分类号_____ 密级_____

UDC _____ 编号_____

# 中国科学院研究生院

# 硕士学位论文

## 高效海量星表融合工具集的开发与 GPU 并行计算的天文应用研究

裴彤

指导教师 张彦霞 副研究员 中国科学院国家天文台

申请学位级别 硕士 学科专业名称 天文技术与方法

论文提交日期 2011 年 4 月 论文答辩日期 2011 年 5 月

培养单位 中国科学院国家天文台

学位授予单位 中国科学院研究生院

答辩委员会主席 赵永恒 研究员

# Development of High-efficient Large-scale Catalogue Oriented Fusion Toolset and Application of GPU in Astronomy

Pei Tong (Astronomical Technology & Method)

Directed by Asso. Prof. Zhang Yan-xia

National Astronomical Observatories
Chinese Academy of Science

May 2011

*Submitted in total fulfillment of the requirements for the master degree in Astronomical technology and method*

# 摘要

随着天文学“数据雪崩”和“信息爆炸”时代的到来，海量天文数据的高效融合与快速处理成为多波段天文学研究的一个热点问题。针对该问题，我们开发了海量星表融合工具集，包括对原有交叉证认程序在性能和易用性方面大幅度的改进，自动入库、创建 HTM 索引表以及证认后数据提取程序的开发，SIMBAD 和 NED 自动证认工具的实现等等。利用该工具集，我们从射电巡天星表 FIRST 和 NVSS 中为郭守敬望远镜（LAMOST）选取了一部分射电观测源。从这个应用实例可以看到，该工具集大大提高了海量星表交叉证认及后续处理的效率。另外，我们还尝试将时下流行的 CUDA 并行计算技术引入天文学研究，利用它来加速 k 近邻（kNN）分类算法和郭守敬望远镜（LAMOST）数据处理程序中的抽谱算法，均获得了较好的效果，加速比达到十几倍乃至几十倍。我们的研究结果表明 CUDA 技术在海量天文数据处理中有其独特的优势。

关键词：海量，交叉证认，并行计算，CUDA

# Abstract

As a result of "data avalanche" and "information explosion" in astronomy nowadays, high-efficient processing and fusion of massive data become a hot issue in multi-band astronomy. In order to solve this problem, we have developed a toolset to fuse large-scale catalogues, including improving the original automated-database-generation program and cross-match program, fulfilling HTM-index-table-creation program and cross-match-result-extraction program, implementing SIMBAD & NED auto-cross-match tools. Using these tools, we have preselected some radio sources from the FIRST and NVSS sky survey catalogs for GuoShouJing Telescope (LAMOST). This application example proves that the toolset we developed greatly improves the efficiency of the identification and follow-up. In addition, we have also tried using CUDA technology to accelerate k-nearest-neighbor (kNN) classification algorithm and the spectrum-extracting algorithm in the data processing program of GuoShouJing Telescope (LAMOST), and got very good results. Our experiments indicate that CUDA technology has unique advantage in the processing of massive astronomical data.



Keywords: Massive Data, Cross-match, Parallel Computation, CUDA

# 目录

# 第一章 引言

天文观测数据来自于不同的观测时间、观测地点和观测仪器，覆盖伽玛射线、X射线、紫外、光学、红外以及射电等多个波段，且分散保存在世界各地的数据中心，管理方式也存在很大差异——这些因素都导致了天文数据的复杂性。因此，如何有效地将不同来源的天文观测数据进行融合以得到更多更全面的天体信息便成为当今天文技术研究的一个热点。

另一方面，随着天文观测设备数量不断增加、观测能力不断增强，人们获得的天文观测数据急剧增长，呈现“数据爆炸”的态势，总体积早已达到 Terabyte 甚至 Petabyte 的规模，并且预计还将长期增长下去。单个星表达到亿行以上已不鲜见，在未来几年内出现千亿、万亿行的星表是极有可能的。面对如此巨量的数据，传统的计算工具和计算方法受到严峻挑战，甚至已经难以应对，因此有必要开拓新的计算方法，引入高性能计算领域的新成果、新技术。

## 1.1 星表融合研究背景

数据融合是一个比较宽泛的概念，在不同学科中有不同的具体含义。在天文学研究中，数据融合一般是指将多个独立的星表或图象、数据库或数据集通过位置或星名等信息整合成一个统一的整体。

天文工作者尤其关注星表数据的融合。通过星表融合，可以将同一目标源在多个星表中的观测信息整合在一起，从而加深对该天体的认识，促进新天体或新现象的发现。将不同波段的星表、特别是巡天项目产生的大型星表进行融合，一直是很多天文学研究的基础，特别是对于多波段数据挖掘和统计分析工作具有重要意义。

而星表融合最关键的一步，就是交叉证认计算。近年来各国的天文工作者都在这一问题进行积极研究，开发出了各种交叉证认工具。

美国虚拟天文台（NVO）[13]提供了基于微软 SQL Server 数据库的纯 SQL 指令的交叉证认服务——OpenSkyQuery，从而为大型巡天项目 SDSS 的数据访问平台整合了其他多家天文台的数据集。但这种方法在实现上受限于特定的数据库系统，证认的规模也受限于内存的容量，且规定一次证认的条数最多不能超过 5000 条。

英国虚拟天文台（AstroGrid）[14]也在其网站的数据查询中提供了简单的交叉证认

服务，但在数据规模和效率上仍然没有大的突破。

VizieR[15]是法国斯特拉斯堡数据中心开发的数据融合工具，提供了目前已发表数据的各种查询方式，如按星表名、日期、图像、光谱、波段、类型等。它仅支持小规模的交叉证认计算，用户可上传自己的星表，在预置星表与自上传星表间进行交叉证认。

Aladin[16]是法国斯特拉斯堡数据中心开发的另一数据整合工具， 可以互动地可视化天空任何一部分图像，在同一视场中还可以叠加来自 VizieR、SIMBAD、NED 或其他星表的源。它在多波段交叉证认方面功能较为完善，但同样尚未实现任意海量数据的交叉证认。

SIMBAD[17]功能类似 VizieR，同样提供小样本多源查询功能，其核心也就是小样本的交叉证认功能。选源时可以通过 SIMBAD 查看所选源的类型及是否已被证认过等信息。

NED[18]是 NASA 的银河系外数据库，它提供的批处理查询功能可以查询某源在其收录资料中的所有信息，包括最新发表文章中的相关信息。功能上类似 SIMBAD，但主要针对河外源。

Topcat[19]是一个功能比较丰富的表数据查看、编辑、分析、绘图工具。它也提供了交叉证认功能，但对超过百万行的数据证认起来就比较吃力，千万行以上的几乎无法处理。

从上面的分析可以看出，以上这些工具虽然各有优点，也有力地促进了天文学研究，但都无法进行大规模的星表交叉证认，远不能满足进行大样本研究工作的天文学家的需求。针对这一问题，国家天文台已毕业博士生高丹同学行了积极的探索，提出了一种基于 HTM 球面索引和 KD-Tree 存储结构的快速交叉证认算法，并开发出相应的交叉证认工具，一定程度上满足了天文工作者对星表融合的需求，但限于程序的效率，仍只适用于几十万条到几百万条的中等数据量。

## 1.2 高性能计算技术发展趋势

高性能计算正在变得与计算密集型应用越来越密不可分，已成为宇宙学模拟、天气预报、气候研究、分子建模、物理仿真、密码分析、石油勘探等领域的重要手段。当今高性能计算系统的发展呈现以下几个趋势：

1. 用多核/众核 CPU 组建大规模集群系统

高性能计算系统将不仅仅依靠扩展系统规模、提高单个处理器的性能来提高整体性能，还需要应用多核/众核技术。未来的集群系统将包含数百个至数百万个计算内核，这将对并行软件开发、系统容错和硬件可靠性提出很高要求。多核/众核技术的进一步发展将开发动态多核处理器，通过改进不同类别的硬件架构以满足并行计算的要求，甚至开发矢量扩展指令集多核处理器。

2. 并行计算不可或缺

随着组成高性能计算系统的部件数量不断增长，应用程序的并行化变得不可避免。并行程序比串行程序更难编写，不同任务间的通信与同步是最常见的难点之一。并行编程工具的发展已远远落后于高性能计算硬件的发展。对于每个需要在多核系统上运行的程序，并行化都是关键点。另外，串行程序向并行系统的迁移仍然有大量的工作要做。

3. GPU 与多核 CPU 的竞争

多核 CPU 正面临着来自 GPU 的强有力的竞争。GPU 具有三大基本优势：运算速度更快，速度提升也更快；成本更低；能耗比 CPU 低。AMD 与 Nvidia 的 GPU 已经实现了每秒百万亿次浮点计算的性能，而通常的多核 CPU 还无法达到这样的水平。GPU 也存在局限性，它们仅适用于一部分类型的任务。这是由于最初 GPU 专为图像处理而设计，主要用作处理数据流。Nvidia[20]公司推出的 CUDA[21]计算平台使开发人员可以不必使用图形处理专用 API 而使用标准 C 或 Fortran 对其 GPU 进行编程，从而将 GPU 用于通用计算。

如今 CPU+GPU 异构并行计算方案已经成为一种流行。2009 年 9 月美国加利福尼亚州圣克拉拉市-橡树岭国家实验室（ORNL）宣布，将采用 NVIDIA 公司代号为“Fermi”的第三代 CUDA GPU 架构来构建世界上最快的超级计算机。而 2010 由中国国防科技大学打造并获得第 36 届“世界超级计算机 500 强”冠军的“天河一号”超级计算机系统，就使用了 7000 多块基于“Fermi”架构 GPU 的 Tesla M2050 高性能计算卡。

4. 绿色 IT 与能耗

百万亿次级以上的高性能计算系统的能耗都很巨大。如果任凭当前的能耗趋势发展下去，那么未来 4 至 5 年内高性能计算中心的能耗将翻番。根据推断，未来的百万万亿次级高性能计算系统的能耗将高达数百兆瓦。因此不论是各高性能计算中心的负

责人，还是各次高性能计算会议，能耗都是关注的焦点。

## 1.3 论文主要内容

本论文的核心是海量星表融合和高性能计算技术在天文研究中的应用，目标是开发出高效易用的星表融合工具和探索 CUDA 技术用于天文研究的可行性，主要包括以下内容：

1）在高丹等人工作的基础之上，对交叉证认程序进行了大幅度改进，开发语言改用 Python 和 C 语言，引入多核并行计算，充分发挥 CPU 的效能，开发了图形用户界面，使程序更加易用。

2）开发了“本地星表入库”、“创建 HTM 索引表”和“证认数据提取”等几个辅助程序，来协助用户完成交叉证认前后的必要步骤。

3）对天文工作者常用的 NED/SIMBAD 网络天文数据库的交叉证认方式进行了调研，对比分析之后，开发出自动证认工具，以帮助天文工作者进行大数据量的快速证认。

4）综合利用上述工具从 FIRST 和 NVSS 射电巡天星表中为郭守敬望远镜(LAMOST)选取射电观测源。

5）实现了基于 CUDA 的 kNN 天体分类算法，相比 CPU 串行程序获得了数十倍的加速比。

6）尝试利用 CUDA 技术加速郭守敬望远镜抽谱算法，在使用双精度浮点运算的情况下比 C 语言串行程序获得约 17 倍的加速比。

# 第二章 高效海量星表融合工具集的开发和应用

天文星表融合的核心步骤是交叉证认计算。交叉证认是指将来自不同星表的源，根据其某些属性（如位置、星名、星等等）的相关性，相互联系起来。交叉证认是LAMOST 科学目标的三大核心子课题之一，有重要的现实意义和科学价值。本章紧紧围绕“交叉证认”这一核心，进行了一系列研究和开发，形成了一套高效的海量星表融合工具集，旨在帮助天文工作者提高工作效率，促进天文学研究的进展。

## 2.1 基于 MySQL 数据库的海量星表交叉证认程序的改进

近年来，随着星表规模的不断增大，天文工作者常用的交叉证认工具（如Vizier、SIMBAD、Topcat、Aladin 等）遇到了内存不足、证认计算量过大等问题，难以胜任海量星表的交叉证认工作。为实现海量星表交叉证认，高丹等人在总结国内外研究现状的基础上，研究比较了多级三角划分法（Hierarchical Triangular Mesh，简称HTM）[22]和多级等面积同纬度划分法（Hierarchical Equal Area isoLatitude Pixelisation，简称 HEALPix）[23]，提出“基于 HTM 索引分区与 kd-tree 找最近邻算法的交叉证认方法”并开发出一套基于 MySQL 数据库的海量星表交叉证认程序[1][2]，初步解决了海量星表交叉证认的问题。其创新点是：（i）通过 HTM 索引将大星表分成多个小表分别证认，解决了内存限制问题，降低了计算量；（ii）证认时利用 kd-tree 找最近邻，提高了证认速度。

同时，该程序也存在一些不足：（i）程序完全采用 Java 语言编写，性能受限；（ii）算法没有并行化，不能充分发挥多核 CPU 的性能；（iii）纯命令行，没有图形界面，不便于用户使用。

为解决这些问题，我们仔细阅读了高丹等人的论文和程序代码，在深入分析其原理的础之上，对交叉证认程序进行了彻底的改进：（i）开发语言改用 Python 和 C 语言，充分利用 C 语言的高性能和 Python 语言的高开发效率；（ii）引入多核并行计算，充分发挥 CPU 的效能；（iii）开发了图形用户界面，以提高程序的易用性。本节将详细阐述改进的整体思路、实施方法和试验结果。

### 2.1.1 整体思路

**编程语言的选择**

Python[24]是一种面向对象的、解释型的程序设计语言，开源、免费，语法简明，功能强大，运行稳定，既可用来快速编写脚本程序，也可用来开发大规模的应用软件，应用十分广泛。Python 拥有极其丰富的类库，使开发变得容易；针对解释型语言性能普遍不高的通病，Python 提供了 C/C++ 编程接口，程序员可通过编写 C/C++ 扩展模块来提升程序性能；Python 支持多种图形库，便于开发图形用户界面。

由于诸多优点，越来越多的科研项目开始使用 Python 来开发应用程序。我们对 Python 语言各方面的表现也很满意，决定选用它作为主开发语言编写程序框架，以提升开发效率；计算密集部分用 C 语言实现，编译成 Python 模块供程序调用，以提升整体性能。

**Python 并行计算方法**

当前通用 CPU 的发展趋势是在一个物理芯片上集成多个处理器核，即多核 CPU。在这种 SMP 架构上编写并行程序最常用的方法是使用多线程。然而，由于 Python 虚拟机使用全局解释器锁（Global Interpreter Lock，简称 GIL）来互斥线程对 Python 虚拟机的使用，使得在同一时刻只能有一个线程访问 Python 虚拟机所提供的 API，从而导致在一个 Python 程序内即使有多个线程，也只能串行执行，不能发挥多核 CPU 的性能。要想使 Python 程序充分利用多核，只能通过“发起多个进程，由系统调度分配给多个处理器核”的方法来实现。

尽管目前已经有一些针对 Python 的并行库（如 pp、mpi4py 等），但在实际应用中，直接操控进程往往比使用封装好的库更加灵活方便。综合考虑之后，我们采用了直接操控进程的方法，利用 Python 的 subprocess 模块发起多个子进程并行计算，在保证较高灵活性的同时，大幅提升了程序性能。

**分割星表数据**

HTM 是一种多层次的、递归的球面分割方法，可将天球分成多级三角网格，每个网格都有一个编号，称为 pcode。利用 HTM 可以将一个大星表从逻辑上分割为多个小星表，具体做法如下：

对天球进行 N 级 HTM 划分。将星表存入数据库，转化为一个数据库表，每条表记录对应星表中一个源，每个表字段对应星表的一个属性列。入库时，给星表增添一个名为 pcode_htmN 的属性列（具体到数据库则是增加一个表字段），对于星表中任一源，pcode_htmN 属性的值为该源在天球上所处的 N 级 HTM 网格的 pcode。这样，就可以根据 pcode_htmN 属性值的不同将星表中的源进行分组。具有相同 pcode_htmN 值的源归为一组，每一组即为一个小星表。这样就把原来的一个星表划分成了多个小星表。得益于 HTM 划分方式的递归性以及 pcode 命名规则的规律性，只要进行了 N 级划分，就可以很容易地得到 M（M<N）级划分的结果。

对 pcode_htmN 字段建立 B-tree 索引，以提升查询性能。

高丹等人[1][2]正是利用上述方法将大星表分割为小表后分别证认，从而解决了内存容量不足的问题，并降低了计算量。事实上，星表分割也为并行计算做好了准备——只要将划分出的小表分发给多个证认子进程同时处理，即可在进程层面上实现并行计算。

基于球面分割方法（不论 HTM 还是 HEALPix）的星表数据划分必然导致一个新问题，即证认时在分块边缘上会有一定数量的漏源。不过，由于漏源数量与总数相比微乎其微，所以未对之进行处理。

### 2.1.2 程序实现

并行化的交叉证认程序流程如图 2-1 所示。主进程（xmatch.py）和子进程（worker.py）是两个独立的 Python 程序。在 xmatch.py 中，通过 subprocess 模块多次调用 worker.py，来发起多个计算子进程（具体个数可由用户指定）。每个进程中都有一个 Python 虚拟机在运行，而 GIL 只作用在单个 Python 虚拟机内部。这些进程由操作系统统一调度，可以分配到多个 CPU 核上，并行执行。

为提高性能，xmatch.py 中的“计算所有 pcode”函数以及 worker.py 中的证认函数均以 C 语言实现，编译成 Python 模块使用。

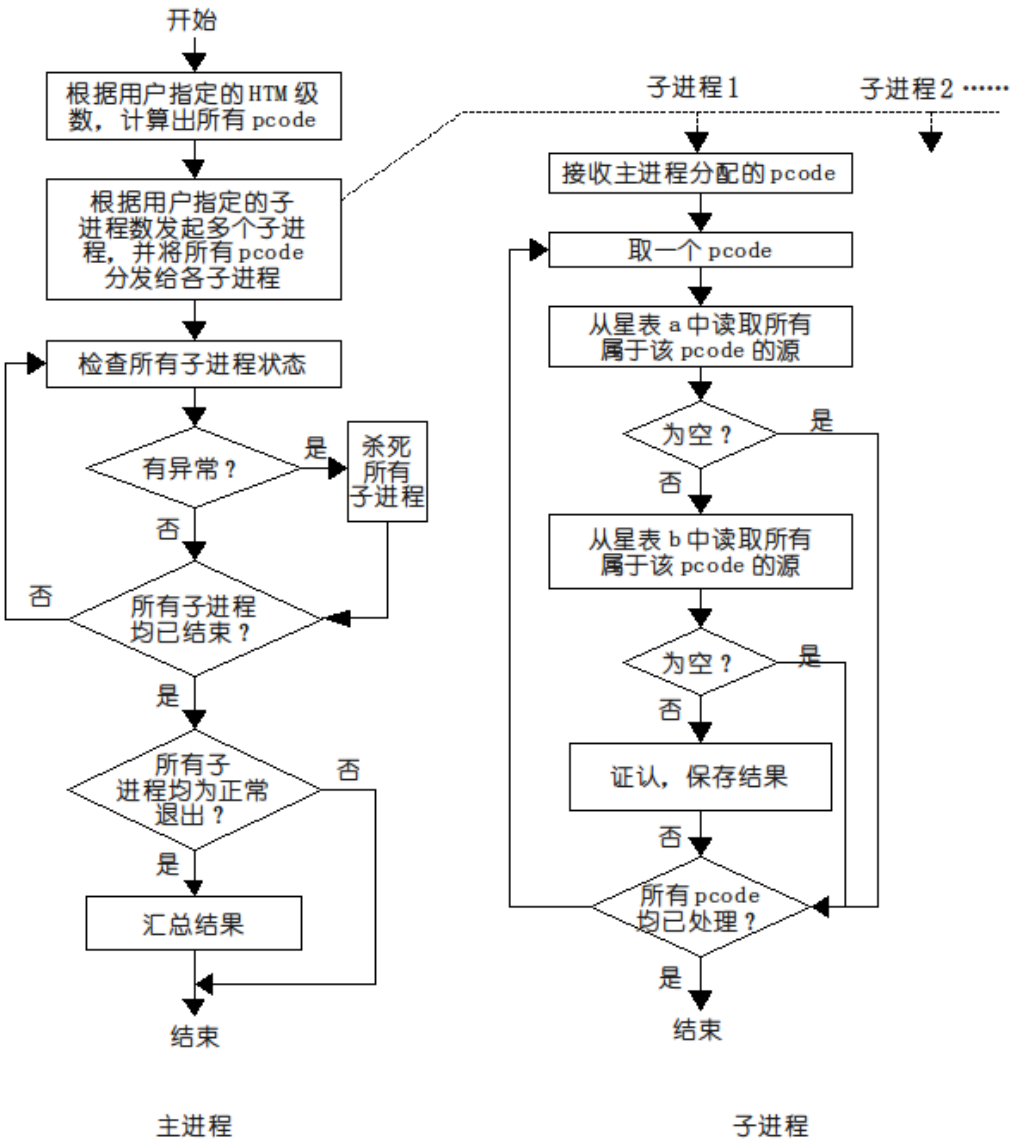


图 2-1 并行化的交叉证认程序流程图

### 2.1.3 图形用户界面

之前高丹等人开发的程序没有图形界面，只能通过命令行操作，不易上手。为方便用户使用，我们用 pygtk 开发了图形界面。考虑到在某些应用场景中 Linux 系统没有 x 环境，我们采取了“图形界面和主程序尽可能独立”的架构设计方案：图形界面只是一张“皮”，其作用是协助用户指定各个参数，并收集起来，按照一定规范生成配置文件，供主程序使用；主程序可以被图形界面调用，也可以手动以命令行模式运行，从配置文件中读取所需的参数，进行交叉证认。也就是说，主程序并不依赖于图形界面，只要有合适的配置文件，就能工作。即使没有图形界面，用户也可以手动编辑配

置文件。这就保证了程序有较好的适应能力。

程序运行界面如图 2-2 所示。


图 2-2 交叉证认程序图形界面

### 2.1.4 可靠性验证

为验证改进后的交叉证认程序的可靠性，我们进行了模拟试验：拿 FIRST 星表（811,117 行）作为“原材料”，对其 ra、dec 字段人为地加上一个绝对值不大于 2 角秒的随机偏差，再与原星表进行交叉证认，看结果如何。

我们分别设定匹配半径为 2.5 角秒和 3 角秒、HTM 级数为 3、4 和 5 进行证认。证认结果如表 2-1 所示。

可以看到，几乎所有数据都正确地证认上了——匹配半径取 2.5 角秒时，证认率达到 98%以上；而当匹配半径取 3 角秒时，证认率更是接近 100%；并且证认错误率非常低。可见改进后的交叉证认程序是可靠的。

同时还可以看到，随着 HTM 级数升高，证认结果有所减少。这是符合预期的：

HTM 级数增高意味着天球面分块数量增多，边界必然增多，从而导致更多的漏证。

表 2-1 模拟证认试验结果

| 匹配半径 | HTM 级数 | 证认结果条数 | 证认率 | 证认错误条数 |
|---|---|---|---|---|
| 2.5 角秒 | 3 | 800359 | 98.67% | 7 |
| | 4 | 800326 | 98.67% | 7 |
| | 5 | 800234 | 98.66% | 7 |
| 3 角秒 | 3 | 811063 | 99.99% | 10 |
| | 4 | 811029 | 99.99% | 10 |
| | 5 | 810935 | 99.97% | 10 |

### 2.1.5 性能测试及分析

我们使用以下两个星表进行性能测试：SDSS DR6 部分数据，共 100 106 811 行，作为星表 A；2MASS 星表，共 470 992 970 行，作为星表 B。测试中使用了两台计算机：一台作为数据库平台，储存星表数据；另一台作为计算平台，运行交叉证认程序，并通过百兆以太网访问数据库平台，读取星表数据。具体配置如表 2-2 所示。

表 2-2 试验平台软硬件配置

| | CPU | 内存 | 操作系统 | 内核 | 应用软件 |
|---|---|---|---|---|---|
| 数据库平台 | 双路 Intel(R) Xeon(TM) CPU 3.20GHz | 8G DDR2 | Red Hat Enterprise Linux AS release 4 (Nahant Update 2) | Linux version 2.6.9-22.ELsmp | Mysql：Ver 14.12 Distrib 5.0.22 |
| 计算平台 | 双路 Dual-Core AMD Opteron(tm) Processor 2212 | 4G DDR2 | Ubuntu 9.04（更新至 2009 年 10 月 20 日） | Linux version 2.6.28-15-generic | Python：2.6.2<br>gcc：4.3.3 |

考虑到 HTM 级数和证认子进程数均可调，我们采用如下测试方法：首先，固定 HTM 级数，取不同的子进程数，分别证认；然后，固定子进程数，取不同的 HTM 级数，分别证认。

**试验 1：HTM 级数固定，子进程数取不同值**

表 2-3 列出了 HTM 级数取 8、子进程数取不同值时的运行情况。其中，“加速

比”是相对于子进程数为 1 时的速度，“CPU 使用率”指证认过程中计算平台 CPU 的平均使用率。可以看到，随着子进程增多，加速比逐渐升高；子进程数为 16 时，加速比最高（7.74），证认速度最快（607 秒）；子进程数继续增加时，加速比则开始下降。图 2-3 给出了加速比随子进程数变化的曲线图。

表 2-3　HTM 级数取 8、子进程数取不同值的运行情况

| 子进程数 | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 耗时/秒 | 4696 | 2486 | 1327 | 869 | 737 | 689 | 637 | 614 | 607 | 625 | 663 | 729 | 810 | 908 | 954 | 1000 |
| 加速比 | 1.00 | 1.89 | 3.54 | 5.40 | 6.37 | 6.82 | 7.37 | 7.65 | 7.74 | 7.51 | 7.08 | 6.44 | 5.80 | 5.17 | 4.92 | 4.70 |
| CPU 使用率 | 15% | 28% | 51% | 57% | 65% | 68% | 70% | 72% | 73% | 72% | 70% | 63% | 57% | 51% | 48% | 46% |

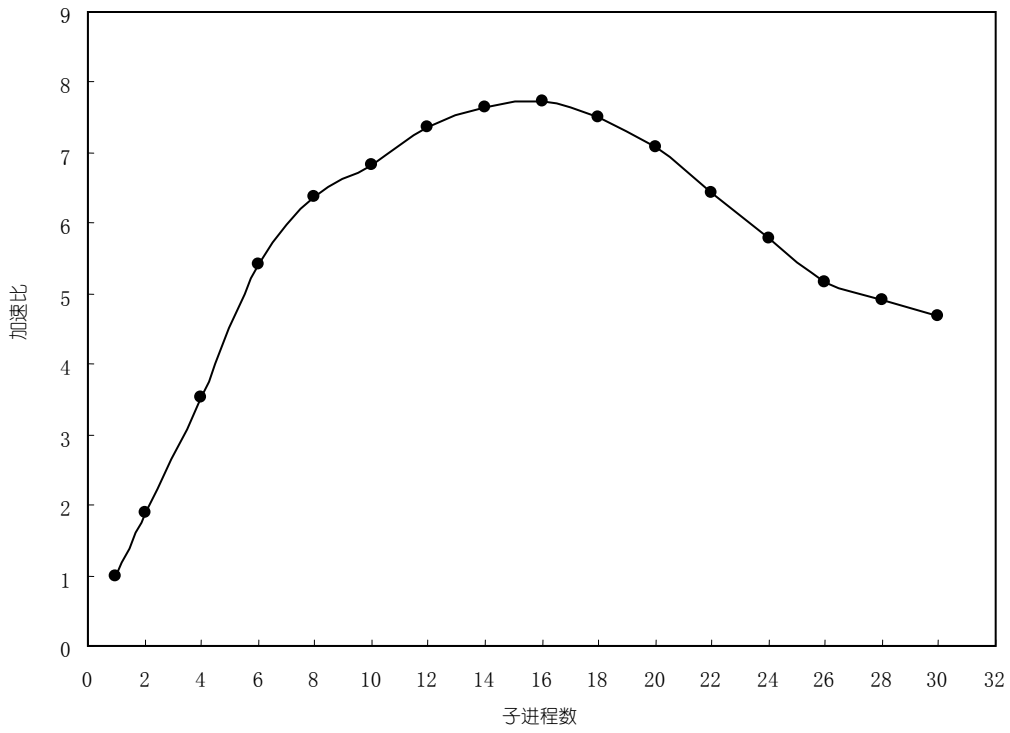


图 2-3　加速比随子进程数变化的情况

这个结果是符合预期的。子进程从一个增加到多个时，可以充分利用多个 CPU 核心，因而性能得以提升。另一方面，在一个子进程内，既有 I/O（读数据库），也有证认计算。当进程执行 I/O 操作、等待来自数据库的数据时，可把 CPU 让给其它需要计

算的进程。从这方面考虑，增加子进程也利于 CPU 的充分利用。这也可以解释为何子进程数为 4（等于 CPU 核心数）时并不是性能最高点，而是直到增加到 16 个（CPU 核心数的 4 倍），才达到性能最佳。子进程数超过 16 后，加速比开始下降，通常并行计算中出现这种情况的原因是：16 个子进程已经占满了 CPU，再增加子进程，徒增调度、切换的开销，因而性能不会提升，反而下降。然而，从表 2-3 可以看出，子进程数为 16 时 CPU 使用率并未达到 100%，说明 CPU 尚有余力。可见性能下降并非上述原因。由于每个子进程都要读数据库，子进程越多，发起的数据库连接就越多，因此我们推测，性能下降是因为 Mysql 数据库在连接数过多时查询效率降低，致使数据不能及时取出，CPU 花费更多时间等待数据，从而影响了证认速度。由此推测，如果能进一步提高数据库性能，证认速度还可以提高。

**试验 2：子进程数固定，HTM 级数取不同值**

表 2-4 列出了子进程数取 16 时 HTM 级数取不同值的运行情况（其中“CPU 使用率”指证认过程中计算平台 CPU 的平均使用率）。可以看出，随着 HTM 级数的增加，CPU 使用率呈降低趋势。这个结果与预期相符。HTM 级数越高，星表分块就越多，则每块中包含的行数就越少。由于证认函数计算量为 O(N2)，因此行数（N）的减少会大大降低计算量，CPU 使用率随之下降。还可以看到，HTM 级数为 8 时性能最好；降至 7 或升至 9，性能都会下降。注意到三种情况下 CPU 使用率均未达到 100%，因此我们推测，很可能是 I/O 效率的不同导致了性能差异。证认程序每次从数据库读取一个分块的数据。HTM 级数由 8 升至 9 时，星表分块数增加 3 倍，读取数据库的次数也增加 3 倍。查询次数过多导致读取数据的效率降低，从而影响了证认速度。反之，HTM 级数由 8 降至 7 时，分块数减至 1/4，数据库查询次数也降至 1/4，但同时每个分块中包含的行数增为原来的 4 倍左右，则每次查询需要提取更多的数据，由此增加的负担超过了查询次数降低带来的益处，从而性能下降。总之，就我们的测试环境而言，8 级 HTM 划分是最佳平衡点。对于不同的测试环境，此值可能会有差异。

表 2-4 子进程数取 16、HTM 级数取不同值的运行情况

| HTM 级数 | 耗时/秒 | CPU 使用率 |
|---|---|---|
| 7 | 712 | 91% |
| 8 | 605 | 75% |
| 9 | 808 | 56% |

**性能对比**

从高丹等人的论文[1]中看到，用改进前的程序对 FIRST 星表（811,117 行）和 2MASS 星表（470,992,970 行）进行交叉证认，耗时 407 分钟。而用我们改进后的程序对相同的星表进行交叉证认，仅耗时 2 分钟，加速比高达 200 倍。此处需要说明，高丹等人的论文中未给出详细的硬件信息，因此我们难以进行精准的程序性能对比。根据其论文撰写时间（2008 年）来推测，其测试平台应该是 2006 年左右的硬件产品，而我们使用的是 2007 年的服务器，硬件水平可能略高一些。但即便如此，200 倍的加速比也是十分可观的。有理由相信，更换编程语言、引入并行计算对程序性能的提升起到了决定性作用。

表 2-5　与高丹等人的程序对比

| | 星表 A | 星表 B | 分割方法 | 分块数 | 子进程数 | 耗时/分 |
|---|---|---|---|---|---|---|
| 高丹等人的程序 | FIRST（811,117 行） | 2MASS（470,992,970 行） | HTM（10 级） | 8,388,608 | 1 | 407 |
| 我们的程序 | | | HTM（6 级） | 32,768 | 16 | 2 |

此外，我们还与 Zhao 等人[3]的程序进行了对比。他们的程序使用 C 语言+MPICH2 开发，选用 HEALPix 方法分割星表数据，并通过 MPI 编程实现并行计算，在一台四核服务器（CPU: Quad-core Xeon X3323，Memory: 2.0 G DDR2 667 MHz，OS: Linux Fedora 10）上对 SDSS DR6 星表部分数据（100,106,811 行）和 2MASS 星表（470,992,970 行）进行交叉证认，耗时 32 分钟[3]。而我们的程序在水平相当的硬件平台（CPU 核数相等、内存带宽相同）上对同样的数据进行交叉证认，仅耗时 10 分钟，性能优势十分明显。推测原因: (ⅰ) 对方的程序子进程数偏少，未能充分发挥 CPU 和数据库的性能; (ⅱ) 对方程序对分块边缘漏源问题进行了处理，增加了计算量。

表 2-6 与 Zhao 等人的程序对比

| | 星表 A | 星表 B | 分割方法 | 分块数 | 子进程数 | 耗时/分 |
|---|---|---|---|---|---|---|
| Zhao 等人的程序 | SDSS DR6 部分数据（100,106,811 行） | 2MASS 星表（470,992,970 行） | HEALPix（13 级） | 805,306,368 | 3 | 32 |
| 我们的程序 | | | HTM（8 级） | 524,288 | 16 | 10 |

### 2.1.6 小结

我们对交叉证认程序的改进主要着眼于性能和易用性两个方面，而实践表明，我们的努力是有成效的：改进后的程序性能大幅提升，用户使用门槛大大降低，基本上可以满足天文工作者快速获得多波段交叉证认数据的需求。我们提出的“Python 多核并行计算方法”用于海量星表交叉证认收到了非常好的效果，对于其它领域的海量数据处理也有一定的参考价值。

同时我们也很清楚，改进后的交叉证认程序虽然比以前有了很大进步，但还远远称不上完美，还有大量的完善工作需要去做，比如性能的进一步提升、证认算法的改进等等，我们将在最后一章中加以总结。

## 2.2 交叉证认辅助程序的开发

如前所述，交叉证认程序是基于 MySQL 数据库的，用于证认的星表需要预先存入数据库中，并建立 HTM 索引。而实际应用中，星表可能以文本形式存放在本地文件系统中，也可能已经入库但没有 HTM 索引，这样一来就不能用本程序进行交叉证认。为解决以上问题，我们特地编写了“本地星表入库”程序和“创建 HTM 索引表”程序，协助用户将文本星表入库或为已入库星表增加 HTM 索引。

另一方面，交叉证认的结果是以文本形式存放在当前用户主目录中，其中只记录了天体源的关键线索（如 sourceID、赤经、赤纬等），并没有详细的信息。之所以这样做，一方面是因为不同的用户可能需要不同的信息（字段），另一方面，用于证认的 HTM 索引表中很可能只有标识性字段而没有详细信息，因而无从提取。针对证认结果，我们编写了一个“证认数据提取”程序，以协助用户提取需要的信息。

这三个程序都有图形界面，仍旧采用了“图形界面和主程序尽可能独立”的设计思路。

### 2.2.1 “本地星表入库”程序

这个程序可以协助用户将存放在本地的星表上传到 MySQL，并自动添加 pcode 字段、建立 HTM 索引，以便进行交叉证认。

用户需要为要上传的星表撰写一个 readme 文件，其格式要求如下：第一行为各列

的列名（即数据库表中的字段，请参照 MySQL 对字段命名的要求），以一个或多个空格分隔；第二行与第一行相对应，为各字段的数据类型（如 int、bigint、double、float、varchar 等，具体请参考 MySQL 数据类型相关文档），同样以一个或多个空格分隔；内容中不能有引号；字段不能为空或 null。以下是一组样例：

数据文件——

182, 338. 666367, 0. 69091709, 640771515, 338. 666314, 0. 691014, 1. 230754809e-08
182, 338. 666367, 0. 69091709, 640771512, 338. 663028, 0. 691848, 1. 202219342e-05
182, 338. 666367, 0. 69091709, 640771522, 338. 67057, 0. 69072, 1. 76956484682e-05
174, 315. 058806, 0. 74610661, 641319283, 315. 058803, 0. 746198, 8. 365822099e-09
174, 315. 0588066, 0. 74610661, 641319284, 315. 059748, 0. 750745, 2. 240070745e-05
174, 315. 0588066, 0. 74610661, 641319286, 315. 0601619, 0. 750942, 2. 521783454e-05
175, 315. 3802678, 10. 88098489, 707252508, 315. 380042, 10. 880242, 6. 02871192e-07
175, 315. 3802678, 10. 88098489, 707252503, 315. 3769697, 10. 883289, 1. 6184731e-05
166, 258. 1987561, 60. 76756876, 996989838, 258. 198725, 60. 76772297, 2. 47634101e-08
166, 258. 1987561, 60. 76756876, 996989837, 258. 1986997, 60. 768075, 2. 59437377e-07
166, 258. 1987561, 60. 76756876, 996989836, 258. 1983779, 60. 76861197, 1. 2313366e-06
. . .
. . .

Readme 文件——

| id_a | ra_a | dec_a | id_2 | ra_2 | dec_2 | distance |
|---|---|---|---|---|---|---|
| bigint | double | double | bigint | double | double | double |

该程序目前仅支持 CSV 格式的文件。由于已实现了一个框架，在此基础上再添加对其它文件格式（如 FITS、ASCII 等）的支持是比较容易的。限于时间和精力，这些工作我们未能完成，只能留待将来去做了。

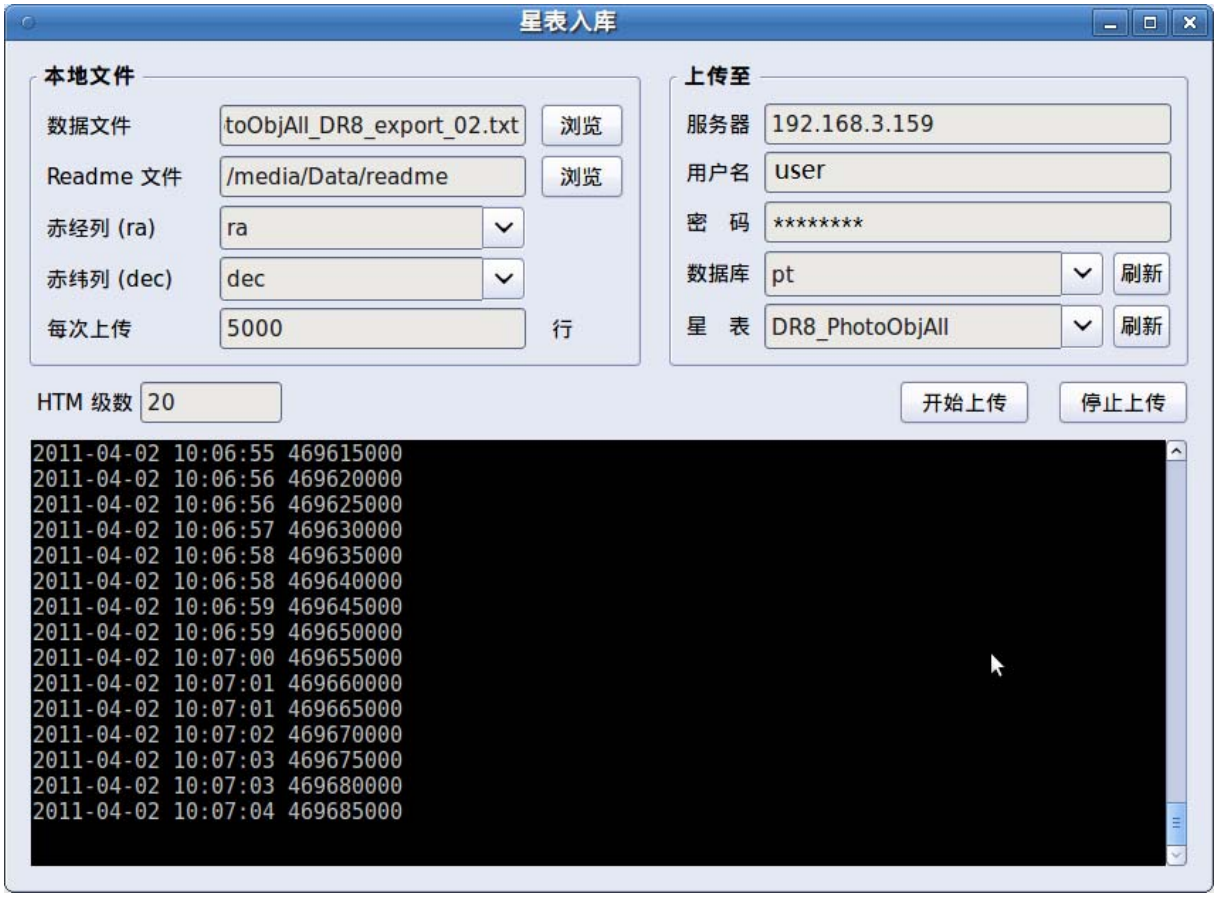


图 2-4 本地星表入库程序运行界面

### 2.2.2 “创建 HTM 索引表”程序

此程序可以为已经存入 MySQL 数据库但没有 HTM 索引的星表创建一个 HTM 索引表。如图所示，左边为原始星表，右边为新创建的带有 HTM 索引的表。程序内部工作流程为：首先创建一个包含 pcode 列的空表，然后逐行读取原始星表，根据每一条记录的赤经和赤纬，计算其所属 pcode 并写入新表。新表建立完成之后，就可以用来进行交叉证认了。

为提高效率，程序中使用了多线程技术。设置了 3 个线程：一个负责读取原表，另外两个计算 pcode 并将处理后的数据上传到新表。粗略观察，使用多线程后，整体效率提升了 2~3 倍。


图 2-5 创建 HTM 索引表程序运行界面

### 2.2.3 “证认数据提取”程序

此程序可以根据证认结果从原星表中提取用户指定的列（字段）。交叉证认结果保存在当前计算机 ~/xmatch_workspace 目录中，并以开始证认的时刻命名。点击“浏览”指定保存证认结果的目录（即任务目录），程序会自动读取相关信息，并加载。接下来，用户指定要提取的星表、字段，点击“开始”按钮，即可开始提取。提取出的数据默认保存在任务目录下 result 文件中。程序还会自动生成配套的 readme 文件，以备有需要时可以利用“本地星表入库”程序将 result 上传到数据库。


图 2-6 证认数据提取程序运行界面

## 2.3 NED、SIMBAD 自动证认工具的实现

NASA_IPAC 河外星系数据库（NASA_IPAC Extragalactic Database，简称 NED）[18]和 SIMBAD 天文数据库（SIMBAD Astronomical Database，简称 SIMBAD）[17]是天文工作者经常用到的网络数据资源，特别是经常用于交叉证认。但由于方方面面的限制，当要证认的数据比较多时，就不太方便了。为此我们调研了 NED 和 SIMBAD 的各种查询方式，对比、分析之后，编写了两个小程序来实现自动证认。

### 2.3.1 NED 自动证认

对于基于坐标的交叉证认，NED 网站最初只提供了两种方式：(i)通过网页提交一个待查询坐标，以网页形式返回结果；(ii)用户将多个待查询坐标存入文本文件，通过电子邮件提交给 NED，NED 自动处理后，回复一封邮件提醒用户下载证认结果——这

种方式称为“NED Batch Jobs”。

第二种方式可以进行批量查询，但对用户提交的文件有行数限制，当欲证认的星表较大时，需要拆分成多个小表，逐一提交、下载证认结果。这个过程不便于借助计算机程序实现自动化，只能人工处理。可以想见，一旦星表的行数达到一定规模（几十万行、几百万行甚至更多），这将是一个极其枯燥、繁琐甚至难以完成的工作。

于是我们尝试用第一种方式实现自动证认。设计流程如下：

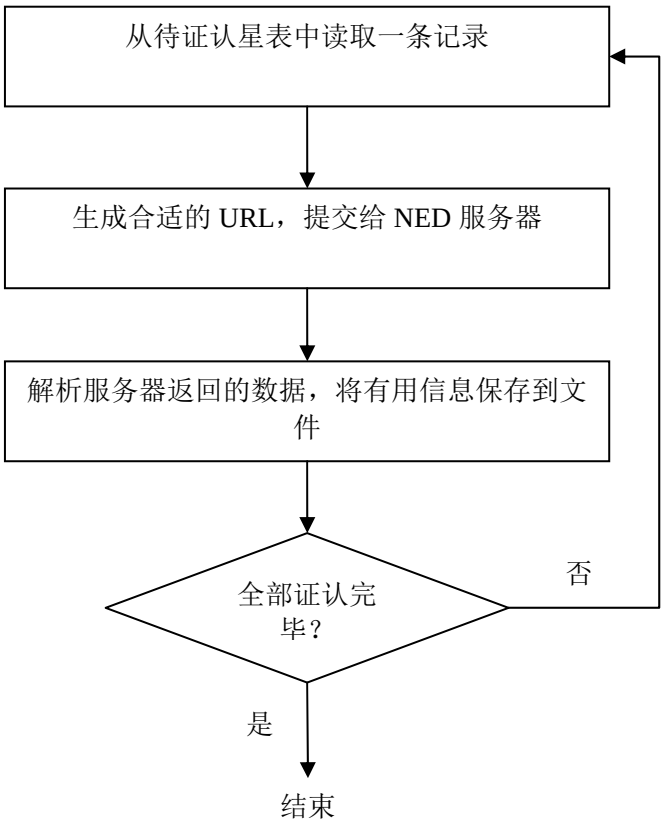


图 2-7　NED 自动证认流程

用 Python 语言按上述流程编写了一段程序，试验后发现，虽然可以工作，但证认速度难以令人满意。以 NVSS 星表为例，将该星表 1773484 条记录分为 6 组，同时开启 6 个进程与 NED 进行交叉证认，总共耗时 70 多个小时才完成。究其原因，主要是这种通过因特网进行的操作，大部分时间都消耗在网络连接与响应上，而每次只能提交一个查询坐标，势必需要大量的连接次数，效率低下也是很正常的。

令人欣喜的是，2010 年年底，NED 增加了第三种查询方式：提供了一个页面，用户可以将多个查询坐标粘贴到该页上的输入框中，点击提交后，证认结果以网页形式返回给用户。如图 2-8 所示。

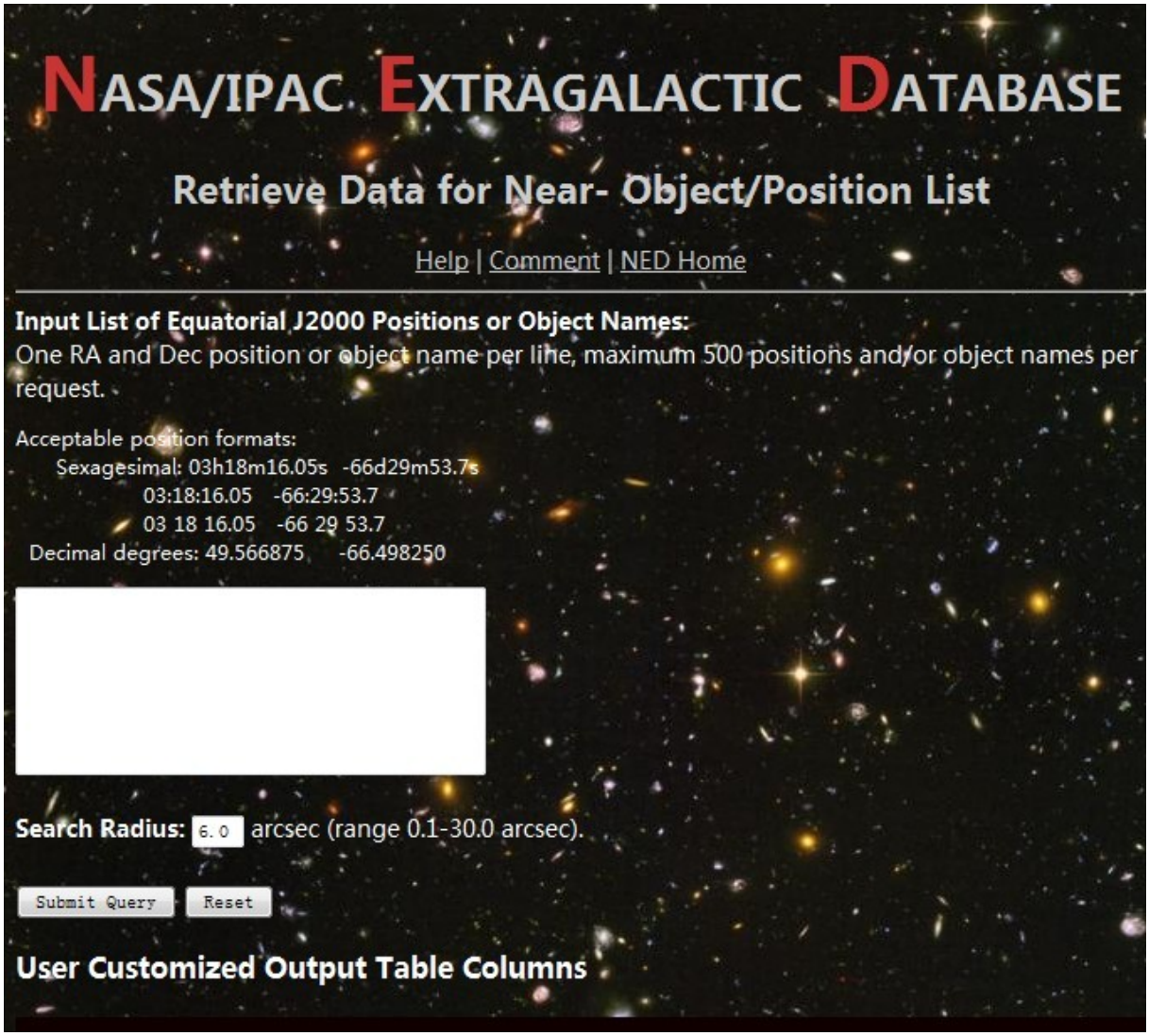


图 2-8 NED 批量查询页面

分析发现，用户提交的多个坐标实际上是被嵌入到一个特定的 URL 中，发送给 NED 服务器。知道了原理，就可以在程序中模拟这个过程，从而实现自动化的批量查询。于是我们重新设计了证认流程，如图 2-7 所示。按此流程重新编写了自动证认程序，性能果然大大改善，同样的数据，只需要 6 个小时即可证认完毕。

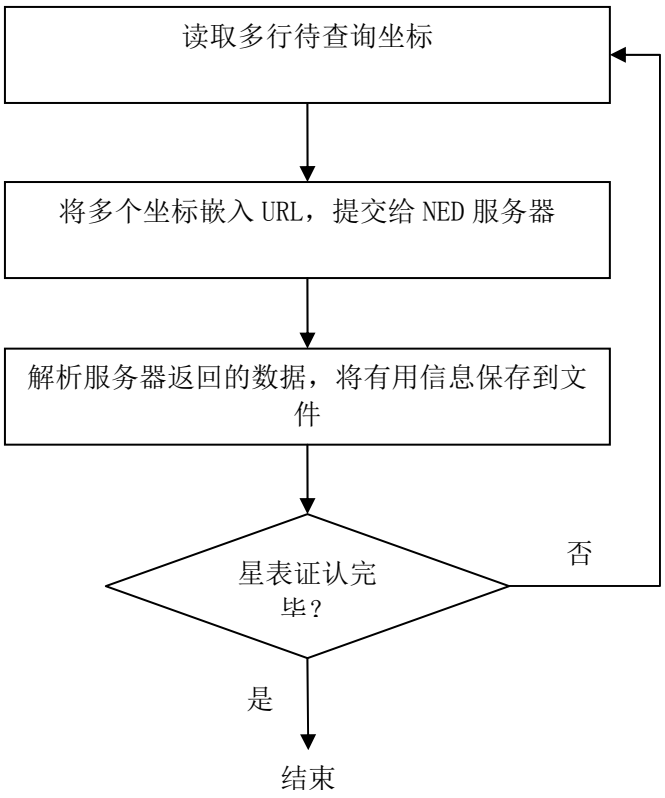


图 2-7　改进后的 NED 自动证认流程

### 2.3.2 SIMBAD 自动证认

SIMBAD 基于坐标的交叉证认与 NED 非常相似，也提供了三种方式：(i)通过网页提交一个待查询坐标，以网页形式返回结果；(ii)提交一个包含多个待查询坐标的文本文件，下载证认结果；(iii)将待查询坐标嵌入 SIMBAD 脚本，提交给 SIMBAD 网站，证认结果以网页形式返回。如图 2-9 所示。

其第三种方式与 NED 原理类似，只不过多了一层逻辑：先将查询坐标嵌入 SIMBAD 脚本，再将 SIMBAD 脚本嵌入特定 URL，提交给 SIMBAD 服务器。关于脚本中可用的控制语句、脚本命令和查询指令，SIMBAD 网站上给出了详细的说明。我们使用的脚本形式如下：

```
format object form1 "%IDLIST(1) | %COO(d;C) | %OTYPE(S)"
set radius  查询半径
echodata START
echodata 1
query coo  赤经  赤纬
echodata ;
echodata 2
```

```
query coo  赤经  赤纬
echodata ;
echodata 3
query coo  赤经  赤纬
echodata ;
...
...
```

按照与 NED 相似的流程编写了一个 SIMBAD 自动证认程序，用 NVSS 星表与 SIMBAD 进行交叉证认试验，同样分成 6 组，开启 6 个进程同时进行查询，完成证认耗时约 3 个小时。

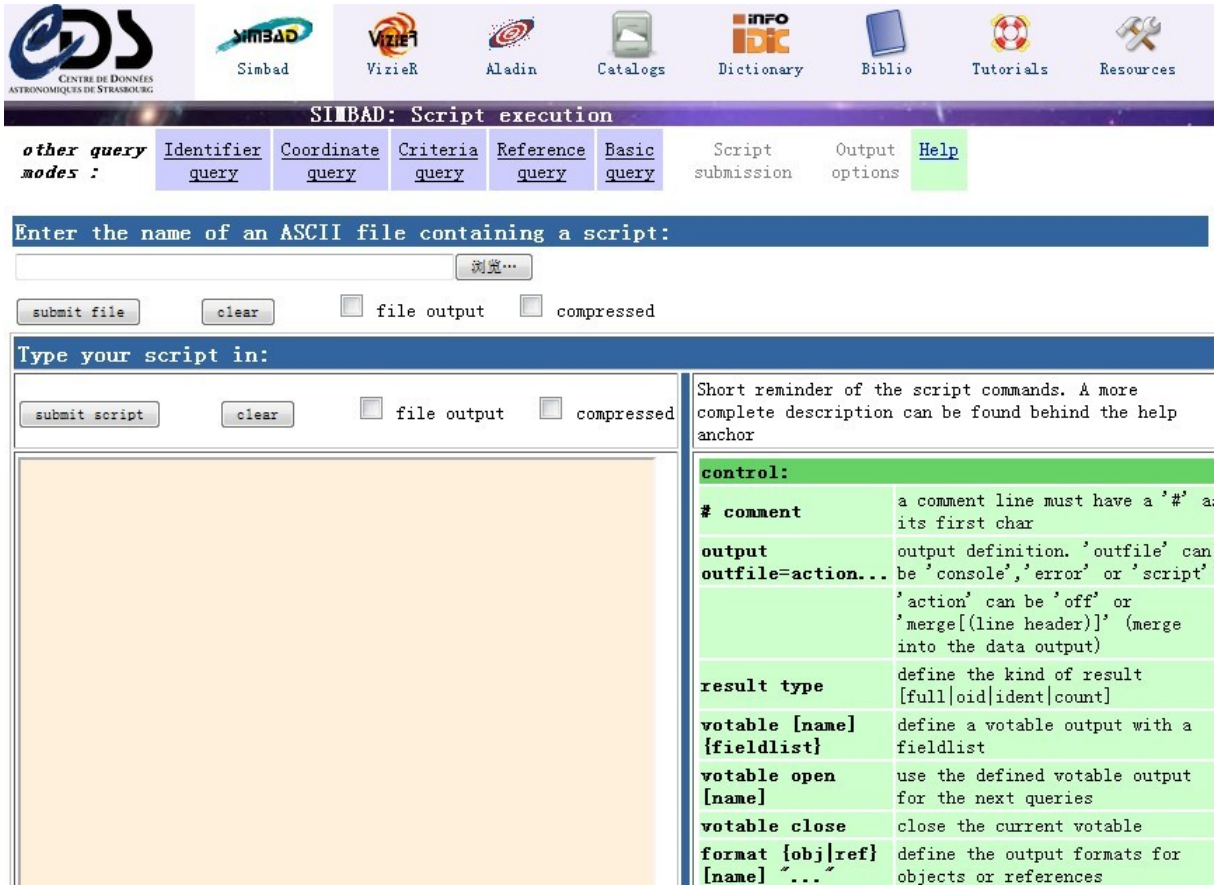


图 2-9 SIMBAD 脚本提交界面

## 2.4 应用实例：为郭守敬望远镜选取射电观测源

### 2.4.1 思路

郭守敬望远镜（原名“大天区面积多目标光纤光谱天文望远镜”，简称LAMOST）是1997年9月国家计划委员会批准的由中国科学院承担的国家重大科技基础设施建设项目，于2001年8月正式批准开工建设，2008年8月落成，总投资2.35亿元，于2009年6月4日通过国家验收。郭守敬望远镜的建成，突破了天文望远镜大视场与大口径难以兼得的难题，成为目前国际上口径最大的大视场望远镜，也是国际上光谱获取率最高的望远镜，是我国光学天文望远镜研制的一个里程碑，使我国在该领域处于国际领先地位，并将对宇宙起源、星系形成与演化、银河系结构、恒星形成与演化等研究做出重大贡献。在郭守敬望远镜建设过程中，工作人员攻克了大量世界级技术难题，如：创造性地应用主动光学技术，首次研制成功反射施密特改正板面形连续可变的光学系统；在国际上首次在大镜面上同时应用薄变形镜面和拼接镜面主动光学技术，并首次在一架望远镜中实现两块大拼接镜面；创造性地提出并实现了并行可控式光纤定位系统，解决了4000根光纤快速精确定位的技术难题，使望远镜一次观测的天体光谱数目比世界上现有望远镜提高近一个数量级。此外，8米直径的高精度大型地平式跟踪机架、多目标光纤光谱技术、海量数据处理等均为世界前沿技术，掌握的国家屈指可数。目前郭守敬望远镜正处于试观测阶段，预计很快将正式投入运行。

郭守敬望远镜的基本参数如下——

有效通光口径：4米

焦距：20米

视场角直径：5度（焦面线直径：1.75米）

光学质量：80%光能量集中在2.0角秒直径的圆内

光纤数：4000根

光谱覆盖范围：370-900纳米

观测天区：赤纬从-10度到+90度的24000平方度

光谱分辨率：1-0.25 纳米

光谱观测能力：以1纳米的光谱分辨率，在1.5小时曝光时间内，极限星等达到20.5等

郭守敬望远镜的观测效率固然很高，然而在无数繁星面前，力量就显得微薄了，

因此需要精心挑选观测价值较高的目标源优先观测。天文学史上人们曾利用射电技术获得意义重大的新发现，包括超新星遗迹、射电星系、类星体、脉冲星和引力透镜源等，宇宙射电源背后经常蕴藏着人们尚不知晓的物理机制与天文过程，是一类很值得观测的天体目标。天文学界有两个常用的射电巡天星表：The NRAO VLA Sky Survey（简称 NVSS）星表[25]和 Faint Images of the Radio Sky at Twenty-Centimeters（简称 FIRST）星表[26]。NVSS 是一个 1.4GHz 射电巡天项目，覆盖赤纬-40 度以北的天区。NVSS 的主要数据包括 2326 个 4 度×4 度、具有 Stocke I、Q 和 U 参数的线性偏振图像的连续“立方体”（continuum "cubes"）组，以及一个包含近两百万独立源的星表。FIRST 巡天是一个正在运行的射电快照巡天，覆盖南北银极近一万平方度。巡天结束时，该巡天星表将包含约一百万个源，分辨率小于 1 角秒。FIRST 巡天以牺牲观测视场为代价，获得了比 NVSS 好的空间分辨率。最后具有 1.8 角秒的射电图集是由每一个中心点附近的 12 张图像叠加而成的。该巡天的星表是从射电图集中得到的，包括峰值流量、积分流量密度和由拟合二维的高斯图像得到的源的大小。近 50%的源在 POSS I 底片的极限内（E≈20）具有可见光对应体。在小于 5%的错误率下，V 达到 24 的源都可以光学证认。选择的巡天区域尽量与 SDSS 巡天一致。

我们决定从 NVSS 和 FIRST 射电巡天星表中为郭守敬望远镜选取一部分观测源。选取方法如下：首先，拿 NVSS/FIRST 星表分别与 SIMBAD、NED、SDSS DR7 SpecObjAll 星表进行交叉证认，通过证认结果判断 NVSS/FIRST 星表中每个射电源是否已经被判明类型，已经明确了类型且类型比较普通的，排除掉；再与 USNO-B 进行交叉证认，匹配上的，可以从 USNO-B 星表中获知其星等信息，亮度低于 20 星等的，排除掉；剩下的源中，满足赤纬 > -10 度的即为郭守敬望远镜可观测的源。

### 2.4.2 具体步骤

首先，用 NED、SIMBAD 自动证认工具分别对 FIRST 和 NVSS 星表进行证认。考虑到星表本身的精确度，对于 FIRST 中的源，证认半径取 3 角秒，NVSS 则取 15 角秒。证认结果如下——

FIRST 在 NED 中有对应体的 392462 个，没有对应体的 418655 个；

FIRST 在 SIMBAD 中有对应体的 260482 个，没有对应体的 550635 个；

NVSS 在 NED 中有对应体的 1708673 个，没有对应体的 64811 个；

NVSS 在 SIMBAD 中有对应体的 309114 个，没有对应体的 1464370 个。

然后，用“基于 MySQL 数据库的海量星表交叉证认程序”分别对 FIRST、NVSS 与 USNO-B、SDSS DR7 SpecObjAll 进行交叉证认。由于原数据库中这些星表都没有 HTM 索引，因此先用“创建 HTM 索引表”程序为这四个星表创建 HTM 索引表，再交叉证认。同样，FIRST 匹配半径取 3 角秒，NVSS 取 15 角秒。结果如下——

FIRST 在 USNO-B 中有对应体的 358334 个，没有对应体的 452783 个；

FIRST 在 SDSS DR7 SpecObjAll 中有对应体的 67638 个，没有对应体的 743479 个；

NVSS 在 USNO-B 中有对应体的 1075635 个，没有对应体的 697849 个；

NVSS 在 SDSS DR7 SpecObjAll 中有对应体的 46305 个，没有对应体的 1727179 个。

利用“证认数据提取”程序根据证认结果从原星表中提取所需字段：从 USNO-B 星表中提取星等，从 SDSS DR7 SpecObjAll 星表中提取天体类型。

接下来汇总证认结果。

FIRST 星表中共有 811117 个源，其中：

1. 在 SIMBAD、NED 和 SDSS DR7 specObjAll 中都没有对应体的有 365936 个，其中与 USNOB 有对应体的有 118404 个；

2. 在 SIMBAD、NED 和 SDSS DR7 specObjAll 中有对应体、但类型仅标注为射电源（RadioS 或 Radio）的有 224777 个，其中与 USNOB 有对应体的有 71910 个；

3. 在 SIMBAD、NED 和 SDSS DR7 specObjAll 中有对应体、且有明确的类型信息的有 220404 个。

从 1、2 中筛选出赤纬 > -10 度、星等 < 20 等的源 150521 个。

NVSS 星表中共有 1773484 个源，其中：

1. 在 SIMBAD、NED 和 SDSS DR7 specObjAll 中都没有对应体的有 58621 个，其中与 USNOB 有对应体的有 33445 个；

2. 在 SIMBAD、NED 和 SDSS DR7 specObjAll 中有对应体、但类型仅标注为射电

源（RadioS 或 Radio）的有 1520999 个，其中与 USNOB 有对应体的有 877024 个；

3. 在 SIMBAD、NED 和 SDSS DR7 specObjAll 中有对应体、且有明确的类型信息的有 193864 个。

从 1、2 中筛选出赤纬 > -10 度、星等 < 20 等的源 549355 个。

最终从 FIRST 和 NVSS 中共筛选出 150521 + 549355 = 699876 个待观测源，均可作为郭守敬望远镜观测目标。

目前郭守敬望远镜正处于试观测阶段。我们以工作人员给定的若干个试观测中心源为圆心、以 2.5 度为半径（对应郭守敬望远镜 5 度视场）划定范围，取范围之内的待观测源作为一个试观测输入星表提交给观测人员。图 2-10 展示了该星表在天球上面的分布情况。

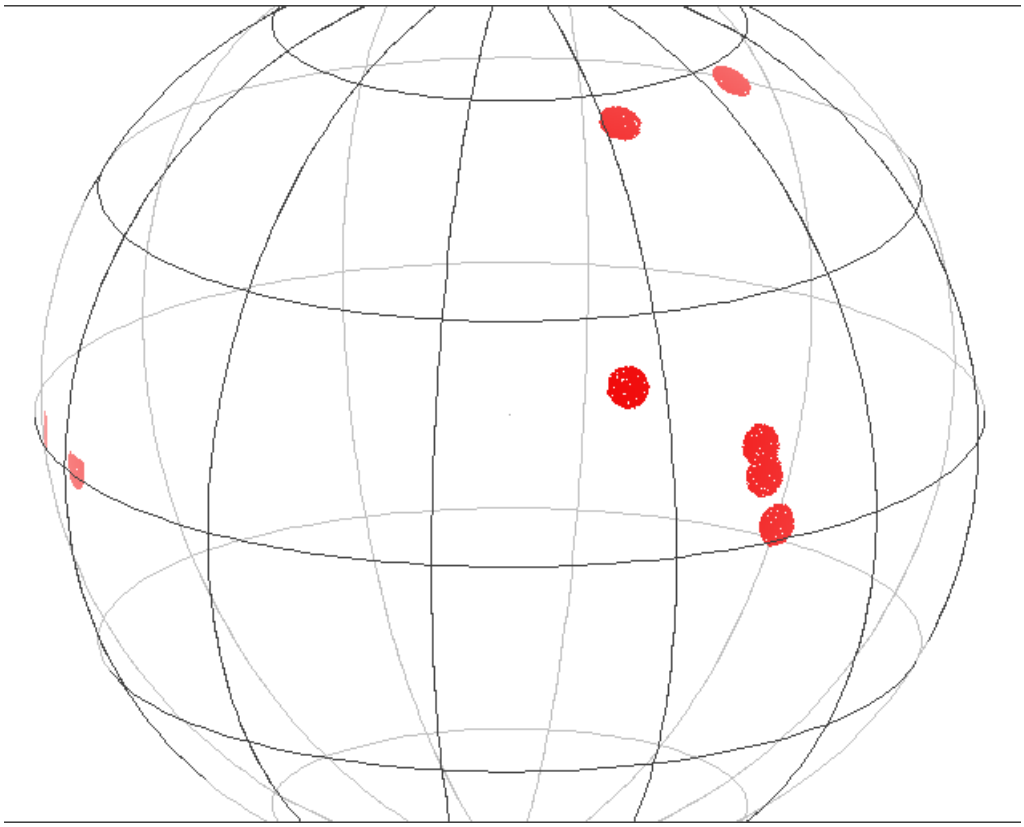

图 2-10 选出的试观测输入星表在天球面上的分布情况

## 2.5 本章小结

本章介绍了我们围绕“海量星表融合”这一课题所作的一系列工作：首先，对高丹等人开发的海量星表交叉证认程序进行了深入的剖析与改进，大幅度提升了其性能和易用性；而后，开发了几个辅助程序，来协助用户完成交叉证认前后的必要步骤；接下来，又针对天文工作者常用的 NED/SIMBAD 网络天文数据库开发了自动证认工具，以帮助天文工作者进行大数据量的快速证认，提高工作效率。最后给出了一个应用实例：综合利用上述工具从 FIRST 和 NVSS 射电巡天星表中为郭守敬望远镜(LAMOST)选取射电观测源。

# 第三章 CUDA 并行计算技术在天文研究中的应用

CUDA[21]是 NVIDIA（英伟达）[22]公司推出的通用并行计算架构，该架构使人们能够利用 GPU 进行通用计算，充分发挥 GPU 强大的浮点计算能力，解决复杂的计算问题。它包含了 CUDA 指令集架构（ISA）以及 GPU 内部的并行计算引擎，开发人员可以使用 C/C++/FORTRAN 语言来为 CUDA 架构编写程序，获得很高的运行性能。

目前 CUDA 已广泛应用于图形动画、科学计算、地质、生物、物理模拟等领域，也有越来越多的天文工作者将这一技术用于天文研究[7-10]。本章中，我们尝试在两个计算密集的天文问题中应用 CUDA 并行计算技术，以期得到较大幅度的加速效果。实践证明，这一努力是有成效的，CUDA 技术的应用使不同的应用程序获得了十数倍乃至数十倍的加速。

## 3.1 利用 CUDA 加速 kNN 分类算法

k 最近邻(k-Nearest Neighbor，kNN)分类算法是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：如果一个样本在特征空间中的 k 个最相似(即特征空间中距离最小)的样本中的大多数属于某一个类别，则该样本也属于这个类别。其中，k 是一个由用户指定的整数常量，其值依具体情形而定，不应过大，也不能太小。这里的“距离”，一般采用欧氏距离，也可以是曼哈顿距离、余弦距离等等。

kNN 分类算法虽然原理很简单，但在实际应用中却有着不错的分类效果，而且实现起来也很容易，因此被广泛应用于多个学科领域。在天文学研究中常常用它来进行天体目标分类[4-6]。不过 kNN 算法也有着与生俱来的短处，那就是计算量。原始版本的 kNN 算法在分类过程中需要计算每一条待分类样本与所有训练样本之间的距离，可以想见，当训练集比较大的时候，计算量会变得非常之大。幸运的是，这一算法具有较高的可并行性：计算距离时，各个训练样本之间是相互独立的，没有依赖关系，因此可以并行来做；同样，寻找 k 个最近邻时，也可以并行操作。这样的算法非常适在 CUDA 平台上实现并行化。

### 3.1.1 串行算法实现

我们用 C 语言实现了一个基于 CPU 的单线程 kNN 算法，流程如下：

1. 将训练集和测试集读入内存。
2. 对每一个待分类样本：
   （1）计算它与每一个训练样本的距离；
   （2）找到 k 个距离被测数据最近的训练样本；
   （3）根据这 k 个最近邻样本的类别，判断被测样本的类别。
3. 重复第 2 步，直到所有测试样本都被分类，算法停止。

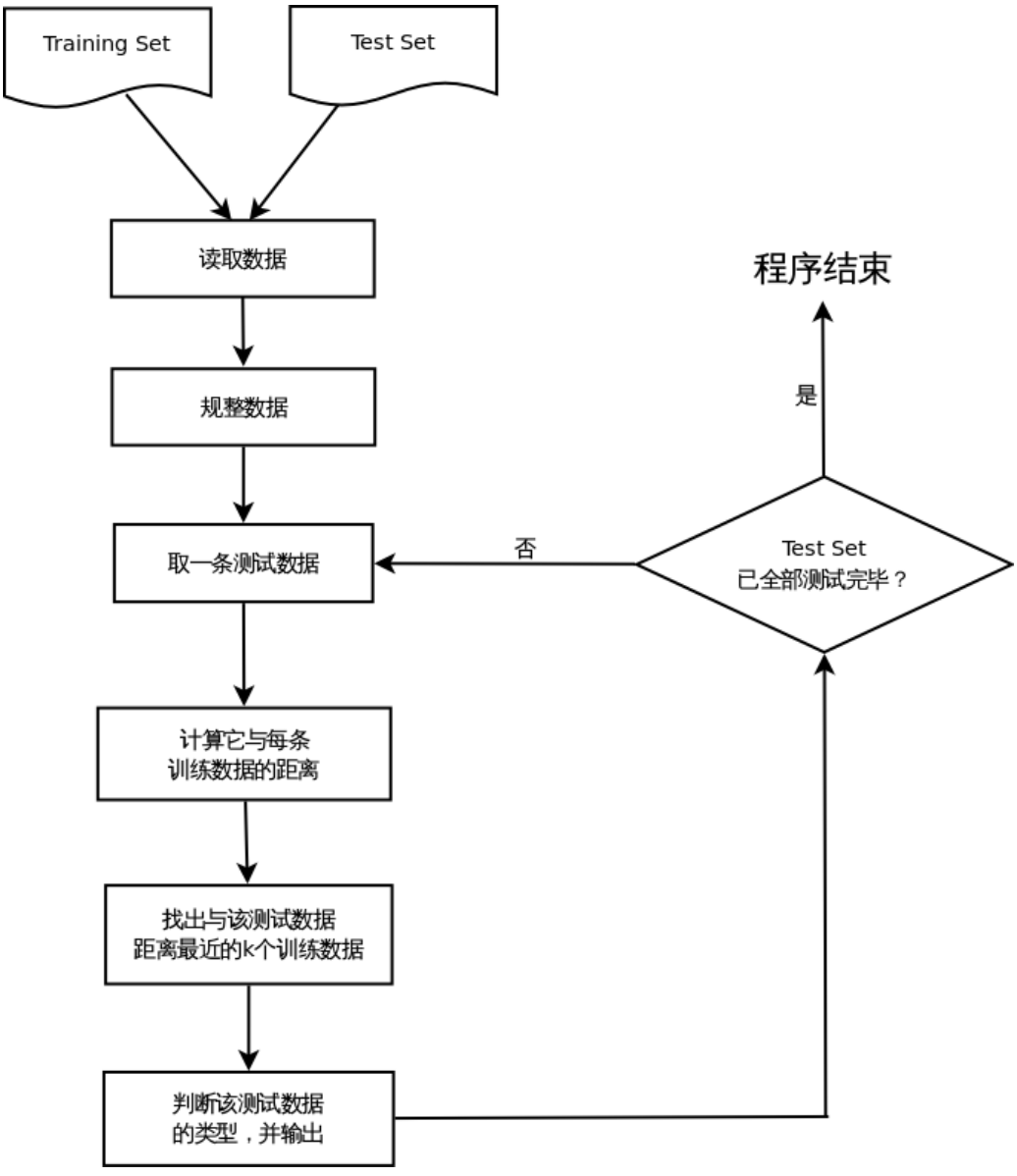


图 3-1 串行 kNN 算法流程图

假设有 m 个训练样本，n 个查询样本，样本空间为 d 维，可以算出：需要计算 n*m 次距离，计算复杂度为 O(nmd)；需要进行 n*k 次寻找最近邻的操作，计算复杂度为 O(nmk)。可见计算量是非常大的。

### 3.1.2 CUDA 并行算法

CUDA 程序比单纯的 CPU 程序要多一些开销：首先，需要在主机内存和 CUDA 设备之间传送数据；其次，数据传入 CUDA 设备之后，将发起多个 CUDA 线程对数据进行并行处理，这也需要一定的开销，不过，由于硬件的支持，发起线程的开销非常小。并非所有的计算都在 GPU 上进行。整个 CUDA 程序中，只有一部分函数在 GPU 上执行，这些函数被称为核函数（kernel functions）。这些函数一旦被调用，将在 N 个不同的 CUDA 线程中并行地执行 N 次。所有 CUDA 线程结束后，计算结果将被传回主机内存，由 CPU 进行后续处理。

一个支持 CUDA 的 GPU 内部含有多个流多处理器（Streaming Multiprocessor，简称 SM），而每个流多处理器又包含一定数量的流处理器核（Streaming processor cores），称为 CUDA 核心。这样的硬件架构在流多处理器层和 CUDA 核心层两个层次上提供了并行化能力，这在 CUDA 线程的组织方式上有所反映：一定数量的 CUDA 线程构成一维、二维或三维的线程块（block）；一定数量的线程块构成一维和二维的线程网格（grid）。程序运行时，一个线程块作为一个整体分配给某个流多处理器（SM），该线程块中的所有线程都将由此 SM 中的 CUDA 核心来执行。

同一线程块中所有线程可以通过核函数中设置的同步点来协调进度：只有所有线程都到达同步点之后，程序才会继续执行。这些线程可以通过共享存储器（shared memory）来共享数据，通过这种方式实现数百个线程的协同合作。而处于不同线程块的线程之间没有同步机制，因此编程时须注意，它们之间应该相互独立，不应有依赖关系。

由于显存容量的有限，CUDA 设备不能一次处理全部数据，只能分多次处理。我们的程序中，将被测样本分成了若干组，每次处理一组。对每一个测试样本，发起一个线程块来并行计算该样本与所有训练样本之间的距离。

图 3-2 展示了 CUDA 并行化的 kNN 算法的执行流程。

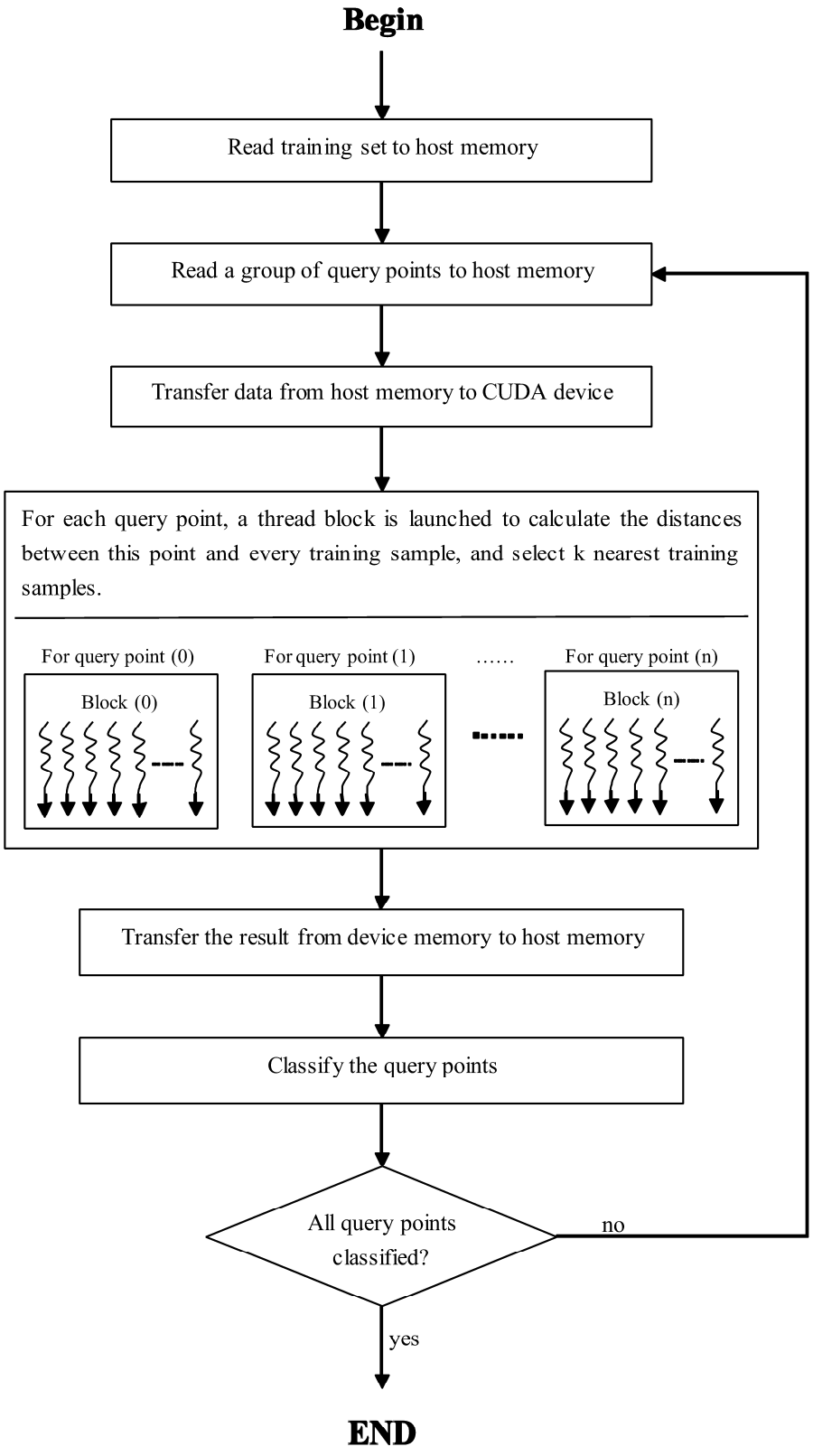


图 3-2 CUDA 并行 kNN 算法流程图

### 3.1.3 性能测试

**输入数据**

我们的输入数据取自斯隆数字巡天项目（Sloan Digital Sky Survey，SDSS）公布的数据。SDSS 始于 2000 年，以阿尔弗雷德·斯隆的名字命名，计划观测 25%的天空，获取超过一百万个天体的多色测光资料和光谱数据。斯隆数字巡天的星系样本以红移 0.1 为中值，对于红星系的红移值达到 0.4，对于类星体红移值则达到 5，并且希望探测到红移值大于 6 的类星体。SDSS 使用口径为 2.5 米的宽视场望远镜，测光系统配以分别位于 u、g、r、i、z 波段的五个滤镜对天体进行拍摄。这些照片经过处理之后生成天体的列表，包含被观测天体的各种参数，比如它们是点状的还是延展的，如果是后者，则该天体有可能是一个星系，以及它们在 CCD 上的亮度，这与其在不同波段的星等有关。另外，天文学家们还选出一些目标来进行光谱观测。目标的位置用钻孔的方式记录在铝板上，小孔的后面接有光纤，将目标天体的光引入摄谱仪。望远镜每次可以同时拍摄 640 个天体的光谱，每晚大约需要 6 到 9 块铝板对天体进行定位。SDSS 运行至今（SDSS-I：2000 年~2005 年；SDSS-II：2005 年~2008 年；SDSS-III：2008 年开始，尚未结束），已经获得了覆盖超过四分之一天空的深入的彩色图像，并建立了包含 930,000 多个星系和 120,000 多个类星体的 3 维空间图。

我们进行实验的时候，SDSS 发布的最新数据是 DR7。SDSS DR7 星表包括从观测图像中抽取出来的 3.57 亿多个天体目标，其中通过光谱观测判明类型的大约有 930,000 个星系、120,000 个类星体以及 460,000 个恒星。我们从中选取了 16000 个点源——包括 80000 个恒星和 80000 个类星体——作为训练集，另取 50000 个点源作为测试集，来进行试验；选取以下属性列作为输入模式：(psfMag u - psfMag g)，(psfMag g - psfMag r)，(psfMag r - psfMag i) 以及 (psfMag i - psfMag z)。

**测试平台**

所有测试都在一台桌面 PC 上进行，其配置如下——

CPU：AMD Phenom 8650 Triple-Core Processor，2.3 GHz

内存：DDR2 800，4.0 GB

操作系统：Ubuntu Linux 9.04， 32-bit

CUDA 设备：一块 GeForce GTX 260 显卡（基于 GT200 架构，拥有 27 个 SM、

216 个 CUDA 核心、896MB GDDR3 显存）

**关于浮点精度**

传统的 GPU 主要用于图像处理，尤其是实时 3D 运算，这些应用不需要太高的浮点精度。而科学计算要求较高的浮点运算精度，这是传统 GPU 不具备的。为满足科学计算的要求，NVIDIA 公司在 GT200 核心架构中增加了对 IEEE 754R 标准 64 位双精度浮点运算的支持，然而与单精度相比，双精度运算要慢得多。

为使测试更有说服力，我们分别开发了使用单精度浮点数和双精度浮点数 CPU 串行程序和 CUDA 并行程序，分别对这四个程序进行性能测试。

**测试结果**

kNN 分类算法速度与测试集的大小和 k 的取值都有关系。在我们的试验中，k 被设定为 111。分别用四个程序对前述数据进行分类，耗时情况如表 3-1 所示。可以看出，对于相同数量的测试样本，双精度程序需要更长的运行时间，这一点在 CUDA 程序上尤其明显；而在测试样本数量相等、精度也相同的情况下，CUDA 并行程序要比 CPU 串行程序快得多。

表 3-1 各程序运行情况

| size of test-set | CPU-only (single-precision) (seconds) | CUDA (single-precision) (seconds) | CPU-only (double-precision) (seconds) | CUDA (double-precision) (seconds) |
|---|---|---|---|---|
| 2000 | 200.572893858 | 3.05481696129 | 221.654909134 | 4.78594517708 |
| 4000 | 412.81862092 | 5.04370093346 | 421.22219491 | 8.45708203316 |
| 6000 | 608.260256052 | 7.00852012634 | 638.151005983 | 11.915500164 |
| 8000 | 807.094192982 | 9.00995111465 | 845.077206135 | 15.7372920513 |
| 10000 | 996.773925066 | 10.9353950024 | 1072.26910305 | 19.3025858402 |
| 12000 | 1206.3944509 | 12.9364061356 | 1302.61053205 | 23.0230331421 |

图 3-3 显示了每个程序耗时增长的情况。分类过程消耗的时间随着测试样本的增多呈现线性增长的态势。

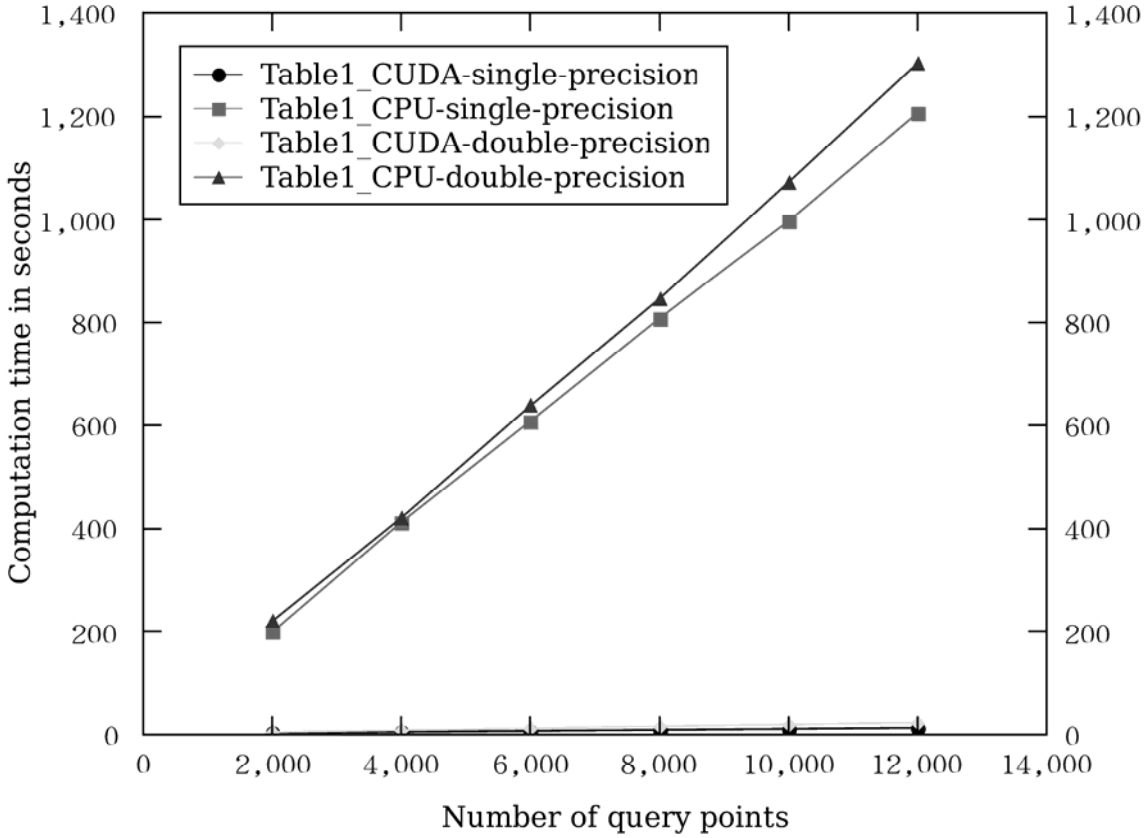


图 3-3 各程序耗时增长态势

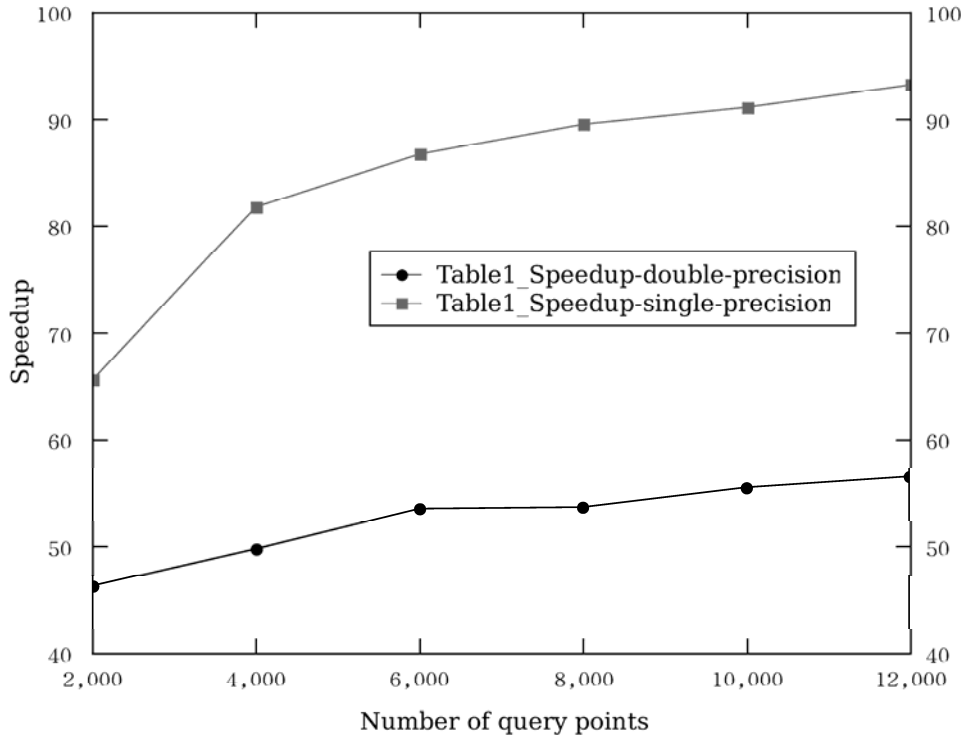


图 3-4 CUDA 程序相对于串行程序的加速比

图 3-4 显示了对不同数量测试样本进行分类时 CUDA 程序相对 CPU 串行程序的加速比。可以看到数据规模越大，越能显出 CUDA 并行计算的优势。其中一个原因是，计算量越大，GPU 就越能够集中精力进行计算，而不是把时间过多地花费在与主机交换数据上。

另一个现象是，相对于双精度程序，使用单精度浮点数能获得更高的加速比。然而单精度并不能满足所有科学计算的要求，有时候为了达到精度要求必须使用双精度浮点数，那么性能代价就不可避免了。

## 3.2 利用 CUDA 加速郭守敬望远镜抽谱算法

郭守敬望远镜（原名“大天区面积多目标光纤光谱天文望远镜”，简称 LAMOST）是 1997 年 9 月国家计划委员会批准的由中国科学院承担的国家重大科技基础设施建设项目，于 2001 年 9 月正式开工，2008 年 8 月落成，总投资 2.35 亿元，于 2009 年 6 月 4 日通过国家验收。现正处于试观测阶段，预计很快将正式投入运行。届时将成为世界上光谱获取率最高的望远镜，同时可观测目标 4000 个，每个观测夜可得到 2 万条光谱。

观测所得数据通过一系列软件进行加工处理，其中首要的一步便是“抽谱”，即，从 CCD 所成的像中将光谱信息抽取出来。这一过程涉及到大量浮点运算，是一个比较耗时的步骤。通过分析发现，算法中的某些部分具有较高的可并行性。于是尝试用 CUDA 将其并行化，充分发挥显卡强大的浮点运算能力，最终在 GTX260 显卡上获得了 17 倍左右的加速比（使用双精度浮点数）。

### 3.2.1 串行算法分析

**输入数据**

输入数据为 5 个矩阵——image、invvar、xcen、sigma 和 nord。这 5 个矩阵行数相同，均为 4136 行。

image 矩阵为一幅 CCD 成像数据，是一个 4136 行、4096 列的浮点数矩阵，对应 CCD 上 4136*4096 个像素。

invvar 矩阵为 image invert variance，列数与 image 相同。

xcen、sigma 和 nord 是与光纤相关的参数矩阵，其列数均为 250。

**处理流程**

实际处理过程是逐行进行的：每次从 5 个矩阵中各取一行进行处理，得出结果后，再取下一行，依次循环，直到将 4136 行全部处理完。

对每一行输入数据，需要计算每一根光纤在其中的投射信息。每个 CCD 对应 250 根光纤，因此要循环 250 次。

每一行输入数据都会产生一个 254 行、4096 列的 prflux 矩阵（中间结果），然后对 prflux 矩阵进行一系列矩阵、向量乘法，最终输出结果为一个 alpha 矩阵（254 行、254 列）和一个 beta 向量（254 维），供后续程序使用。

算法流程如图 3-5 所示。

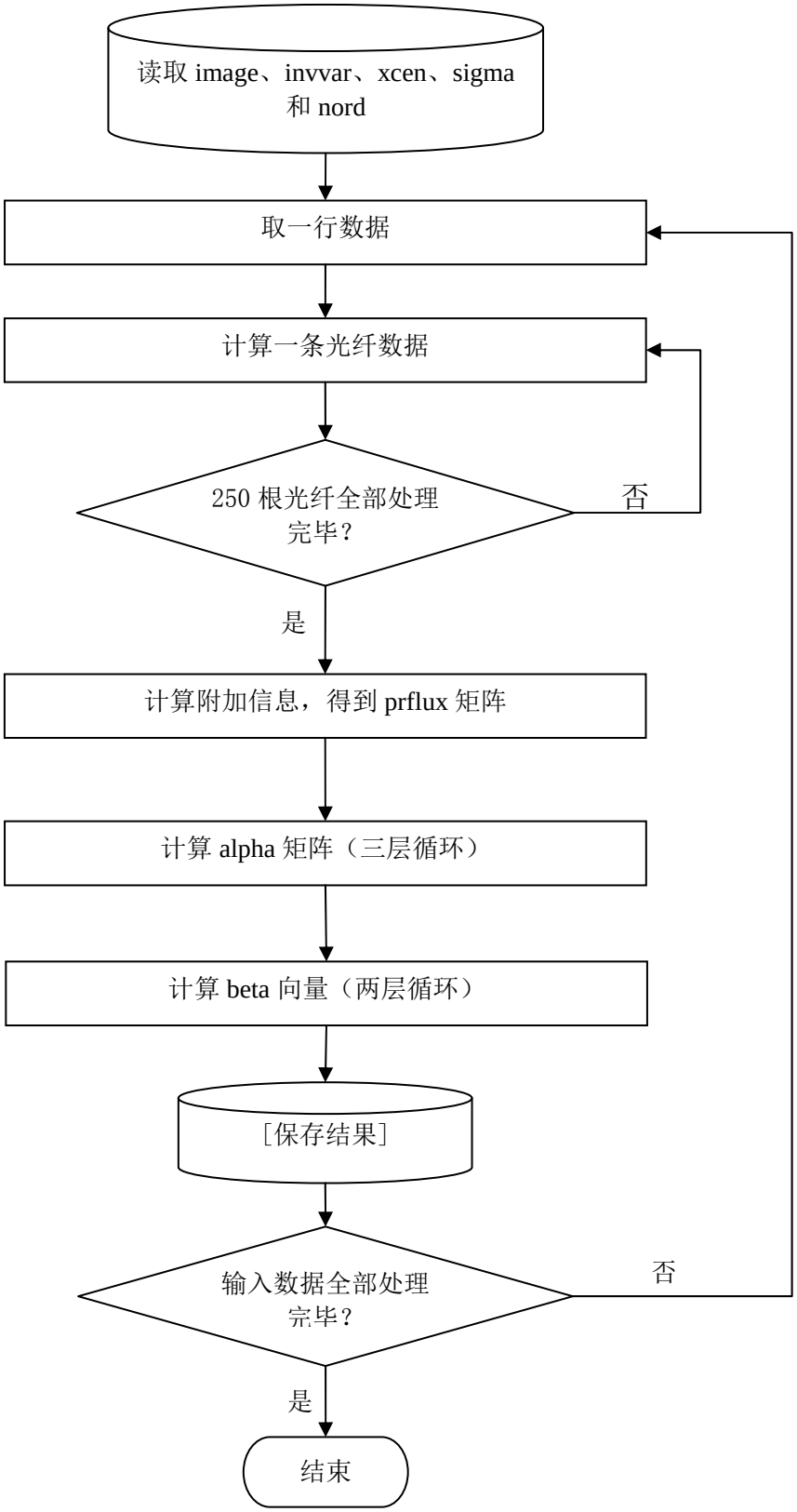


图 3-5 串行抽谱算法流程图

### 3.2.2 CUDA 并行方法

通过分析发现，输入数据的行与行之间并无关联，因而可以并行处理；另外，处理一行输入数据时，对 250 根光纤的计算也相互独立，因此可以并行。综合考虑后，决定采取如下并行方法——

一，每次将多行数据传入显存，由 GPU 并行处理。

由于每行输入数据都将生成一个 prflux 矩阵、一个 alpha 矩阵和一个 beta 向量，数据类型为双精度浮点数，可算出体积为 （ 4096*254 + 254*254 + 254 ） * 8 Byte = 8.3 MB 。对于拥有 896MB 显存的 GTX260 显卡来说，大约可容纳 100 组，因此，每次传入显存的原始数据不应超过 100 行。

二，将整个处理流程分成 4 部分，对应 4 个 kernel 函数：cuda_calculate_prflux_part1（计算 prflux 矩阵第一部分）、cuda_calculate_prflux_part2（计算 prflux 矩阵第二部分）、cuda_calculate_alpha（计算 alpha 矩阵）和 cuda_calculate_beta（计算 beta 向量），从而实现 CUDA 并行计算。

这 4 个 kernel 函数的并行粒度不尽相同。cuda_calculate_prflux_part1 和 cuda_calculate_alpha 流程比较复杂，计算量大，因此将任务进一步细分，每次发射，整个 grid 只负责一行原始数据的处理工作，每个 block 负责其中一部分。通过多次发射，完成多行数据的处理。

而 cuda_calculate_prflux_part2 和 cuda_calculate_beta 计算量较小，可由每个 block 负责一行输入数据，一次发射便可完成多行输入数据的处理任务。

CUDA 并行算法流程如图 3-6 所示。

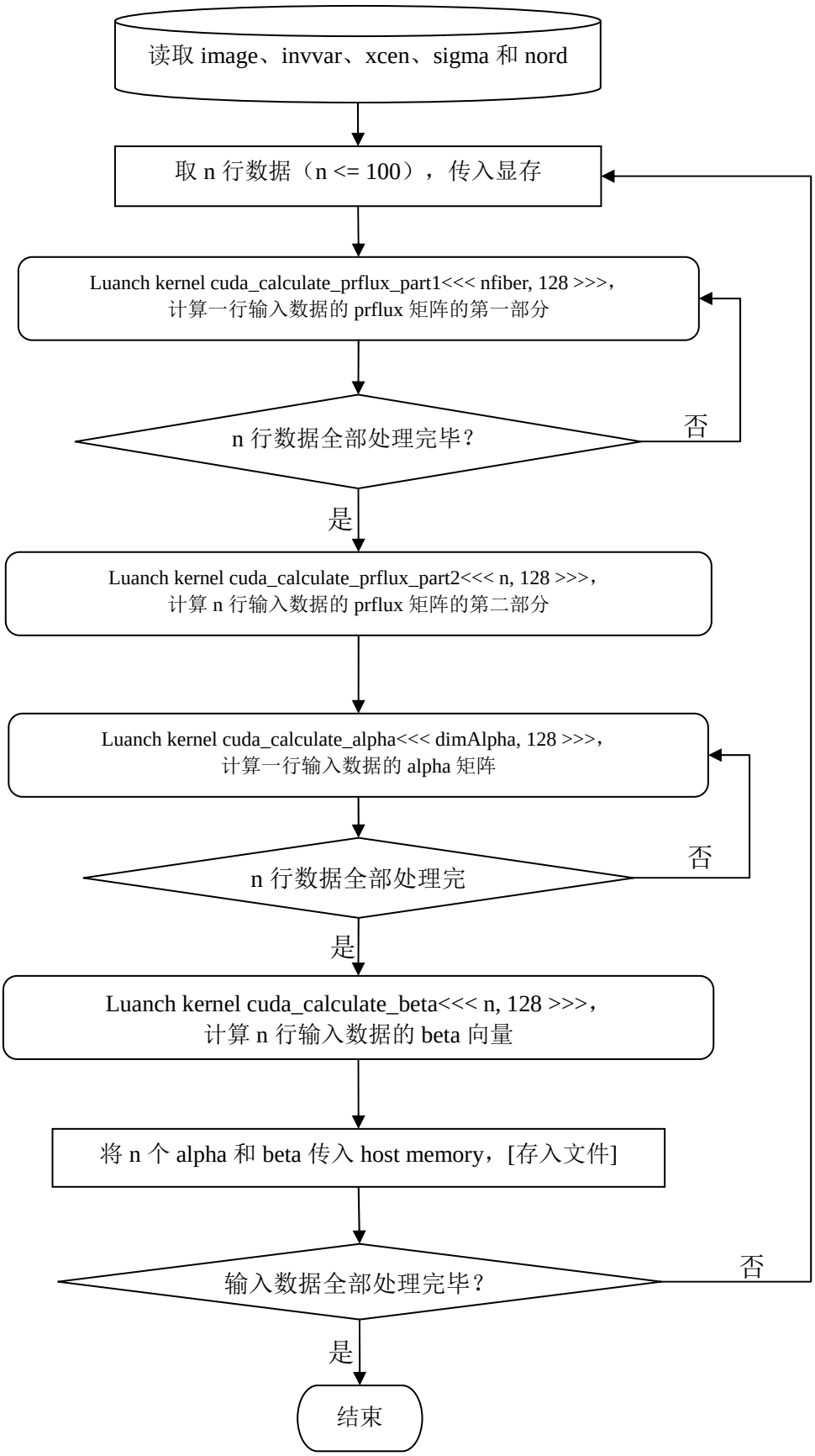


图 3-6 CUDA 并行抽谱算法流程图

### 3.2.3 并行算法优化策略

由于 CUDA 计算平台独特的硬件架构，代码需要进行必要的优化，方能获得最好的加速效果。结合实际情况，我们采取了以下优化策略：

一，充分利用共享存储器。

共享存储器集成在 SM 内部，是除寄存器之外访问速度最快的存储器，其地位类似于 CPU 的缓存。因此，充分利用共享存储器可大大减少访存时间，有效提升程序性能。在我们的程序中，中间结果都尽量存放在共享存储器中。例如，在 cuda_calculate_prflux_part1 中使用了 5 个共享存储器变量——

```
__shared__ double pfiber[102];
__shared__ double ff[4];
__shared__ int startPix;
__shared__ int endPix;
__shared__ double totalPfiber。
```

这些变量被 block 中所有线程使用。这 5 个变量总体积为 864 字节，每个 SM 16KB 的共享存储器可供 18 个 block 使用，从而保证 SM 上有足够多的活动线程块。

二，对全局存储器（显存）合并访问。

访问全局存储器，延迟要高得多，可能比寄存器或共享存储器慢上百倍。然而有些数据在共享存储器中放不下，只能放到全局存储器中，此时做到合并访问就显得至关重要。在我们的程序中，中间结果 prflux 矩阵存放在全局存储器中，被内核函数 cuda_calculate_alpha 和 cuda_calculate_beta 多次访问。为做到合并访问，我们将 prflux 矩阵按列（而不是按行）存储在一维线性存储空间中。但这样一来，cuda_calculate_prflux_part1 和 cuda_calculate_prflux_part2 将不能合并访问 prflux。然而由于后两者访问次数比前两者要少得多，这样的存储方式带来的益处远大于造成的损失。

此外，我们还注意控制内核函数中声明变量的数量，以避免寄存器不够用导致变量被放入局部存储器中。还有一些优化措施，如使用纹理存储器提升访存性能、使用 cudaMallocHost 分配主机端存储器提高 PCI-E 传输速度、使用流式处理隐藏与主机的通信时间等等，由于时间关系我们未能全部进行调试、引入。即便如此，依旧获得了

可观的加速比，可见“充分利用共享存储器”与“实现全局存储器合并访问”是简单却极为有效的优化策略。

### 3.2.4 性能对比

测试平台使用一颗 AMD Phenom(tm) 8650 三核处理器，4GB DDR2-800 内存，一块 GTX260 显卡。软件环境为 Ubuntu 9.10 32-bit + CUDA 3.1。分别使用 C 语言串行程序和 CUDA 并行程序处理不同行数的原始数据，耗时情况如表 3-2 和图 3-7 所示。

表 3-2 程序运行耗时情况

| 输入数据量（行） | 10 | 20 | 40 | 80 | 160 | 320 | 640 |
|---|---|---|---|---|---|---|---|
| C 串行程序耗时（秒） | 4.12 | 8.24 | 16.43 | 33.07 | 65.52 | 132.36 | 266.27 |
| CUDA 并行程序耗时（秒） | 0.25 | 0.50 | 1.00 | 1.98 | 3.93 | 7.83 | 15.61 |
| 加速比 | 16.48 | 16.48 | 16.43 | 16.70 | 16.67 | 16.90 | 17.06 |

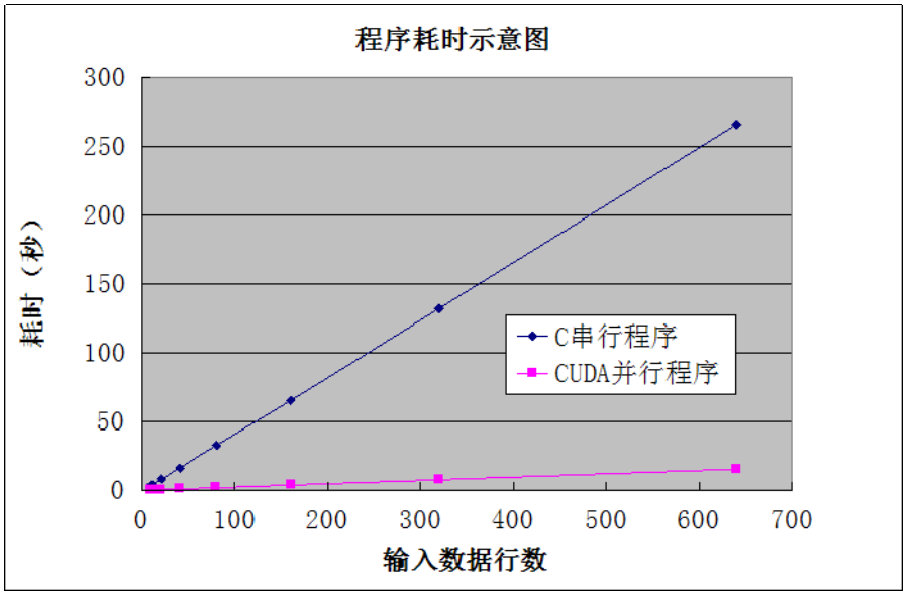


图 3-7 程序耗时示意图

可以看到，使用 CUDA 后，加速效果是显著的。而且随着处理数据量的增加，加速比保持稳定（略升），说明程序有较好的可扩展性。

值得一提的是，抽谱算法要求使用双精度浮点数，而 GTX260（GT200 核心）显

卡双精度性能只是单精度的 1/8，这阻碍了加速比的进一步提升。我们很期待新一代 Fermi 架构的 CUDA 计算卡，其双精度性能为单精度的 1/2，相信会在科学计算中发挥更大的威力。另一方面，新核心带有二级缓存，可以缓冲一些反复读取的数据，应能明显缓解全局存储器的访问压力，进一步提升性能。

对比 C 程序和 CUDA 程序的计算结果，发现有细微的差别。我们推测这是由于 CPU 和 GPU 在硬件上对浮点运算的实现有差异而导致的。差别非常微小，可以忽略其对最终结果的影响。而新的 Fermi 架构全面支持 IEEE 754-2008 标准，相信可以得到更加精确的结果。

## 3.3 本章小结

本章中我们尝试利用 CUDA 并行计算技术解决天文学研究中一些计算密集的问题：首先实现了基于 CUDA 的 kNN 天体分类算法，相比 CPU 串行程序获得了数十倍的加速比；而后利用 CUDA 技术加速郭守敬望远镜抽谱算法，在使用双精度浮点运算的情况下比 C 语言串行程序亦有约 17 倍的加速比。这些实例说明 CUDA 并行计算技术在解决某些计算密集的天文问题中是有用武之地的。

# 第四章 总结和展望

本文致力于高效、易用的海量星表融合工具的开发和将 CUDA 并行计算技术引入天文学研究。针对天文数据的海量性、分布性和异地异构性等特点，我们分析了现有交叉证认工具的优缺点，总结了前人的研究成果，结合实际需求，开发出一套高效的海量星表融合工具集。其中，自动入库工具可以协助天文工作者方便地存储、管理和处理数据；交叉证认工具使天文工作者能够快速获得天体的多波段信息，从而更加全面地了解天体的多波段特性，加深对天体的认识，促进新发现的产生。利用我们开发的 NED/SIMBAD 自动证认工具，可以很方便地进行大量天文数据与 NED/SIMBAD 数据库的交叉证认，从而将天文工作者从枯燥、繁琐的重复性工作中解脱出来。另外，我们也探索了 CUDA 技术在天文计算中的应用，尝试用它加速 k 近邻天体分类算法和郭守敬望远镜抽谱算法，最终获得了十几倍乃至几十倍的加速比，而且随着软硬件的发展还会有提升的空间。我们的试验结果表明，面对日益增长的海量天文数据，应用 CUDA 技术加速天文计算是一个可行的方案。

整个“海量星表融合工具集”中最重要的部分就是“基于 MySQL 数据库的海量星表交叉证认程序”。我们在前人工作的基础上，对原有程序加以改进和创新，优化程序代码，引入并行计算，还开发了图形用户界面，使程序在性能和易用性方面都有了大幅度的提升。为获得的成果感到喜悦的同时，我们也清醒地认识到，这些工作只是一个大课题中很小的一部分，还有很多值得探究的问题。例如，实验结果表明，CPU 的效能并未得到充分发挥，瓶颈存在于 I/O 方面，要想进一步提升程序整体性能，就要从提升数据库性能着手。改进数据库的方法有很多，其中最简单有效的就是“分表”。目前我们的存储方式是一个星表对应一个 MySQL 数据库表，像 2MASS、SDSS 以及 USNO-B 这些上亿行甚至上十亿行的星表，对应的数据库表是非常大的（数十 GB 乃至数百 GB），而表体积过大会严重影响 MySQL 查询性能。如果能将单一星表拆分为多个 MySQL 表进行存储，查询性能会大大提升。这就涉及到如何拆分已及应用程序如何配合等问题，需要进行深入调研、多方交流方能有效实施。另外，还可以尝试使用 MySQL 集群等分布式存储技术，配合高性能网络传输设备，以获得更好的 I/O 性能。将来，随着星表规模的进一步增大，传统的关系型数据库很可能会不再适用，而发源于海量分布式数据存储、管理和应用的 NoSQL 技术则日益凸显出优势，很可能会替代关系型数据库成为新的海量天文数据存储解决方案，届时交叉证认程序又会有

新的面貌。

除了 I/O 问题，计算性能也需要关注，可以尝试一些高性能计算技术——如集群分布式并行计算、基于 CUDA、OpenCL 的显卡通用计算等——来实现更加快速的计算。

证认算法也有很大的改进空间。目前我们只是简单地根据坐标来进行判断，距离足够近的，就认为是对应体——虽然对于超大样本、统计性的研究，这样做是可以接受的，但随着研究的深入，势必需要更精准、更智能的判断，这时候就有必要将多个参数（如星等、流量等）一并纳入考量，对多个足够近的对应体做出有效的甄别，从而发展出更复杂的算法。

此外，星表数据分割导致的漏源问题是一个比较棘手的问题。如果不对之进行处理，尽管漏掉的数量极少，却有可能漏掉意义重大的发现；如果进行处理，则要付出相当大的性能代价。一个折衷的方案是：采用两种不同的分割方式（如 HEALPix 和 HTM）分别证认。分割方式不同，分块边缘的形状和位置也不同，所漏的源也不同。二者的结果可以相互检验、补充。

在基于因特网的交叉证认方面，我们开发了 NED/SIMBAD 自动证认工具，大大提高了天文工作者的工作效率。不过，这两个程序目前仅支持命令行模式，通用性也不够好，这都是将来需要改进的地方。

上述工具都是用 Python 语言编写的，开发和运行环境都是 Linux，目前尚不能运行于 Windows 环境。不过由于 Python 是一种跨平台的语言，将程序迁移到 Windows 或其他平台也并非难事。

CUDA 并行计算方面，我们尝试把这一新技术应用于天文研究的某些领域，获得了一些经验。应该说，CUDA 平台的计算能力是非常强悍的，尤其适合用来解决密集的浮点运算问题。在同等功耗、同等资金投入的前提下，使用 CUDA 计算技术常常可以使程序获得数十倍乃至上百倍的加速比。当然，世上没有万能药，CUDA 技术也并非全是优点，也有自身的局限性。首要的问题就是编程的难度比较大，需要投入较多精力。其次，并非所有的问题都有并行解决方案；有些可以在 CPU 上并行的应用，不一定能在 CUDA 平台上实现并行。最后，由于 CUDA 设备源自图形处理器（GPU），而图形浮点运算一般不需要双精度，致使 CUDA 平台在双精度计算方面先天不足。不过 NVIDIA 公司一直在努力弥补这方面的缺陷，新的 Fermi 架构的双精度计算能力有了长足进步。总之，没有最好的技术，只有最合适的技术。现实应用中，我们需要结

合实际问题进行全面分析，做出明智的判断，灵活运用各种技术，不断促进天文学研究的进展。

# 参考文献


[1] 高丹，路勇，张彦霞，等. 海量星表融合系统(XMaS_VO)的设计与开发. 天文研究与技术, 2008(2):137~144

[2] 高丹. 海量天文数据融合系统的开发与数据挖掘算法的研究. 博士学位论文. 北京: 中国科学院国家天文台, 2008

[3] Zhao Q, Sun J Z, Yu C, et al. A paralleled large-scale astronomical cross-matching function. In: Hua A and Chang S-L. ICA3PP 2009. Taipei: Springer-Verlag Berlin Heidelberg, 2009. 604~614

[4] Gao, D., Zhang, Y., and Zhao, Y.. Support vector machines and kd-tree for separating quasars from large survey data bases, Mon. Not. R. Astron. Soc., 2008(386):1417~1425

[5] L, L., Zhang, Y., and Zhao, Y.. k-Nearest Neighbors for automated classification of celestial objects. Science in China G: Physics and Astronomy, 2008(51):916~922

[6] Ball, N. M. and Brunner, R. J.. Data Mining and Machine Learning in Astronomy. arXiv:0906.2173 (June 2009)

[7] Garcia, V., Debreuve, E., and Barlaud, M.. Fast k Nearest Neighbor Search using GPU. arXiv:0804.1448(Apr. 2008)

[8] Sainio, J.. CUDAEASY - a GPU accelerated cosmological lattice program. Computer Physics Communications, 2010(181):906~912

[9] Karimi, K., Dickson, N. G., and Hamze, F.. High-Performance Physics Simulations Using Multi-Core CPUs and GPGPUs in a Volunteer Computing Context. arXiv:1004.0023 (Mar. 2010)

[10] Sawerwain, M. and Gielerak, R.. GPGPU based simulations for one and two dimensional quantum walks. arXiv:1003.3779 (Mar. 2010)

[11] 金文敬. 天体测量星表与巡天观测的进展. 天文学进展, 2009(3):247~269

[12] 张舒，褚艳利，赵开勇，张钰勃. GPU 高性能计算之 CUDA. 第一版. 北京：中国水利水电出版社, 2009. 1~180

[13] US National Virtual Observatory, http://www.us-vo.org/

[14] http://www.astrogrid.org/

[15] VizieR Service, http://vizier.u-strasbg.fr/viz-bin/VizieR

[16] The Aladin Sky Atlas, http://aladin.u-strasbg.fr/aladin.gml

[17] SIMBAD Astronomical Database, http://simbad.u-strasbg.fr/simbad/

[18] NASA_IPAC Extragalactic Database, http://nedwww.ipac.caltech.edu/

[19] TOPCAT, http://www.star.bris.ac.uk/~mbt/topcat/

[20] http://www.nvidia.com/page/home.html

[21] http://www.nvidia.com/object/cuda_home_new.html

[22] Hierarchical Triangular Mesh, http://www.skyserver.org/HTM/

[23] Hierarchical Equal Area isoLatitude Pixelization, http://healpix.jpl.nasa.gov/

[24] Python, http://www.python.org

[25] NVSS, http://www.cv.nrao.edu/nvss/

[26] FIRST, http://sundog.stsci.edu/index.html

# 发表文章目录

1. 裴彤, 张彦霞, 彭南博, 赵永恒. Python 多核并行计算在海量星表交叉证认中的应用. 中国科学: 物理学 力学 天文学, 2011, 41: 102 ~ 107, doi: 10.1360/132009-1163

2. Tong Pei, Yanxia Zhang, and Yongheng Zhao. A High Efficient and Fast kNN Algorithm Based on CUDA. Proc. of SPIE, 2010, Vol. 7740, 77402G, doi: 10.1117/12.856768

3. Nanbo Peng, Yanxia Zhang, Tong Pei and Yongheng Zhao. A Simple and Effective Algorithm for Quasar Candidate Selection. Proc. of SPIE, 2010, Vol. 7740, 77402X, doi: 10.1117/12.856766

# 致谢

转眼间，在国家天文台的硕士研究生生涯就要结束了。这几年经历了很多，也学到了很多，回首往事，感恩之情常萦绕心头。

由衷地感谢我的导师张彦霞副研究员。张老师学术基础扎实，治学勤奋，踏实严谨，对天文科学有着深刻的领悟，视野广博，常常提出超前的想法，令人敬佩。在教学方面，我作为学生，深切感受到张老师的对学生培养工作的尽职尽责。大到科研方向，小到论文中某处遣词造句，张老师都尽心尽力地给予指导，我所取得的每一点成绩，都离不开张老师的心血。生活中，张老师为人热情、坦诚，总是设身处地地为学生着想，感谢张老师在各个方面给予我的数不清的照顾和理解！

真诚地感谢赵永恒研究员。赵老师有着“平淡是真”的大科学家风范，即使在我这样最初级的学生面前也毫无架子，认真地倾听我们幼稚的想法。近年来赵老师为 LAMOST 项目倾注了全部心血，总是十分繁忙。尽管对我指导的次数不多，但他敏锐的思维、高屋建瓴的远见卓识仍令我受益匪浅。看着赵老师越来越多的白发，不由得想道声：“赵老师，您辛苦了！”

真诚地感谢罗阿理研究员。罗老师学术上一丝不苟，生活上和蔼可亲，像一个父辈，又像一个兄长。作为支部书记，罗老师出色地组织了党员工作和活动。感谢罗老师在各方面给我的帮助！

特别要感谢杜红荣和艾华两位老师。两位老师对工作的认真负责、对学生的关怀帮助大家都有目共睹，于我而言，感受尤其深刻：有段时间身体状况很差，不得不休学，病痛的折磨再加上牵扯到各方面繁琐的手续和流程，心中很是彷徨无助。这时杜老师和艾老师一面暖言相慰，一面不厌其烦地帮我办理各种手续，使我没有后顾之忧，安心休养，终于顺利返校，重拾学业。感激的话语已无需多说，只在心里默默祝福：好人一生平安！

感谢 LAMOST 项目组全体老师：袁晖老师、陈英老师、王丹老师、张昊彤老师、白仲瑞老师、陈建军老师、孙士卫老师……感谢你们对我的关心和帮助。感谢白仲瑞和陈建军老师提供抽谱源程序。感谢科技处崔辰州老师帮忙提供试验数据。感谢国家天文台赵永恒研究员、吴宏研究员和北师大天文系姜碧沩教授对我的论文给予评阅，以及国家天文台马骏研究员、罗阿理研究员、北京交通大学赵瑞珍教授在百忙之中抽出时间参加论文答辩。谢谢你们！

感谢 LAMOST 项目组的所有同学：彭南博、邬科飞、杜薇、宋逸晗、杨三军、吴玉中、韩金姝、李荫碧、李君、左芳、韦鹏、潘元月、任娟娟、河林、姜斌、樊东卫、李明……和你们相处的日子是我最美好的回忆。感谢天津大学赵青同学对我的帮助。谢谢你们！

感谢球友白仲瑞、陈建军、张伟、王凤飞、FAST 组管皓、CAPS 组胡超、探月组夏金超等等，跟你们打球很快乐！

感谢我的父母、兄弟、姐妹，感谢我的妻子，你们是我生活的意义。

——感谢所有曾给予我指导、帮助和鼓励的人们，谢谢你们！

最后要感谢 WPS 的创造者——金山软件公司。这篇硕士毕业论文就是用“永久免费”的 WPS Office 2010 个人版完成的，而多年前做本科毕业论文时也是用的 WPS。WPS 二十年的坚持令人赞叹，WPS 人不屈的精神给我鼓舞，感谢你们！

# LAMOST 致星光

那，那一点光芒
你来到了吗...

我知道 我知道
这一路太艰辛
母星的荆棘 割伤你躯体
尘埃埋伏 耗散你力量
你躲过黑洞猎杀
忍过无尽空旷
时空将路扭曲
弯弯转转
脚步踉跄

我知道 我知道
这时间太漫长
你启程时
这大陆或还战马嘶鸣，羽箭飞舞
又或是一片蛮野
古猿投向星空无意的目光
不对不对
那还是第一缕生命的闪电？
更早，更早
这星球尚未冷却，火热杀场

太阳刚刚点燃
你已在路上

这一路太漫长
这一路太漫长
终于今日，你到了
我也到了
将你收入眼
透过最后的纱帐

宇宙就像计算机，基本规律造就大千世界，问题在于，宇宙的 0、1 是什么？
计算机就像宇宙，简单字符搭起复杂系统，问题在于，多少字符才足够创出一个灵魂？
而人，宇宙计算机中一个小小的进程，有能力理解计算机本身吗？？