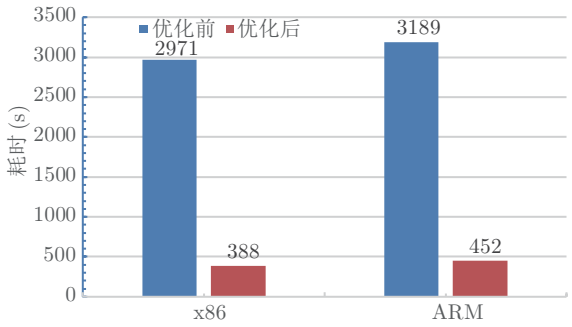

图 5 fits_warp在x86和ARM上的并行优化效果

Figure 5 Parallel optimization result of fits_warp on x86 and ARM.

3.2 wsclean优化结果

wsclean会根据调用它的阶段(❸校准或❹成像)执行不同的子函数, 我们分别分析在不同阶段的加速效果.

在运行参数选择上, 我们尝试不同的wsclean组合寻找最佳运行参数. 在优化之前, 线程数−j参数被设置为各平台的CPU核心数, 进程数设置为1. 优化后, 线程数−j在x86和ARM平台分别设置为7和24, 并启动4个进程, 使其乘积分别对应于两个平台的处理器核心数.

以之前已经并行化的wsclean代码作为基线, 我们进一步优化后的运行耗时如图6所示. 原有的wsclean已经是并行运行的, 在❸校准阶段, wsclean在x86和ARM平台上分别实现了1.5和1.6的加速, 相对于串行版本的加速比分别达到5.4和13.4. 在❹成像阶段, wsclean在x86和ARM平台上分别获得了4.2和5.0的加速比, 相对于串行版本的加速比分别达到15.1和42.0.

分析表明, ❸校准阶段调用的子函数runFirstInversion的运行时间占主导地位, 该函数串行运行的比例较高(70%–80%), 根据阿姆达尔定律, 预期能达到的加速比较低. 相对地, ❹成像阶段中的热点子函数predict和imageMain有很大一部分是可并行的, 根据阿姆达尔定律, 这两个子函数获得了较高的加速比. 进一步分析发现, imageMain进行并行写优化之后, 内

表 4 原型机上的软硬件环境

Table 4 Software and hardware configuration on CSRC-P

平台	处理器	内存 (GB)	操作系统	内核版本	软件环境
x86	Intel Xeon Gold 6132 × 2	384	CentOS 7	3.10.0	Python2.7 gcc8.3 wsclean2.7
ARM	Kunpeng 920 CPU × 2	512		4.14.0	

存带宽成为瓶颈, 导致无法达到理想加速比. 这类访存密集应用的优化仍然是难点, 未来可考虑重新设计算法, 减少每次算术操作的访存次数.

3.3 shell_loops优化结果

如图7所示, 在对image.sh和snapshot_correct.sh进行Shell多任务优化后, 运行时间大为减少. 这种优化方法在x86和ARM平台上都适用.

在加速比方面, L2-img获得了最高加速比, 在x86和ARM平台上分别得到了12.3和24.7的加速比. 其他循环的加速比如下: L1-img分别获得了4.8和5.0的加速比, L1-cor分别获得了3.5和6.6的加速比, L2-cor分别获得了3.7和3.8的加速比; 整体上, 我们在x86和ARM平台上分别获得了$\frac{1103}{259}$ = 4.26和$\frac{1017}{191}$ = 5.80的加速比.

在并行效率方面, L1-img在x86和ARM平台上的并行效率分别为$\frac{4.8}{24}$=20%和$\frac{5.0}{96}$=5%, L2-img在x86和ARM平台上的并行效率分别为$\frac{12.3}{24}$=52.1%和$\frac{24.7}{96}$=25.8%, L1-cor在x86和ARM平台上的并行效率分别为$\frac{3.5}{24}$=14.6%和$\frac{6.55}{96}$=7%, L2-cor在x86和ARM平台上的并行效率分别为$\frac{3.69}{24}$=15.4%和$\frac{3.84}{96}$ =4%.

我们在运行时间、加速比和并行效率方面对x86和ARM平台进行了比较. 由于有更多的CPU内核, ARM在所有循环的运行时间和加速比方面均优于x86平台. 只有L2-image使用了所有24和96个CPU内核, 其他3个循环没有表现出足够的并行性. 相较于ARM在CPU核心数上的优势($\frac{96}{24}$ = 4.0), 循环代码中相对较低的并行性制约了ARM, 使其在运行时间上相比x86仅获得较小的优势(提升了$\frac{259}{191}$ = 1.4倍).

整个MWA成像管线的优化结果总结在图8中. 管线的运行时间在x86上从7479 s降至2759 s, 在ARM上从9666 s降至4016 s, 分别获得了2.7和2.4的加速比. 在5个阶段当中, ❸校准阶段的运行时间在优化后降低了5%. ❹成像和❺数据后处理阶段的运行时间大为下降, 因为wsclean和shell_loops主要在这两个阶段执行.

3.4 成像结果验证

由于成像管线侧重于连续谱科学研究, 本节主要比较和分析总强度(Stokes I)的图像结果. 如表5的右4列所示, 优化前后获得的总强度图像的均方根(Root Mean Square, RMS)噪声值和PSF或BEAM的形状大小(包括BEAM的长半轴长BMAJ、短半轴长BMIN和位置角PA)基本相同. 为了进行对比分析, 我们使用射电源查找工具aegean, 分别对原始和优化后的成像管线获得的图像进行源搜寻并生成这些图像的射电源星表结果. 当源搜寻时, aegean采用了与MWA GLEAM星表建立所设置的相同参数, 即seedclip参数设为4, maxsummits参数设为5, 其余参数采用默认值. 源搜寻获得的射电源星表的对比情况如表5所示. 表5中同时给出了星表之间的匹配情况, 主要通过使用Topcat[41]工具分别对优化前后获得的星表结果进

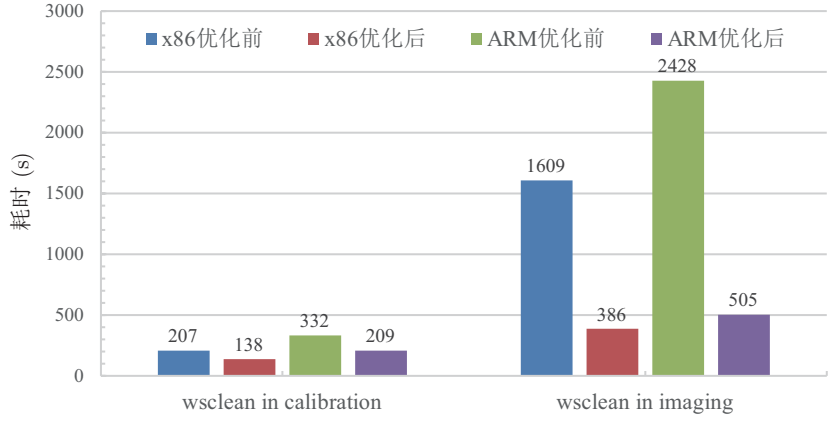

图 6 wsclean在x86和ARM平台上的并行优化效果

Figure 6 Parallel optimization result of wsclean on x86 and ARM.

行交叉匹配以及将它们和已发布的GLEAM星表进行匹配获得, 交叉匹配允许的最大误差设为100 arcsec.

从表5可以看出, 原始的管线在两种处理器上的结果大致相同, 只有近0.6%的差异. 优化后的管线获得的结果与原始管线获得的结果的匹配度分别为91.9% (x86)和92.0% (arm). 经过统计分析发现, 优化后获得的源表失去的193 (x86)和189 (arm)个射电源均为信噪比在4–5倍的弱点源, 优化后获得的源表多出的176 (x86)和169 (arm)个射电源也是相同信噪比范围内的弱点源. 造成优化前后差异的原因是, 优化后的管线使用了并行反卷积(parallel-deconvolution 512)和并行网格化(parallel-gridding 8). 在并行反卷积

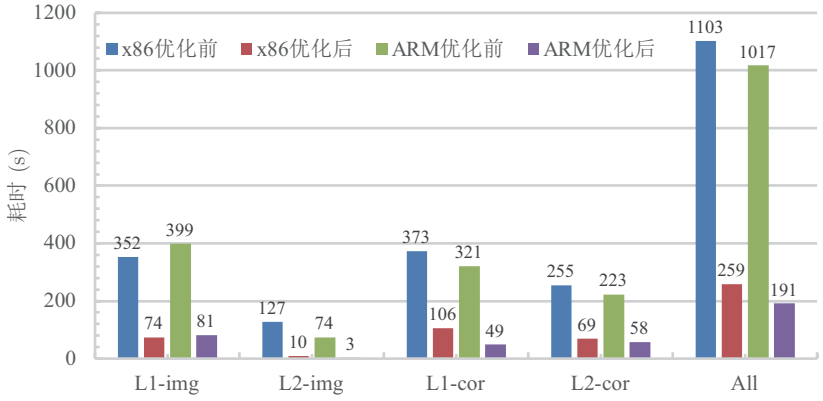

图 7 image.sh和snapshot_correct.sh中四个循环的多进程结果以及运行时间总和

Figure 7 Multiprocess results for the four loops in image.sh and snapshot_correct.sh, and the sum of the run time.

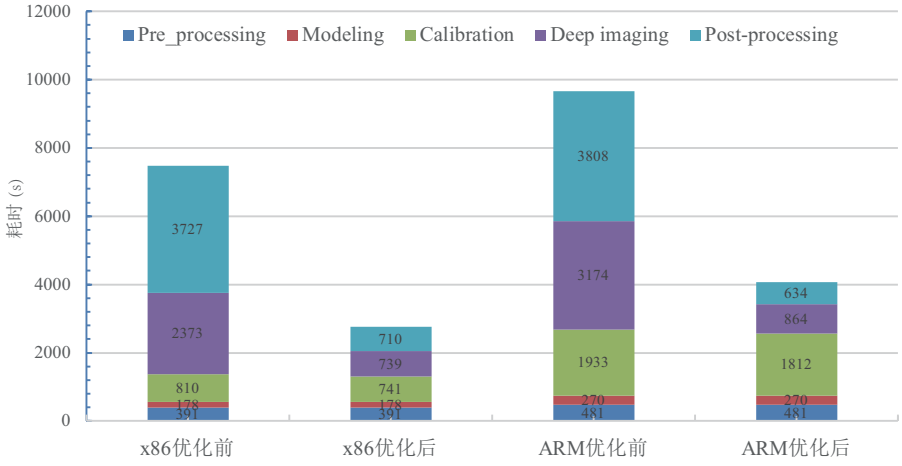

图 8 成像管线整体优化结果

Figure 8 Overall optimization results of the imaging pipeline.

表 5 成像结果对比

Table 5 Comparison of imaging results

平台	软件版本	检测数	与原始版本匹配数	与GLEAM发布星表匹配数	RMS (mJy)	BMAJ (arcsec)	BMIN (arcsec)	PA (°)
x86	原始版本	2374	2374	2083	18.02	108.27	96.37	106.42
x86	优化版本	2357	2181	2064	17.76	108.29	96.38	106.42
ARM	原始版本	2361	2361	2078	18.31	108.30	96.39	106.41
ARM	优化版本	2341	2172	2062	18.30	108.30	96.39	106.40

中, 使用Dijkstra算法将原始图像分割成许多最大尺寸为512的子图像, 该算法计算出子图像的最小分割路径, 并以所需的分割路径为界, 最小的分割路径在每次主循环的反卷积迭代之前都要重新确定[42]. 在并行网格化中, 原始网格被分割成8个子网格并同时执行网格化. 这两种并行方法的使用, 将使得合并后的图像在子图像或子网格的临界位置出现轻微的混叠效应, 这种影响将能够在后续的数据处理中得到消除[43].

在x86平台上, 优化后的成像管线获得的射电源的峰值与积分流量密度、位置和拟合BEAM的准确性分析如图9和10所示. 在图9中, 优化前后射电源的峰值流量密度的比值$S^{\rm peak}_{\rm x86_opt}/S^{\rm peak}_{\rm x86_orig}$和积分流量密度的比值$S^{\rm int}_{\rm x86_opt}/S^{\rm int}_{\rm x86_orig}$的分布均集中在0.99左右, 说明优化后的流量密度与原始流量密度基本相等. 优化前后射电源的位置赤经之差$\Delta_{\rm RA}$的分布集中在0.0左右, 优化前后射电源的位置赤纬之差$\Delta_{\rm DEC}$的分布集中在0.0左右, 表明优化后获得的射电源的位置与原来的基本相同. 此外, 在图10中, 优化前后获得的射电源的拟合BEAM的BMAJ, BMIN和PA之差的分布均集中在0.0左右, 表明优化后获得的图像的本地BEAM和分布与优化前的基本相同. 但也存在微小差异, 这种差异主要由于图像的本地BEAM存在微小差异, 根本原因在于优化后的管线采用了并行栅格化, 也正是这种差异导致了峰值流量和积分流量出现微小不同.

在ARM平台上, 优化后的成像管线获得的射电源的峰值与积分流量密度、位置和拟合BEAM的对比分析与在x86平台上相同, 如图11和12所示. 这说明优化后的成像管线得到的结果的准确度较高, 优化方法是有效的.

在完整性方面, 如表5的第5列所示, 是优化前后获得的源表与GLEAM公开星表的匹配结果. 以优化前的作为参考标准, 优化后获得的源表的完整度分别

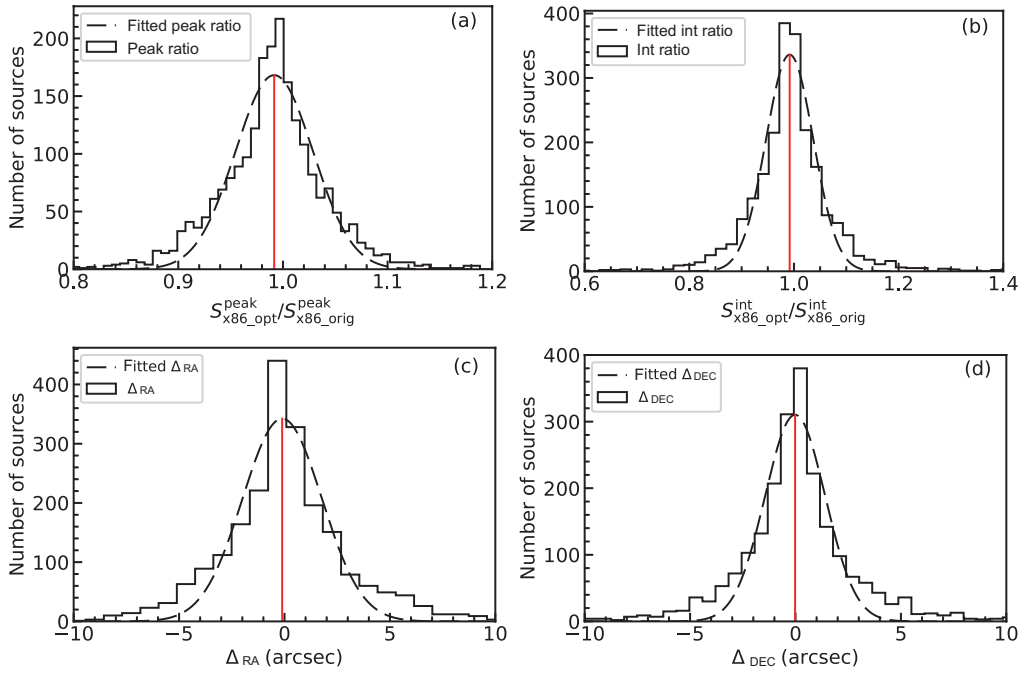

图 9 优化前后管线在x86上获得的星表的交叉匹配中流量密度和位置结果分析图. (a)和(b)图分别是优化前后射电源的峰值流量密度比值($S^{\rm peak}_{\rm x86\_opt}/S^{\rm peak}_{\rm x86\_orig}$)和积分流量密度比值($S^{\rm int}_{\rm x86\_opt}/S^{\rm int}_{\rm x86\_orig}$)的柱状图和高斯拟合结果, (c)和(d)图是分别是优化前后射电源的赤经之差($\Delta_{\rm RA}$)和赤纬之差($\Delta_{\rm DEC}$)的柱状图和高斯拟合结果. 4幅图中的红色直线均为高斯拟合曲线的均值位置

Figure 9 The analysis figure of flux and location in the cross-matching results of the catalog obtained by the pipeline before and after optimization on x86. Sub-figures (a) and (b) are the histogram and Gaussian fitting results of the peak flux ratio ($S^{\rm peak}_{\rm x86\_opt}/S^{\rm peak}_{\rm x86\_orig}$) and int flux ratio ($S^{\rm int}_{\rm x86\_opt}/S^{\rm int}_{\rm x86\_orig}$) of the radio source before and after optimization, respectively. Sub-figures (c) and (d) are the histogram and Gaussian fitting results of the difference between the right ascension ($\Delta_{\rm RA}$) and declination ($\Delta_{\rm DEC}$) of the radio source before and after optimization, respectively. The red straight lines in the 4 sub-figures are the mean positions of the Gaussian fitting curves.

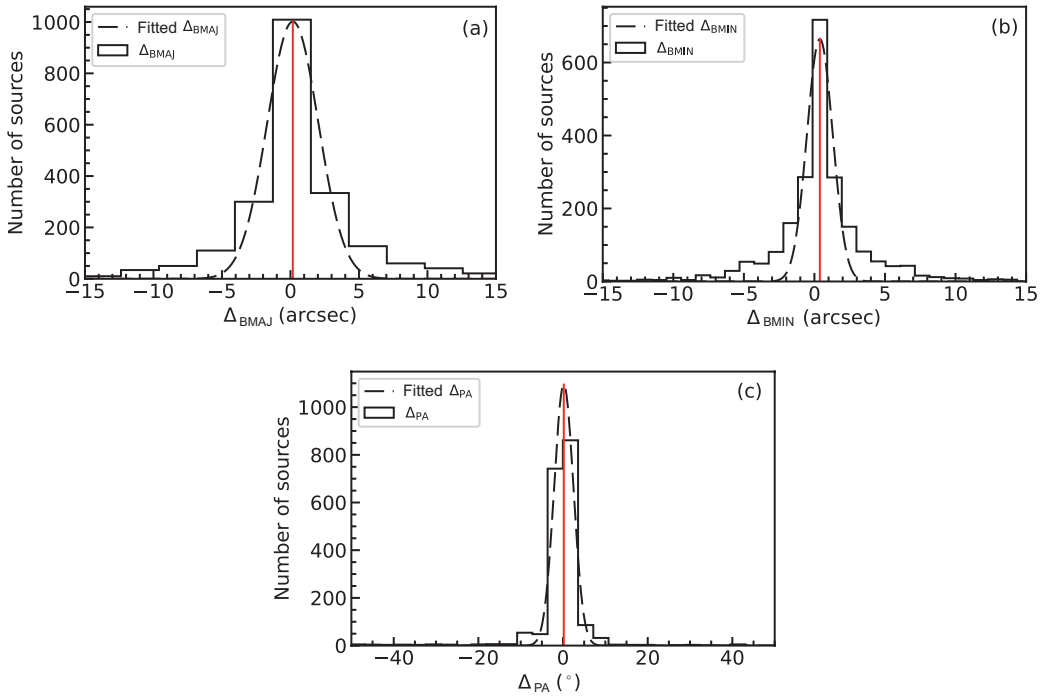

图 10 优化前后管线在x86上获得的星表的交叉匹配中射电源拟合BEAM结果分析图. (a)–(c)分别是优化前后射电源拟合BEAM的长半轴长BMAJ之差(Δ_{BMAJ})、短半轴长BMIN之差Δ_{BMIN}和位置角PA之差(Δ_{PA})的柱状图和高斯拟合结果

Figure 10 Analysis of the results of fitted BEAM of radio sources in the cross-matching of the catalog obtained by the pipeline before and after optimization on x86, respectively. Sub-figures (a)–(c) are the histogram and Gaussian fitting results of the difference of the long semi-axis length BMAJ (Δ_{BMAJ}), the difference of the short semi-axis length BMIN (Δ_{BMIN}) and the difference of the position angle PA (Δ_{PA}) of the fitted BEAM of the radio source before and after optimization, respectively.

为99.1% (x86)和99.2% (ARM). 综上所述, 优化后的管线得到结果的准确度和完整度都较高, 且流量分布的下限符合预期.

4 相关工作

在ARM处理器应用性能评估方面, 本文相比先前工作[28,44], 使用射电天文学中被广泛采用的低频成像工作流作为优化和测试对象, 这个工作流不仅包含C++等需要编译到目标平台的二进制程序, 还包含Shell和Python等无需编译的脚本程序, 拓宽了ARM处理器适用的科学计算领域.

本文优化结果在性能和实用性上也优于当前在多核CPU, GPU和FPGA上实现的结果. 文献[45]将SKA算法参考库(Algorithm Reference Library, ARL)的griddding函数分别移植到多核心CPU和GPU环境, 使用与本文同代x86处理器, 性能只提升了6.16%和8.18%. 文献[42]将wsclean的gridding函数移植到GPU和FPGA处理器, 相比CPU提高了能效, 但并未对比运行时间. 本文优化的流水线, 已经通过容器镜像部署在CSRC-P供平台用户使用, 在用法几乎不变的情况下, 提高了处理速度. 本文计划将优化后的wsclean代码反馈至上游社区.

在优化过程发现, 低频成像流水线是一个受内存带宽和并发任务数限制的应用. 异构平台天然受限于主存系统与GPU和FPGA系统间的通信带宽, 并且当前任务只有数十并发量, 不能充分发挥GPU数以千计的并行优势. 如何设计适用于GPU等异构计算平台的成像算法, 并获得与CPU版本一样的科学精度, 将是未来一段时间的研究热点.

SKA数据处理管线通常由天文领域研究人员以“脚本集合”的形式发布, 会被使用五年、十年甚至更长时间. 管线开发人员在发布脚本时, 难以预计脚本运行环境的巨大差异——可能是只有几个核心的

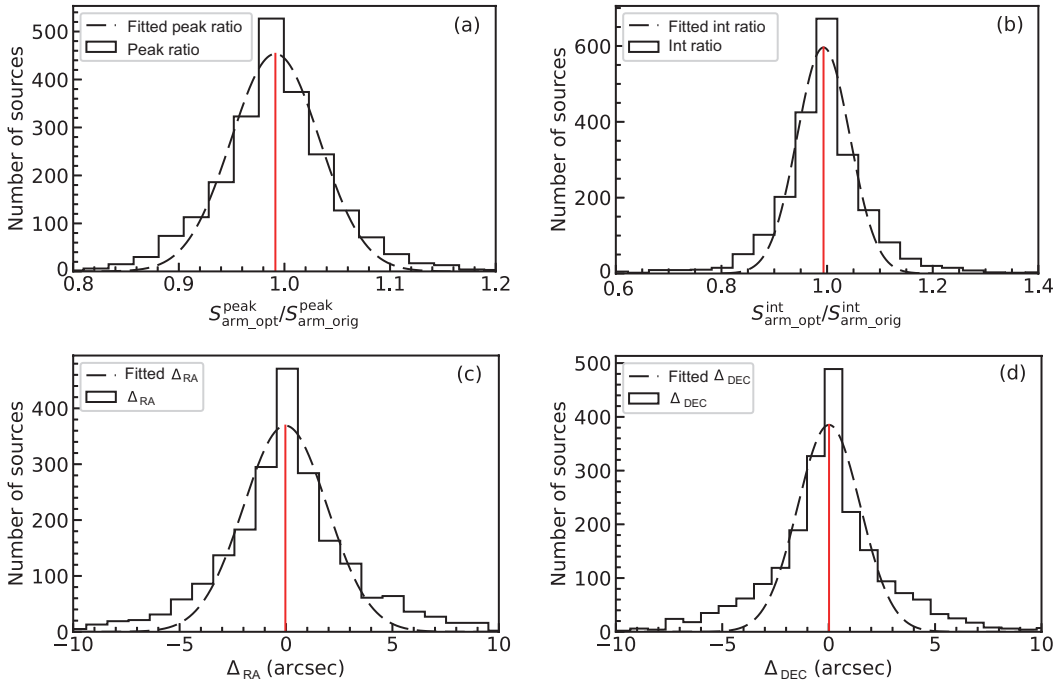

图 11 优化前后管线在ARM上获得的星表的交叉匹配中流量密度和位置结果分析图. (a)和(b)分别是优化前后射电源的峰值流量密度比值($S^{peak}_{arm\_opt}/S^{peak}_{arm\_orig}$)和积分流量密度比值($S^{int}_{arm\_opt}/S^{int}_{arm\_orig}$)的柱状图和高斯拟合结果, (c)和(d)图是分别是优化前后射电源的赤经之差(Δ_{RA})和赤纬之差(Δ_{DEC})的柱状图和高斯拟合结果. 4幅图中的红色直线均为高斯拟合曲线的均值位置

Figure 11 The analysis figure of flux and location in the cross-matching results of the catalog obtained by the pipeline before and after optimization on ARM. Sub-figures (a) and (b) are the histogram and Gaussian fitting results of the peak flux ratio ($S^{peak}_{arm\_opt}/S^{peak}_{arm\_orig}$) and integrated flux ratio ($S^{int}_{arm\_opt}/S^{int}_{arm\_orig}$) of the radio source before and after optimization, respectively. Sub-figures (c) and (d) are the histogram and Gaussian fitting results of the difference of the right ascension (Δ_{RA}) and declination (Δ_{DEC}) of the radio source before and after optimization, respectively. The red straight lines in the 4 sub-figures are the mean positions of the Gaussian fitting curves.

处理器, 也可能是几十到几万核心的超算系统——在脚本使用年限中, 如何挖掘代码潜力、适配新的计算系统, 本文进行了有益尝试. 这些方法也能被其他以脚本组织的管线所借鉴, 包括挖掘数据级并行, 使用Python和Shell语言快速提升处理能力; 移除内存同步屏障提高并行处理速度; 尝试多个参数组合, 选取最优组合, 在优化过程中仔细检验结果正确性等.

5 总结

本文在CSRC-P上, 对MWA低频成像应用进行了并行性能优化. 通过热点剖析, 发现了应用整体流程的性能热点在于fits_warp.py, wsclean, calibrate和shell_loops函数. calibrate在原有的并行方法下已经取得了较高的并行效率, 因此没有进一步优化. 对于shell_loops和fits_warp.py, 本文主要使用Python multiprocessing进行了多进程的并行, 取得了很高的并行加速比. 对于wsclean, 本文优化其C++多线程并行算法, 验证了算法精度能满足SKA使用要求. 本文的优化方法被证明在x86和ARM平台上都有很好的并行效果, 相比原先使用的未充分优化版本, 运行时间分别从7479和9666 s, 降低为2759和4061 s, 实现了2.7和2.4倍整体加速. ARM平台相比技术成熟、应用广泛的x86平台, 尽管在总运行时间和加速比上仍有差距, 但在保证计算精度满足科学要求的前提下, 运行时间已经能满足当前数据处理的需求, 也能方便地支持SKA项目使用的开源计算软件. 本文介绍“整体优化方法”从程序热点剖析入手, 分析理论性能与实际性能, 寻找合适的优化函数, 并且关注了流水线优化过程中易于被忽略的Python和Shell脚本并行加速潜力. 这对SKA领域的其他数据处理应用具有很好的借鉴意义.

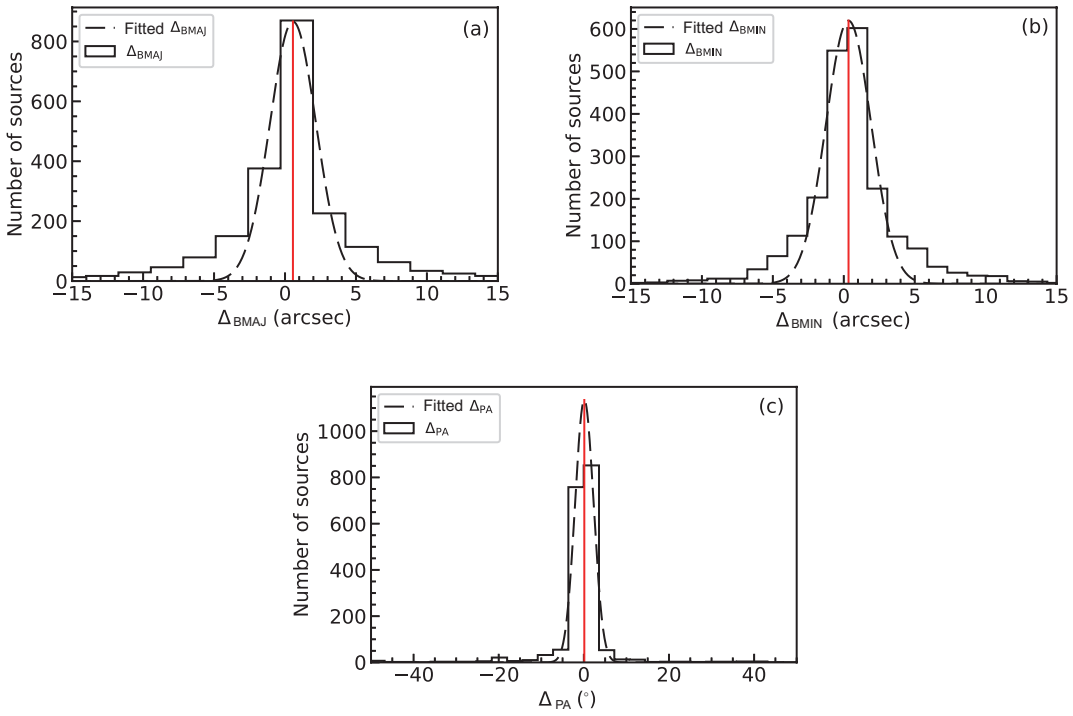

图 12 优化前后管线在ARM上获得的星表的交叉匹配中射电源拟合BEAM结果分析图. (a)–(c)分别是优化前后射电源拟合BEAM的长半轴长BMAJ之差(Δ_{BMAJ})、短半轴长BMIN之差Δ_{BMIN}和位置角PA之差(Δ_{PA})的柱状图和高斯拟合结果

Figure 12 Analysis of the results of fitted BEAM of raido sources in the cross-matching of the catalog obtained by the pipeline before and after optimization on ARM, respectively. Sub-figures (a)–(c) are the histogram and Gaussian fitting results of the difference of the long semi-axis length (Δ_{BMAJ}), the difference of the short semi-axis length Δ_{BMIN} and the difference of the position angle (Δ_{PA}) of the fitted BEAM of the radio source before and after optimization, respectively.

参考文献

1 Prandoni I, Seymour N. Revealing the physics and evolution of galaxies and galaxy clusters with SKA continuum surveys. arXiv: 1412.6512

2 Van Eck C L, Haverkorn M, Alves M I R, et al. Diffuse polarized emission in the LOFAR two-meter sky survey. Astron Astrophys, 2019, 623: A71

3 Riseley C J, Galvin T J, Sobey C, et al. The POlarised GLEAM survey (POGS) II: Results from an all-sky rotation measure synthesis survey at long wavelengths. Publ Astron Soc Aust, 2020, 37: e029

4 Hale C L, McConnell D, Thomson A J M, et al. The rapid ASKAP continuum survey paper II: First Stokes I source catalogue data release. Publ Astron Soc Aust, 2021, 38: e058

5 Botteon A, Shimwell T W, Cassano R, et al. The Planck clusters in the LOFAR sky. Astron Astrophys, 2022, 660: A78

6 Norris R P, Marvil J, Collier J D, et al. The evolutionary map of the universe pilot survey. Publ Astron Soc Aust, 2021, 38: e046

7 Wang Y, Tuntsov A, Murphy T, et al. ASKAP observations of multiple rapid scintillators reveal a degrees-long plasma filament. Mon Not R Astron Soc, 2021, 502: 3294–3311

8 Helmboldt J F, Hurley-Walker N. Ionospheric irregularities observed during the GLEAM survey. Radio Sci, 2020, 55: 1–5

9 Becker R H, White R L, Helfand D J. The FIRST survey: Faint images of the radio sky at twenty centimeters. Astrophys J, 1995, 450: 559

10 Condon J J, Cotton W D, Greisen E W, et al. The NRAO VLA sky survey. Astron J, 1998, 115: 1693–1716

11 Lacy M, Baum S A, Chandler C J, et al. The Karl G. Jansky very large array sky survey (VLASS). Science case and survey design. Publ Astron Soc Pacific, 2020, 132: 035001

12 Wayth R B, Lenc E, Bell M E, et al. GLEAM: The GaLactic and extragalactic all-sky MWA survey. Publ Astron Soc Aust, 2015, 32: e025
13 Intema H T, Jagannathan P, Mooley K P, et al. The GMRT 150 MHz all-sky radio survey. Astron Astrophys, 2017, 598: A78
14 Shimwell T W, Röttgering H J A, Best P N, et al. The LOFAR two-metre sky survey. Astron Astrophys, 2017, 598: A104
15 Jarvis M J, Taylor A R, Agudo I, et al. The MeerKAT international GHz tiered extragalactic exploration (MIGHTEE) survey. arXiv: 1709.01901
16 Norris R P, Hopkins A M, Afonso J, et al. EMU: Evolutionary map of the universe. Publ Astron Soc Aust, 2011, 28: 215–248
17 Hurley-Walker N, Callingham J R, Hancock P J, et al. GaLactic and Extragalactic All-sky Murchison Widefield Array (GLEAM) survey. I. A low-frequency extragalactic catalogue. Mon Not R Astron Soc, 2017, 464: 1146–1167
18 White S V, Franzen T M O, Riseley C J, et al. The GLEAM 4-Jy (G4Jy) sample. I. Definition and the catalogue. Publ Astron Soc Aust, 2020, 37: e018
19 George L T, Dwarakanath K S, Johnston-Hollitt M, et al. A study of halo and relic radio emission in merging clusters using the Murchison Widefield Array. Mon Not R Astron Soc, 2017, 467: 936
20 For B Q, Staveley-Smith L, Hurley-Walker N, et al. A multifrequency radio continuum study of the Magellanic Clouds. I. Overall structure and star formation rates. Mon Not R Astron Soc, 2018, 480: 2743–2756
21 Hurley-Walker N, Filipović M D, Gaensler B M, et al. New candidate radio supernova remnants detected in the GLEAM survey over $345^\circ < l < 60^\circ$, $180^\circ < l < 240^\circ$. Publ Astron Soc Aust, 2019, 36: e045
22 Riseley C J, Lenc E, Van Eck C L, et al. The POlarised GLEAM Survey (POGS) I: First results from a low-frequency radio linear polarisation survey of the southern sky. Publ Astron Soc Aust, 2018, 35: 43
23 Tremblay C D, Jones P A, Cunningham M, et al. A molecular line survey around orion at low frequencies with the MWA. Astrophys J, 2018, 860: 145
24 Gong H Y, Zhang Z L, Xue M Y, et al. Search and detection of northern pulsars in the side lobes of the murchison wide-field array (in Chinese). Sci Sin-Phys Mech Astron, 2020, 50: 109501 [龚宏宇, 张仲莉, 薛梦瑶, 等. 利用默奇森大视场阵列旁瓣探寻北天脉冲星. 中国科学: 物理学 力学 天文学, 2020, 50: 109501]
25 An T, Wu X P, Hong X. SKA data take centre stage in China. Nat Astron, 2019, 3: 1030
26 An T, Wu X, Lao B, et al. Status and progress of China SKA Regional Centre prototype. Sci China-Phys Mech Astron, 2022, 65: 129501
27 Lao B Q, Zhang Y K, An T, et al. Software platform on China SKA Regional Center prototype system (in Chinese). Sci Sin-Phys Mech Astron, 2023, 53: 229507, ChinaXiv: 202206.00173 [劳保强, 张迎康, 安涛, 等. 中国SKA区域中心原型系统——软件平台. 中国科学: 物理学 力学 天文学, 2023, 53: 229507, ChinaXiv: 202206.00173]
28 Jackson A, Turner A, Weiland M, et al. Evaluating the arm ecosystem for high performance computing. In: Proceedings of the Platform for Advanced Scientific Computing Conference. New York: ACM, 2019. 1–11
29 Wei J W, Zhang C F, Zhang Z L, et al. Parallel optimization of the pulsar search pipeline (in Chinese). Sci Sin-Phys Mech Astron, 2023, 53: 229506, ChinaXiv: 202206.00186 [韦建文, 张晨飞, 张仲莉, 等. 射电脉冲星搜索的优化方法. 中国科学: 物理学 力学 天文学, 2023, 53: 229506, ChinaXiv: 202206.00186]
30 An T. Science opportunities and challenges associated with SKA big data. Sci China-Phys Mech Astron, 2019, 62: 989531
31 Lao B Q, An T. Deployment of SKA low frequency imaging system in China SKA Regional Centre (in Chinese). Sci Sin-Phys Mech Astron, 2020, 50: 059501 [劳保强, 安涛. SKA低频成像系统在中国SKA区域中心的部署. 中国科学: 物理学 力学 天文学, 2020, 50: 059501]
32 Offringa A R, Wayth R B, Hurley-Walker N, et al. The low-frequency environment of the murchison widefield array: Radio-frequency interference analysis and mitigation. Publ Astron Soc Aust, 2015, 32: e008
33 Offringa A R. Compression of interferometric radio-astronomical data. Astron Astrophys, 2016, 595: A99
34 Offringa A R, Trott C M, Hurley-Walker N, et al. Parametrizing epoch of reionization foregrounds: A deep survey of low-frequency point-source spectra with the Murchison Widefield Array. Mon Not R Astron Soc, 2016, 458: 1057–1070
35 Offringa A R, McKinley B, Hurley-Walker N, et al. wsclean: An implementation of a fast, generic wide-field imager for radio astronomy. Mon Not R Astron Soc, 2014, 444: 606–619
36 Hurley-Walker N, Hancock P J. De-distorting ionospheric effects in the image plane. Astron Computing, 2018, 25: 94–102
37 Hancock P J, Trott C M, Hurley-Walker N. Source finding in the era of the SKA (Precursors): Aegean 2.0. Publ Astron Soc Aust, 2018, 35: e011
38 Large M I, Cram L E, Burgess A M. A machine-readable release of the molonglo reference catalogue of radio sources. Observatory, 1991, 111: 72–75
39 Tange O. GNU parallel: The command-line power tool. The USENIX Magazine, 2011, 36: 42
40 Wu C, Tobar R, Vinsen K, et al. DALiuGE: A graph execution framework for harnessing the astronomical data deluge. Astron Computing, 2017, 20: 1–15

41 Taylor M B. TOPCAT & STIL: Starlink table/VOTable processing software. In: P. Shopbell, M. Britton, and R. Ebert, eds. Astronomical Data Analysis Software and Systems XIV ASP Conference Series, Vol. 347. San Francisco: Astronomical Society of the Pacific, 2005. 29
42 Veenboer B, Van der Tol S, Offringa A R, et al. Radio-astronomical imaging with wsclean and image-domain gridding. In: Proceedings of the 33rd General Assembly and Scientific Symposium of the International Union of Radio Science. Rome: IEEE, 2020. 1–4
43 Offringa A R, Mertens F, van der Tol S, et al. Precision requirements for interferometric gridding in the analysis of a 21 cm power spectrum. Astron Astrophys, 2019, 631: A12
44 Abdurachmanov D, Elmer P, Muzaffar S. Initial explorations of ARM processors for scientific computing. J Phys-Conf Ser, 2014, 523: 012009
45 Hu X Y, Zhao Y Q. Gridding algorithm in ARL based on GPU parallelization. In: Proceedings of the 6th ACM/ACIS International Conference on Applied Computing and Information Technology. New York: Association for Computing Machinery, 2018. 13–18

Optimization of parallel processing of the Square Kilometre Array low-frequency imaging pipeline

WEI JianWen[1], ZHANG ChenFei[1], LAO BaoQiang[2,3*], LIN James[1] & AN Tao[2]

[1]*Network & Information Center, Shanghai Jiao Tong University, Shanghai 200240, China;*
[2]*Shanghai Astronomical Observatory, Chinese Academy of Sciences, Shanghai 200030, China;*
[3]*School of Physics and Astronomy, Yunnan University, Kunming 650500, China*

Data processing of the square kilometer array (SKA) is performed in pipeline mode, and the execution efficiency of pipeline mode is an important factor in SKA data processing. Continuum imaging is a primary observation mode of SKA and a prerequisite for many other scientific works. In this paper, we take the imaging pipeline of the SKA low-frequency precursor Murchison widefield array as an example and optimize the parallel processing pipeline on the China SKA regional centre prototype (CSRC-P). Previous optimization schemes have focused on a few performance hotspots and lacked systematic optimization of the overall pipeline, resulting in a relatively poor overall speedup ratio. In this paper, we propose a global optimization scheme that combines C++ multi-threading, Python multi-processing, and Shell multi-tasking parallelism for pipelines using multiple programming languages and image datasets that can be processed independently and verify the accuracy of the optimization results. Experiments show that the optimized pipeline achieves an overall speedup of 2.7- and 2.4-fold on the x86 and advanced RISC machine (ARM) nodes of CSRC-P, respectively, and the ARM compute nodes show good adaptability to SKA applications. The optimization strategies and methods in this paper also apply to other SKA applications and will be useful for the scientific operation and future operation of the SKA precursor telescope.

square kilometre array, murchison widefield array, high performance computing, parallel optimization

PACS: 07.05.-t, 07.05.Bx, 07.05.Kf, 95.85.-e, 95.85.Bh

doi: 10.1360/SSPMA-2022-0262