天津大学博士学位论文

面向海量数据的高效天文交叉证认的研究

Research on High-Efficient Massive Data Oriented Astronomical Cross-Match

学科专业：计算机应用技术
研 究 生：赵青
指导教师：孙济洲 教授

天津大学计算机科学与技术学院
二零一零年五月

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得 **天津大学** 或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：　　签字日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解 **天津大学** 有关保留、使用学位论文的规定。特授权 **天津大学** 可以将学位论文的全部或部分内容编入有关数据库进行检索，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

中文摘要

天文交叉证认是实现多波段数据融合的关键技术。经过交叉证认形成的多波段或全波段数据蕴含了更多的可揭示天体物理本质的信息，是加深对天体的认识、促进天文学新发现的关键。天文数据的海量性使其必须要依靠并行计算、分布式计算和海量数据处理等计算机技术加以解决。本文在前人研究的基础上，针对多核处理器环境、大规模集群环境分别研究并实现了高效的并行交叉证认方法和分布式交叉证认方法，并在攻克它的主要性能瓶颈——过于频繁、耗时的数据 I/O 操作方面取得了一定的突破，真正实现了海量数据上的大规模交叉证认。

本文首先研究设计了面向多核环境的并行交叉证认方法，应用 HEALPix 伪二维球面索引方法在加快数据查询速度的同时实现了数据的区域划分，降低了证认计算的时间复杂度。然后研究并解决了交叉证认的常见问题——边界漏源问题，保证了结果的完全性。实验表明，此方法对交叉证认计算的效率提升明显。此后，针对该方法的最主要耗时环节——数据库查询操作进行了优化，提出了基于限制生长模型的全新数据加载、计算流程，以及任务分配调度的基本单元——最大生长块，既降低了数据重复读取率，又实现了对稀疏数据集中空白区域的过滤，使交叉证认的效率得以继续提高。除此之外，通过理论分析结合实验测试的方式对此并行交叉证认方法在 HTM 索引下的可行性进行了验证，从而打破了对单一索引方式的依赖，保证了此方法的广泛适应性。

为了突破关系数据库在处理海量数据时的性能制约，同时也为了满足海量数据的存储需求，本文继续提出了基于 MapReduce 分布式并行计算模型及分布式文件系统的交叉证认方法。算法设计上，依照 MapReduce 模型的特点，通过规划数据在各节点间的分布，尽量地避免了交叉证认计算过程中的节点间通信，保证了接近线性的加速比。实验显示，在大规模数据集上此方法的性能远远优于多核环境下基于关系数据库的并行交叉证认算法，为今后在线实时交叉证认服务的实现打下了基础。

此外，本文提出的基于位运算的快速邻域编码计算算法不仅是高效交叉证认得以实现的一个基础性保证，也对诸如锥形检索等多种天文数据处理应用有着重要的作用。本文应用并行计算技术、分布式计算技术、以及海量数据处理技术研究设计的高效交叉证认算法对今后解决同类大规模天文数据处理应用的效率问题有着重要的参考价值。

关键词：天文交叉证认 限制生长模型 MapReduce HEALPix HTM

ABSTRACT

Astronomical cross-match is the kernel technology to realize multi-band data aggregation. After this operation, the multi-band or full-band data contains more information to reveal celestial objects' physical essence. Therefore it is a key step to deenpen astronomers' understanding of celestial objects and accelerate new scientific discoveries. Because the astronomical data sets are usually very large, this problem must be resolved by computer technologies such as parallel computing, distributed computing and massive data processing technologies. In this thesis, based on previous research, both efficient parallel cross-match function and distributed cross-match function have been designed and implemented separately for multi-core environment and large-scale cluster enironment, and some breakthroughs have been made for its main performance bottleneck – too frequent and time-consuming data I/O operations. As a result, large-scale cross-match computing on massive data sets becomes reality.

Firstly, an parallel cross-match function in multi-core environment has been designed. By adopting HEALPix, which is a pseudo two dimensional spherical index method, not only the speed of data querying has been increased, but also the time complexity of cross-match computing has been reduced by implementing data regional partition. And then for the common and classic problem in cross-match – the border source-leakage problem, a solution has been issued which can guarantee the results' integrality. Experiments show that this method has a great contribution to the efficiency improvement of cross-match. After that, a new data loading and computing flow model named boundary growing model, and a basic task assignment and scheduling unit named biggest growing block are proposed. They not only reduce the data re-reading frequency, but also implement the data filtration of space areas. The experiment results show that it can furtherly improve the efficiency of cross-match by about 50 percent. In addition, the thesis also validate the feasibility of these cross-match methods under the HTM index function through both theoretical and experimental analysis. Therefore, it can be believed that these methods have broken the dependency on unitary index function comparing with traditional functions.

In order to break through the performance limitation of relational database when processing magnanimity data, as well as to satisfy the storage requirements for the massive astronomical observation data, the paper furtherly presents a new cross-match

function based on MapReduce distributed computing model and its corresponding distributed file system. According to the distingrish feathers of MapReduce, by re-arranging the data distribution among computing nodes, the inter-node communication has been reduced as much as possible, as a result, a near-linear speedup has been achieved. The experimental results show that this method outperforms the above-metioned parallel cross-math methods based on relational database in multi-core platform greatly. It makes a foundation for the implementation of real-time online cross-match service in future.

On the other hand, the quick bit-operational algorithms issued in this thesis which are used to calculate the index numbers of the neighbor blocks are not only a basic guarantee for realizing high-efficiency cross-match, but also play an important role on multiple kinds of astronomical data processing applications such as cone search. The efficient cross-match approaches issued in this thesis using parallel computing technologies, distributed computing technologies, and massive data processing technologies, have high reference values for resolving other large-scale astronomical data processing problems in future.

KEY WORDS： Astronomical Cross-Match, Boundary Growing Model, MapReduce, HTM, HEALPix

目　　录

第一章 绪论

随着科学技术的发展，天文学已经步入了数据爆炸、信息丰富的全波段时代，从射电、红外、可见光、紫外、X 射线、伽玛射线都有观测项目在进行。对不同波段的天文观测数据进行数据融合以得到蕴含更多信息的多波段或全波段数据，是天文学研究的基础，尤其对多波段数据挖掘和统计分析工作具有重要意义。交叉证认计算正是这种多波段数据融合的核心技术，是加深对天体的认识、促进天文学新发现的关键一步。

近年来望远镜技术的飞速发展使各个研究机构采集的数据量呈类摩尔定律增长，交叉证认处理效率的高低直接关系到天文学家对这些数据的利用效率的高低，关系到这些高端天文观测设备的技术投入能否快速转换为天文研究成果，因此，实现面向海量数据的高效交叉证认迫在眉睫。

本课题的工作是中国虚拟天文台（China-VO）建设的核心内容之一。国家天文台信息与计算中心已经归档了国内外的多个数据源，这是 China-VO 的主要数据来源，此外还将不断有新的观测数据归档。面对快速增长的数据规模，本文利用大规模数据索引技术、并行计算技术提出了两种计算环境下的高效交叉证认方法：面向多核环境的并行交叉证认和面向 MapReduce 集群环境的分布式交叉证认，使海量数据规模上的多波段交叉证认在效率上真正地具备了实际应用价值。

1.1 课题研究背景及意义

1.1.1 天文数据处理与计算机信息技术

天文学是一门古老而又充满活力的学科，从人类文明史揭开第一页开始，天文学就有着重要的地位。几乎所有的自然科学学科研究的都是地球上的现象，只有天文学从它诞生之日起就与我们头顶上那相隔万里、可望而不可及的璀璨星空联系在一起，它蕴含的是人类对未知世界最原始最永恒的好奇心和探索精神。早在原始社会天文学就开始萌芽了，由于生产和生活的需要，人们开始观察记录天文现象，经过世代连续不断的努力，积累了越来越多的天文学知识，推动了人类社会的进步和科技的发展，在人类文明发展史上写下了辉煌的篇章。

科学技术高度发展的今天，天文学更是集中了物理学、数学、化学、生命科

学、信息科学等各学科最为前沿的技术、理论，成为了最富生命力和创造力的学科，它更直接关系着航天、气象等高尖端领域的发展，关系着人类的未来，其研究水平标志着一个国家与民族在科技发展前沿中的位置。

进入二十世纪中期之后，随着探测器和空间技术的发展以及天文研究工作的深入，天文观测扩展到了包括可见光、射电、红外、紫外、X 射线和 γ 射线在内的电磁波各个波段，形成了多波段天文学。伴随着这些空前壮大的多波段数据的不断获得，人类对探索各类天体和天文现象的物理本质的认识迈入了全新的阶段。陶醉于一个个最新研究成果的不断涌现，各国更是加快了对高精端探测设备的研究和建设。这是因为，与其它学科不同，天文的新发现与新成果的获得必须依赖于对宇宙的观察、对数据的采集。正如国家天文台台长严俊所指出“天文学是一门观测科学，天文望远镜‘高瞻远瞩’的能力代表着一个国家的天文学发展水平”。当今世界各国都加大了对先进观测仪器的研究和投入，仅 2008 年一年，仅我国就有 3 座大型天文望远镜建成或奠基，分别是：每夜光谱观测量上万的国际上最具威力的光谱巡天望远镜 LAMOST；正在建造的口径最大、最具威力的单天线射电望远镜 FAST；可在厘米、分米波段上同时实现以高空间、高时间和高频率分辨率观测太阳动力学过程的射电频谱日象仪。然而，在各国天文观测设备日新月异、空前繁荣的今天，是否意味着天文学研究效率的直接提升，是否意味着各种新天文成果也会自然而然地接踵而来？或者说从设备的启用到成果的孵化是否是一个简单而平坦的过程？答案是否定的，这其中的困难首先来源于如何对这些高新设备所收集的呈级数增长的海量观察数据的有效利用。明知数据中包含着极具科学价值的信息，然而面对着动辄上 T、上 P 量级的数据规模，如何对数据进行存储、转换、整合从而可以保证在短时间内高效地提取出这些信息不仅关系到对高尖端天文设备研究和建设投入的回报率，而且关系到整个天文学科发展的进程。在这方面，当今各国的研究都已经证明，要高效地处理日益增长的海量天文数据必须要依靠计算机信息科学。

分析当今天文数据的几大特点[1]就可以得出结论：天文数据处理与信息科学的结合是一种必然，而且这种趋势从天文学告别照相底板时代而走向 CCD 时代就已经开始，且将无可逆转地一直走下去。具体而言，天文数据的特点一般可以概括为以下几点：

1) 数据海量性、增长爆炸性

正如前文所述，当今的天文数据量已经达到 T 量级甚至 P 量级，而且在可预计的未来几十年里，天文数据的这种增长趋势还将长期继续下去。事实上，在空间技术、望远镜设计、制造技术的多重推动下，过去二十年间天文学的数据收集能力的增长情况已经显现出了一种超摩尔定律式的增长规律，图 1-1 为近年来

欧洲南方天文台科学数据中心所拥有的数据的增长情况[2]，已经很好的映证了这一规律。

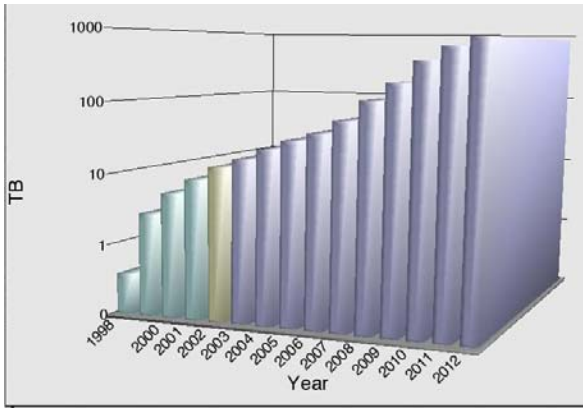


图 1-1 欧洲南方天文台科学数据中心所拥有数据增长情况

从图中可以看出，欧洲南方天文台的数据量的增长速度已经超过了摩尔定律下的速率，而且这并非只是一个特例，其他各家大型天文台也有着类似的情况报道[3] [4]。伴随着各国新一代天文观测设备的继续建立，这种增长趋势只会越来越明显。比如，我国刚刚建立落成的直径 4 米的 LAMOST 望远镜每天的观测量可达 3GB，美国哈勃太空望远镜每天的数据量约为 5GB，而美国另一个正在建立的 8.4 米口径大尺度概要巡天望远镜（LSST）[5]每晚产生的数据量更将达到惊人的 18TB，相关文献预测[6]，在未来 10 年间，LSST 产生的数据总量大概为：15PB 的数据库数据和 55PB 的文件数据。

2) 全球范围高度分散性

各国天文组织分别选择最优越的地理位置建立起了一批先进的观察仪器。经过多年艰苦的观测和积累产生了许多重要的数据库，如哈勃空间望远镜 HST，斯隆数字巡天 SDSS、两微米全天巡天 2MASS[7]、钱德拉 X 射线天文数据和 Palomar 帕洛玛天文台数字化巡天 DPOSS 等。这些数据地理位置上遍布全球各个角落，因为数据量巨大，不可能统一同地存储，而很多科学研究都需要综合利用各个数据集联合后的数据，才能完成多波段、多时序的深入分析，所以基于互联网、局域网多层次网络结构的分布式计算技术对天文数据的处理意义重大[8]。

3) 数据格式多样性

天文数据形式多种多样，按照内容分类主要有星表、星图、光谱等，而格式上更是纷乱复杂，各个天文数据库都有自己的格式标准。所以实现这些数据网络平台上的统一访问本身就要依靠信息技术。

4) 开放性和非盈利性

天文学科比所有其他学科都更具有共享精神，它的非盈利性特征使各研究组

织之间可以不加保密地相互共享数据、交流研究成果。按照天文界的传统，一个观测项目的数据将在一两年后向世界开放，这既维护了国际上大多数天文观测项目的优先享用的权益，也有利于整个学科的发展。随着互联网时代的到来，很多天文观测项目都已经或正在计划将他们的观察数据通过互联网向全世界天文学家进行发布。随着像 LSST 等一批超高采集能力的望远镜的诞生，海量的天文数据将向高性能计算技术、网络技术等计算机信息技术提出全面的挑战，与此同时，天文数据所具有的海量性和开放性又使它成为了计算机信息技术继续发展的最好的试验场。在这方面并非没有先例，美国微软公司开放的 SQL Server 数据库管理系统就是在与美国天文界的合作中得益于天文数据的海量性完成了性能提升和突破的。

由以上归纳的天文数据特点可以看出，信息技术与天文处理存在着相辅相成、互相促进的关系，而本文研究的高效多波段交叉证认方法正是天文数据处理中的一个基础性课题，它对更加复杂、更加深入的天文数据处理有着重要的参考意义。除此之外，天文多波段交叉证认还是虚拟天文台项目中多波段联合查询的核心技术，而虚拟天文台项目本身正是当今天文与信息技术相互结合的最直接、最集中的体现。

1.1.2 虚拟天文台

从古至今，人类对浩瀚宇宙的探索始终不曾停止。步入当代，望远镜制造技术日新月异，主动光学技术、薄镜面拼接技术、自适应光学技术等崭新技术的引入，使可观测的范围越来越广，效率越来越高，巨大的数据产出使天文学开始步入了数据富饶时代，一时间，人们开始发现，以往那个辛辛苦苦观测许久但数据还是不够用的年代一去不复返，取而代之，人们开始为如何处理这些庞大、形式多样的数据集而发愁。因此，像许多其它学科一样，寄希望于信息技术发展所带来的强大处理能力和快速便捷的使用方式，是天文学继续保持高速发展的必然出路。正是在这样的背景下，虚拟天文台诞生了，并且立刻在全球范围内显示出了它强大的生命力，科学家们开始预言，它将成为开创“天文学发现新时代”的关键因素[9]。

数据是天文学赖以发展的基础，从这一角度，虚拟天文台的含义可以归纳为三个层面[10]：数据归档层、数据查询层和信息探索层。从数据归档层面上来看，它实现了全球各个天文中心 γ 射线巡天、X 射线巡天、紫外巡天、光学巡天、红外巡天和射电巡天所得的观测数据的统一归档，形象地来说，相当于构成了一个全波段的数字虚拟天空。从数据查询层面上看，它为用户提供了面向各类数据的精准的查询服务，其作用相当于一架虚拟的超级天文望远镜；从信息发现层来看，

它着眼的是功能强大的计算工具、统计分析工具和数据挖掘工具的开发，其作用就好似全功效的探测设备。因此可以说，虚拟天文台就是虚拟数字天空、虚拟望远镜和虚拟探测设备的总和[31]，借助于它的强大功能，天文学家将迎来全新的E-Science 时代的工作模式。

虚拟天文台的诞生，使数量高达 TB 甚至 PB 量级、波长遍及从伽马射线到射电波段的天体图像库，功能强大的数据挖掘和分析工具，大量 PB 量级容量的存储设备以及每秒百万亿次、千万亿次的超级计算设备通过高速互联网络无缝地融合为一体，成为一个数据密集型的在线科学研究平台；使世界各地的天文学家可以直接获得当前各国最为先进的探测设备采集的多波段数据，并利用其提供的各种在线工具完成更为复杂的数据处理、结果分析等一整套的科学研究过程，大大地增加了复杂规律和稀有天体的发现机会；使各国的知识、技术共享更为便利，促进各国天文界的相互协作，成为全世界共有的交流平台。

虚拟天文台一经提出就得到了全世界各国的广泛响应和积极投入，被公认为未来天文学研究的一个重要发展方向。1999 年，美国国家科学院天文学及天体物理学发展规划委员会召集全国最优秀的天文学家，发表了题为“新千年的天文学和天体物理学”的未来十年发展规划，其中把建立国家虚拟天文台（NVO）作为最优先推荐的项目[11]。2002 年 6 月在德国召开了一个名为“走向国际虚拟天文台”的国际会议，从此国际虚拟天文台联盟（IVOA）[12]正式成立。2002 年 10 月，中国虚拟天文台（China-VO）成为了国际虚拟天文台联盟的正式成员，它将作为国际虚拟天文台不可或缺的一部分，完成它的中国部分，引领中国天文学进入数据密集型在线科学研究新时代。

1.1.3 课题研究意义

当今天文学已进入全波段天文学阶段，射电技术、红外探测技术、航天和空间技术的兴起，使当今的天文观测早已不局限于光学波段，而是包括了射电、红外、可见光、紫外、X 射线、γ 射线的各个波段，极大地提高了人类的认知能力。然而，多波段或全波段数据，并非与生俱来，各个望远镜在设计上都只具备对一段特定波段的数据的观测能力，多波段或全波段的数据是在对多个星表进行数据统一整合加工之后的产物，然而由于各种望远镜存在着广泛而各不相等的误差，这种整合并不是一个容易的过程，它需要一个名为“交叉证认”的步骤。

通常认为，在星表间精确地确认天体间的对应关系需要考虑各个波段上的多重属性，而在方法上也要视证认星表对象的不同而灵活的应用数据挖掘、机器学习等方法进行复杂的判断和分析。然而这种复杂的交叉证认方法虽然较为精确但效率低下，通用性差，仅适合于对特定研究目标下的小数据集进行精确分析比对。

考虑到天文数据的海量性，首先实现基于位置信息的交叉证认是实现大规模批量数据处理的唯一方法，它为天文学家提供了一个粗略但快速的参考答案，在一定程度上可以满足一些科学研究的需要。而当对证认精度有较高要求时，这种经过初步位置证认所得的结果可以帮助天文学家过滤掉绝大多数不可能的候选结果集，降低了后续精确分析中的搜索空间。这也是当今各国在构建虚拟天文台中的交叉证认服务、多源联合查询服务时所采用的解决方案。

基于位置信息的交叉证认原理：

1） 距离公式

以图 1-2 中的两点 A、B 为例，它们分别来源于星表 A 和星表 B，设其坐标分别为（α_1,δ_1）、（α_2,δ_2），它们之间的球面角距离 d 可以按如下步骤进行计算：

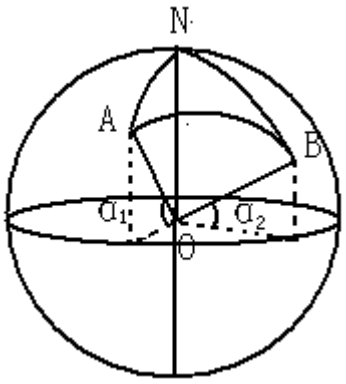


图 1-2 球面两点交叉证认原理

1° 若 $|\alpha_1-\alpha_2|\le 180°$， $\angle ANB=|\alpha_1-\alpha_2|$

2° 若 $|\alpha_1-\alpha_2|>180°$， $\angle ANB=360°-|\alpha_1-\alpha_2|$

根据球面余弦定理：

$$\angle AOB=\cos\angle AON\cos\angle BON+\sin\angle AON\sin\angle BON\cos\angle ANB$$
$$=\sin\delta_1\sin\delta_2+\cos\delta_1\cos\delta_2\cos(\alpha_1-\alpha_2)$$
$$d=\angle AOB=\arccos(\sin\delta_1\sin\delta_2+\cos\delta_1\cos\delta_2\cos(\alpha_1-\alpha_2))$$

当 A、B 两点之间角距离很小时，$\delta\approx(\delta_1+\delta_2)/2$

所以有 $d^2=((\alpha_1-\alpha_2)\cos\delta)^2+(\delta_1-\delta_2)^2$

2） 证认成功判断公式

$$d=\sqrt{((\alpha_1-\alpha_2)\cos\delta)^2+(\delta_1-\delta_2)^2}\le 3\sqrt{r_1^{\,2}+r_2^{\,2}} \qquad (1\text{-}1)$$

其中 r_1 和 r_2 为两个星表的误差半径，也就是当两点之间的距离满足上式时，可以认为两点证认成功，互为匹配的对应体。

基于位置信息的交叉证认方法，在数据库研究领域属于空间连接（Spatial

Join）[14][15]技术范畴，即根据空间位置信息实现对两个或两个以上数据集合的合并。它在空间数据库技术的研究中有着重要的意义。空间数据库是随着地理信息系统的诞生而出现的一种新型数据库技术，它通常用于储存在地球表面某一范围内与空间地理信息相关的、反应一定主题信息的数据集合，它突出的特点就是以空间目标作为存储对象，并根据空间信息的特殊性提供了一系列优化方法。除了地理方面，空间数据库在天文、城市规划等领域均有重要而广泛的应用，已成为当前数据库领域的一个热门研究课题。而本文所涉及的空间连接运算正是空间数据库研究中的一个重点和难点[16]，可以说它是其中最复杂、最耗时的一项基本操作，上世纪九十年代开始已有了一些应用并行计算技术解决这一问题的研究[17][18]，它的处理效率关系到空间数据库的整体性能，关系到空间数据库技术发展的前景。当前，空间数据库系统的理论和技术尚处于发展的初级阶段，常规关系数据库仍然是当今的主流，所以它通常只作为常规、传统数据库的扩展出现。考虑到关系数据库在当前天文数据处理中的主流地位，本文在研究多核环境下的交叉证认方法时，同样也采用了关系数据库作为数据管理方案，在分布式集群环境下则采用了分布式文件系统，希望通过在此二者方案上的实验和分析为今后这类问题的解决提供参考。

基于位置信息的交叉证认存在着一定的局限性，这是因为望远镜等探测仪器的分辨能力有限，从而导致所有的观察数据都存在不同程度的误差。不同的仪器针对不同波段采集数据时，同一个天体的探测误差半径可以小到几角秒大到一度，一般情况下，红外、可见光波段的分辨率优于高能波段。所以要想精确地对不同星表间的数据进行匹配，不可避免地要利用概率、数据挖掘、机器学习等更加复杂的方法针对不同数据进行更有针对性地专门分析[19][20][21][22]，但显然，这样的特殊方法在海量数据的自动交叉证认上并不适合。所以，基于位置的交叉证认虽然难以获得很高的精度，但它是进一步确认的基础，它的证认结果一般可以表示为图 1-3 中的几种形式，天文学家可以利用相关服务，在获得的经过位置交叉证认的结果数据后，根据自己的研究需要选择特定的小规模数据，对于其中一对一、一对多和多对一的结果集，再依照特定的具体方法，如概率分析、数据挖掘、机器学习等方法进行二次判断，从而最终获得精确组合后的多波段数据。而对于图 1-3 中的一对无，多对无的情况，也有着重要的研究价值，科学家可以通过一些特殊分析从这些信息里获得额外的发现。这样就避免了高复杂度算法直接应用于大证认空间上所造成的高昂时间代价。

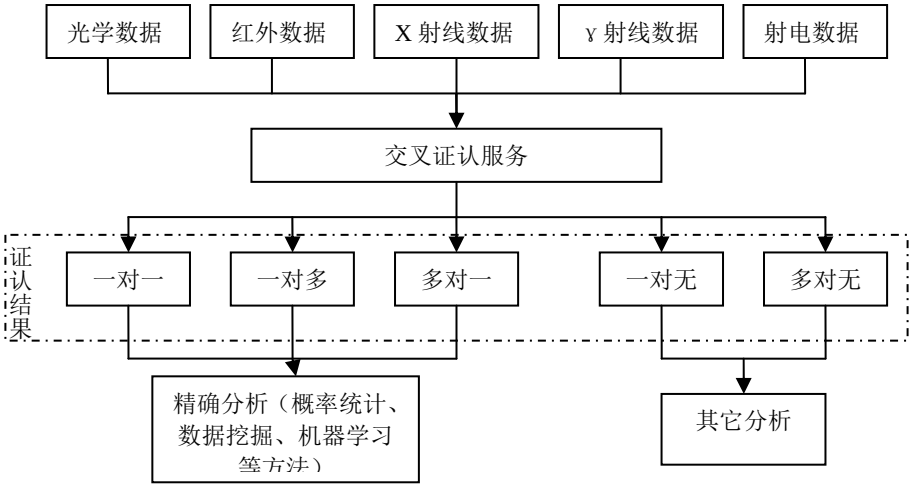


图 1-3 多波段天文数据交叉证认

交叉证认方法看似简单，实际中，它却是构建天文数据访问服务中的重点和难点。经过分析，在对海量多波段的交叉证认计算中，主要存在以下难点：

1) 数据量大，计算复杂度高

当今的探测仪器、望远镜的一个重要特点就是具有强大的数据采集能力，所以要想通过交叉证认来整合全世界各种来源的星表，数据处理量是相当大的。而且，理论上交叉证认的时间复杂度是$O(n^2)$，所以，如果不加优化，其计算时间会相当大，会严重的阻碍新观测天文数据的利用效率。

2) I/O 访问量大，耗时多

根据国内外关于交叉证认的研究报告，它首先是一个数据密集型的计算，而且大多情况下最主要的性能瓶颈也出现在对大量数据的 I/O 访问上。而且，天文数据资源本身的高度分散性所带来的网络上的通信延时非常长，这些都是制约高效实现全球天文数据相互融合的重要原因。

3) 多种算法混合使用

仪器灵敏度、分辨率和误差精度等方面的差别导致单纯位置交叉证认并不能达到精确证认的目的，但数据挖掘等复杂算法要根据数据的不同而进行专门的设计，不但不通用，过大的复杂度也使它们不能直接应用于海量、远程的初始数据。所以，位置交叉证认是所有其他交叉证认算法的基础，应该尽量达到最高的处理效率，除此之外，尚缺乏可扩展的多层次的支持自动并行化的交叉证认服务，以使天文学家在完成位置证认的基础上，可以便捷的加入特定的再处理算法。

对于海量数据的处理，一种最自然最根本的方法就是应用并行计算技术，通过充分地利用更多、更强大的计算资源来获得更高的效率。在计算机技术日新月异的今天，拥有足够强大的计算资源已不再是难事，各种多核处理器、大规模集群设备已经被越来越多的天文观测中心所拥有，然而这并不意味着天文数据处理

的效率能够直接随之提升。特别是考虑到以上归纳的交叉证认计算所存在诸多难点，更可以看出，只有借助并行计算技术、分布式计算技术等高性能计算技术才能最终解决交叉证认的效率问题。简言之，适当的并行处理技术是解决交叉证认乃至更多、更复杂天文数据处理问题的根本方案，同时也是最困难之处，这也是为什么本文要利用并行计算、分布式计算等高性能计算技术针对交叉证认这一天文数据处理中的重要课题展开研究的最主要原因。详细概括起来，本文的研究意义主要有以下几点：

1) 是加深对天体的认识、促进新现象和新理论发现的重要一步。

在当今全波段天文学的背景下，从γ射线、X 射线、紫外、可见光、红外到射电波都有观测项目在进行着。这些数据高度相关，通过对它们进行多波段交叉证认可以在高维空间中对天体的物理性质、演化规律获得更全面更系统的认识，是当今天文学家科学研究的重要步骤。因此，应用并行计算、分布式计算技术来解决交叉证认中因为数据的海量、高度分布等特性所造成的低效问题对提升天文数据的价值、加速天文成果产出有着重要意义。

2) 是虚拟天文台、天文科学数据主题库等数据访问服务的重要模块。

正在全球范围内开展的虚拟天文台项目的一个重要目标就是实现天文数据的统一融合，其核心模块就是交叉证认计算；我国正在进行的天文科学数据主题库项目正在努力整合包括国家天文台、上海天文台、紫金山天文台在内的多家单位的多年积累的观测数据、模拟数据，并构建网络化数据共享环境，交叉证认同样是该项目中实现数据关联的索引层的建立的核心技术。

3) 关系到我国多个自主研发的天文望远镜项目的观察和数据发布。

以国家投资 2.5 亿于 08 年刚刚建成竣工的重大科学工程 LAMOST 望远镜为例，它在利用成像数据进行选源时，需要对各种天体进行识别，这一天体识别过程也与交叉证认紧密相关。此外，LAMOST 作为世界光谱望远镜之王，对全世界天文学发展意义重大，按照天文研究中的惯例，其他手段搜集到的天体资料，如射电、红外、X 射线、γ射线等最终都要通过光谱来确认，由此可见多波段交叉证认将是 LAMOST 数据发布中一个必备的服务，关系到 LAMOST 数据能否被全世界天文界高效的享用。

4) 是高性能计算技术与天文数据处理技术的一次重要结合，对今后相关领域的研究有着重要的参考意义。

随着天文观测数据的迅速增长，基于数据处理的各种天文研究越来越离不开高性能计算技术，本文研究的交叉证认计算问题正是一种典型的数据密集型天文数据处理难题，因此，本课题应用并行计算技术、分布式计算技术对这一问题效率上的解决对今后此类海量天文数据问题的处理方法有着重要的参考价值。

1.2 多波段交叉证认的研究现状

交叉证认计算是多波段天文观测数据融合的基础，也是诸多天文研究的必要步骤。近年来各国的计算机专家都在进行交叉证认这一棘手问题的积极研究。图灵奖获得者吉姆·格雷（Jim Gray）曾是美国虚拟天文台（NVO）[23]负责这一问题的首席科学家，他最早提出解决交叉证认问题必须要依靠并行计算技术[24]。他以自己在数据库方面的深厚造诣，为 NVO 设计了内置于微软 SQL Server 的纯 SQL 指令[25] [26] [27] [28] 的交叉证认服务：OpenSkyQuery，从而为大型天文巡天项目 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。虚拟天文台平台功能最为完备的英国虚拟天文台（AstroGrid）[29]也在其网站的数据查询中提供了简单的交叉证认服务[30] [31] [32]，但在数据规模和效率上仍然没有大的突破。

当前的交叉证认服务和软件还有很多，但同样只支持小规模的交叉证认计算，如 VizieR[33]、Simbad[34]、Aladin[35]、NED[36]等。

VizieR 是法国斯特拉斯堡数据中心开发的数据融合工具。它提供了目前已发表数据的各种查询方式，按星表名查、按日期查、按图像或光谱查、按任务查、按波段或源的类型查等。它所支持的简单交叉证认功能，向用户提供了可上传感兴趣的星表的接口，可自由地在预置星表与自上传星表间进行交叉证认，但证认结果还需要用户二次加工。

Simbad 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。

Aladin 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。在多波段交叉证认方面它的功能较为完善，但同样尚未实现任意海量数据上的交叉证认。

NED 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，主要针对河外源的认证。但对多源的联合查询，它需要通过邮件来完成，而不是在线同步查询，并且也只限于小样本的证认服务。

通过对以上的分析可以看出，当前各个交叉证认工具都有其各自的性能局限，它们都无法进行大规模数据的交叉证认计算，也未考虑对分布式并行计算环境的支持，远不能满足进行大样本工作的天文学家的需求。

近年来，我国对这一问题的研究需求也越来越迫切。刚刚落成的国家重大科学工程项目 LAMOST 望远镜以其超千万条的一夜观测光谱量成为了世界上光谱获取率最高的天文望远镜。按照天文研究的惯例，其他手段搜集到的资料要通过光谱来确认，这一确认过程其核心就是交叉证认，可见多波段交叉证认关系到 LAMOST 数据能否被世界天文界所享用。在这样的背景下，我国在高效交叉证认方面也取得了一定的成果。高丹[37] [38] [39]提出了一种基于 HTM 球面索引和 KD-Tree 的快速交叉证认算法，一定程度上满足了当时国内天文查询服务的需求，但该方法的效率仍只适用于几十万条到几百万条的中等数据量。

1.3 论文主要工作

针对海量数据上的大规模交叉证认问题，通过多种并行计算技术及数据 I/O 处理中的一些优化算法的设计，探索突破其效率瓶颈的可行途径，具体包括：

1) 提出了基于 HEALPix 索引方式的多核环境下的交叉证认算法，同时重点解决数据划分后出现的常见问题——边界漏源问题，在提高交叉证认效率的基础上保证证认结果的完全性。

2) 针对效率上的主要障碍——过于耗时的数据 I/O 处理，通过对数据加载、计算流程的优化，任务分配、调度的改进，以及对稀疏数据的预先过滤处理等方法，研究减少 I/O 访问频率的可行对策。

3) 针对关系数据库在海量数据处理上的性能限制以及快速增长的观测数据对分布式存储环境的需求，研究如何利用 MapReduce 分布式计算模型及文件数据库实现集群环境下的交叉证认，寻求性能突破的新的可能性。

4) 分析研究多核环境及分布式集群环境上的两种交叉证认方法在另一种常用天文数据索引方式 HTM 下是否同样具有可行性，并通过理论结合实验的方法对两种索引方式在实现交叉证认中的性能加以对比分析，为天文数据主题库、虚拟天文台等项目提供综合的技术参考。

5) 研究设计 HEALPix 和 HTM 两种索引方式下的邻域编码快速计算算法，进一步保证本文提出的几种高效交叉证认方法的可操作性，同时为其他相关天文数据访问服务的研究贡献力量。

1.4 论文的创新点

创新点一：针对关系数据库上的交叉证认性能提高的主要瓶颈——大量数据的频繁 I/O 操作所造成的过多耗时，创新地提出了全新的数据加载、处理流程—

—限制生长模型，可有效地减少交叉证认中大量的重复数据查询操作，并在此模型下重新设计了并行程序任务分配、调度基本单元——最大生长块，既保证了较低水平的重复读取操作也实现了对观测数据集中存在的大量天然空白区域的过滤，成功地提高了算法的整体效率。

创新点二：针对 HEALPix、HTM 两种天文数据索引方式，提出了基于位运算的快速邻域编码计算算法，使交叉证认划分后的常见问题——边界漏源问题得到了高效地解决，同时也避免了对原始数据的污染。除此之外，该快速算法在很多其它天文数据处理应用中同样具有重要的应用价值。

创新点三：创新地将 MapReduce 分布式并行计算模型应用于大规模天文交叉证认计算中，用文件数据库代替了传统的关系数据库，突破了海量数据在关系数据库上的性能制约，并通过重新规划数据的分布式存储方式，尽量地减少了证认计算过程中的进程间通信，从而克服了并行程序规模扩展的主要障碍，在较大规模集群上保持了接近线性的加速比，为今后天文数据规模的继续扩展预留了处理空间。

创新点四：本文在交叉证认这一问题的研究较前人更具全面性的特点，这种全面性既体现在对多种并行环境的支持，也体现在摆脱了对单一数据索引方式的依赖，通过理论和实验上的分析已经证明其在当前两大最常用天文数据索引方式——HEALPix 和 HTM 上均有良好的适应性，此外在关系数据库、文件数据库两种实现方式上的实践更为突破交叉证认的关键性能瓶颈——数据的 I/O 操作做出了贡献。

1.5 论文章节安排

第一章指出了研究内容的背景情况，通过分析天文数据处理与计算机信息技术相互结合、相互促进的发展关系以及多波段交叉证认所服务于的虚拟天文台项目引出了多波段交叉证认的研究意义，然后概要地介绍了该课题的国内外发展情况、本文的研究内容及主要创新点。

第二章对相关领域和技术的发展情况进行了综述，包括高性能计算技术在天文数据处理中的应用情况、MapReduce 分布式并行模型的概念及其应用、常见天文球面索引方法的分类及各自特点等。

第三章提出了多核环境下基于 HEALPix 索引的并行交叉证认方法，并解决了研究中的常见问题——边界漏源问题，然后通过理论结合实验的方式验证了该方法在 HEALPix 和 HTM 两种索引方式下的可行性、并对比分析了它们的性能差别。

第四章针对初始并行方法中过于频繁和耗时的数据库I/O操作和大量存在的空白区域提出了 HEALPix 索引下新的数据加载、处理流程——限制生长模型，及任务调度的基本单元——最大生长块的概念，使交叉证认的效率得以继续提升。在此之后，继续讨论了该方法在 HTM 索引下的适应性。

第五章提出了基于 MapReduce 分布式并行计算模型的交叉证认方法，通过分析该模型的设计原则给出了详细的方法设计，并分析解决了该方法实现中存在的效率与存储间的平衡问题。

第六章详细地给出了 HEALPix 和 HTM 两种索引方式下相邻区域编码的快速计算算法，是对本文研究的高效交叉证认方法的重要性能支持，同时也在其他多种天文数据处理工作中有着重要意义。

第七章是对本文全部工作的总结以及后继工作的展望，综合评价了文中设计的几种并行交叉证认计算方法，并指出了今后需要继续深入研究的几个关键点。

第二章 相关工作介绍

2.1 高性能计算技术在天文数据处理领域的应用

望远镜制造技术可以反映一个时代的工程技术所能达到的最高水平[40][41]。由于许多崭新技术的引入，如主动光学技术、自适应光学技术、光干涉技术、薄镜面拼接技术，使观测设备的口径越来越大、技术越来越先进、观测效率越来越高。如同IT界反映计算能力随时间指数增长的摩尔定律[42]一样，在过去十多年中，上述技术的进步使得天文学的数据收集能力也遵循着类似的规律[43][44]。虽然光学望远镜的接收面积每二十五年才增长一倍，但天文探测仪器上装备的CCD像素数每两年即可增长一倍。天文数据收集能力的增加速率正好与计算机科学第一定律摩尔定律的速度相近，而事实上天文数据的处理赖以依靠的正是计算机技术，这两个发展速率上的接近不仅是一个巧合，它更给这两个学科的发展带来了希望和驱动力。

首先，以高性能计算技术为首的计算机技术是对天文科学继续高速发展的有力支持。对于大多自然学科，一般情况孕育新发现、新成果的三个来源分别是理论、实验和数据，这一方法论同样也适合于天文学，只是在这三者之间，数据显得尤为重要，它是理论形成的手段和基础，这是由天文研究的特点直接决定的。首先，天文学研究的对象具有特殊性，并非我们身边的事物而是与我们相隔万里，无法触及的遥远星空。对象的特殊性决定了手段的特殊性，它不能像其他学科那样人为地设计实验，只能通过“被动”的观测去搜集、积累信息，寻找线索。而且这并非是一个简单的过程，由于各种观测仪器的分辨率、灵敏度的限制，导致误差的存在在所难免，同时，由于每个观测仪器只是针对于某段特殊波段设计的，全波段无法直接展现，而在物理意义上它们本身应该是相互关联无可分割的，而当今望远镜的观测却好似盲人摸象一般，单独的数据因为无法展现事物的全貌而限制了它们的利用价值，只有通过交叉证认等处理才能实现数据的整合，还原天体属性的全貌，透视它们的物理本质，然后再通过分析、推理才能获得深层次的理论和发现。这一研究过程可以简单地概括为“观测→积累数据→分析数据→升华理论”，可见，在整个研究链条中，数据的处理是衔接从观测到理论升华的核心环节、关键环节。所以说工程技术的发展所带来的观测能力的迅速提升只是为天文学新现象的发现、新理论的形成提供了新的契机和可能性，而真正要使这种可能性快速转化为现实则必须要依靠对数据的快速处理能力，而这种处理能力最

主要的又是高性能计算技术和网络信息技术。这是因为：首先，遵循着类摩尔定律而逐渐提高的观测能力带来了越来越多的数据量，而且天文观测具有长期积累性的特点[45]，只有在一个相当大的时间跨度里长期地、连续地积累的数据才会具有研究价值，比如开普勒就是在其老师第谷花费毕生精力留下的行星观测资料中发现了三大定律，再如第一颗脉冲星的发现是在距今900多年的历史记载中找到了其形成的证据的[46]。这就使得天文数据并不会随着时间的延续而慢慢失效，它永远只会在时间长河中增量式地加速积累着。因此，天文的数据处理首先就是一个数据密集型的计算难题。其次，从数据到信息，从信息到知识是一个复杂而艰难的过程，对于天文数据来说，天文学家需要借助数据挖掘、模式识别、大规模统计交叉证认、机器学习等复杂方法来对数据进行深层的挖掘和加工，而这其中的很多方法都具有很高的计算复杂度，所以说对于海量天文数据的处理也是一个计算密集型的难题。因此，计算机技术的发展关系着天文科学能否持续高速的发展。

天文学与计算机科学的结合已经持续了很多年，事实上，当今很多重要研究成果的诞生都是得益于计算机技术的发展的。这其中一个最典型的需要依靠强大计算能力、并已成为当今科学家最重要的研究手段之一的就是计算机模拟实验。科学家借助最先进的计算机，模拟了很多重要的天文现象，不仅解开了很多科学谜团，也让枯燥严谨的科学研究绽放出了绚丽的色彩。图 2-1 所示的是美国国家能源部 SLAC 国家加速装置实验室的超级计算机模拟图，图中的低质量双恒星可能仅形成于宇宙大爆炸后 2 亿年[47]。再如图 2-2 中科学家通过超级计算机模拟的黑洞合并过程，看上去颇似两个黑洞在一起跳探戈舞。艾伯特-爱因斯坦的广义相对论预测黑洞合并应当释放强烈的重力波，在时空中产生涟漪[48]。目前科学家正在加利福尼亚艾梅斯研究中心和马里兰州戈达德太空飞行中心通过超级计算机研究如何探测和识别重力波[49]。其实不仅是天文现象，数值模拟在物理反应、药物原理等方面都有着重要的应用。这一方法开始于上世纪 80 年代，限于当时的计算机发展水平，当时的模拟只局限在几万个粒子的规模。但得益于近 20 年来计算机技术的飞速发展，这一领域的进展之快可谓让人惊叹，接连产生了许多重要的研究成果，成了最活跃的研究领域之一。这种研究方法中的一个重要代表就是于 2005 年公布的力求再现冷暗物质模型下的星系形成与演化过程的千年模拟[50]，它涉及的粒子数量达到2160^3，约合10^{10}数量级，实验动用了 IBM p690 超级计算机的 512 个处理器连续工作了一个月之久，产生的总数据量惊达 25TB。可以想象，如果不是得益于计算能力的提高，面对如此规模的数据量和计算量，科学家根本束手无策。然而，这一轰动一时的千年模拟很快就被更大规模的 N 体宇宙学工作所超越，在其稍后发表的地平线模拟所计算的粒子数高达4096^3个，

覆盖尺度约占整个可见宇宙的一半，计算时所动用的硬件规模更为惊人，在法国CEA超级计算中心的BULL超级计算机上利用6144个CPU连续运行了两个月。除此之外，还有很多高性能计算技术在天文数值模拟中成功应用的例子，比如关系到第一代恒星形成过程的宇宙再电离过程的数值模拟研究[51] [52]、对太阳耀斑演化过程的数值模拟等。

图 2-1 低质量双恒星

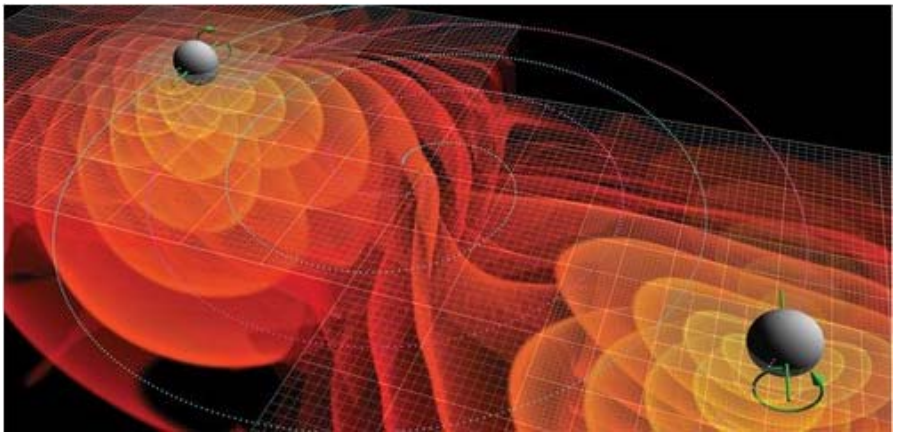


图 2-2 黑洞合并模拟

近两年来我国天文学也取得了长足的进步，其中一个颇为重要的方面就是天文学观测研究方面的进步。在2009年国际天文年纪念大会上中科院常务副院长白春礼曾经提到[53]“我国天文学家在宇宙物质分布、暗物质粒子性质、星系形成与演化的数值模拟、银河系磁场的测量、银河系化学演化、以及太阳活动机理研究等前沿领域做出了一系列具有国际影响力的工作，特别在暗物质和宇宙大尺度结构领域的一系列工作”。而这些成果很大程度上都是在高性能计算机的强大的计算和存储能力的基础上诞生的，或是源于对大量数据的统计分析，或是计算量巨大的数值模拟。这些成功和我国近些年来大力致力于大型超级计算机的研究策略密不可分，伴随着我国在计算机技术上的快速崛起，可以预料，今后还将有更多

震惊世界的成果问世，必将使我国现代天文学研究的国际影响达到前所未有的高度。2004年我国研发并部署在上海超级计算中心的曙光4000A曾以11.2Gflops的峰值Linpack性能历史性地闯入了世界超级计算机十强之列。而到了2008年，同样位于上海超级计算中心的曙光5000A再次摘到了TOP500.org组织的全球计算机五百强榜单[54]中的第十名，成为了前十名中唯一非美国研制的机器。同时，在TOP500的综合排名实力上，中国也取得了可喜的成就，一共有16台我国自主研发的计算机挤进了这个榜单，比08年多了4台，成为继日本（18台）之后成绩最好的亚洲国家。在这次满载而归之后，仅过了一年时间，我国再次刷新了历史最好记录：在2009年第34届全球超级计算机500强榜上，我国国防科技大学研制的“天河一号”名列世界第五、亚洲第一，并且成为我国首台千万亿次计算机。中科院软件所研究员张云泉对这一成绩给出了高度评价“‘天河一号’第一次进入全球前五名，标志着中国超级计算机的峰值性能达到世界先进水平。同时，‘天河一号’是第一台采用cpu/gpu混合异构系统的超级计算机，体现了体系结构的创新。”

2.2 并行程序设计及PCAM方法学

随着天文观测数据的不断增长，以及并行计算硬件平台的不断发展、推广与普及，并行计算技术将成为天文数据处理中的一种重要手段。在并行程序开发过程中，并行程序的设计方法极其重要，其指导了并行程序开发系统的建立和实现过程，决定了最终的并行加速效果。

在程序开发上，与串行程序开发相比，并行程序开发所要考虑的因素更为复杂，因为并行程序面对的是多线程、多进程间的控制，同时需要更大的状态空间。并行程序员必须考虑同步原语、锁定设计、临界区识别以及更大状态空间上下文中的死锁[55]。

在程序设计上，为了获得优秀的并行加速效果，一般会按照四个关键过程展开程序的设计，即任务划分（Task Partitioning）、通信分析（Communication Analyses）、任务组合（Task Agglomeration）和任务映射（Task Mapping）。因此，通常将上述并行程序设计方法统称为 PCAM[56][57]（Partitioning，Communication，Agglomeration，Mapping）方法学。

1) 任务划分，指对所处理的问题进行分析，尤其是任务的并行性分析，从而将解决该问题所需执行的任务划分成若干个数据集和计算集互不相交并且可并行执行的子任务。一般而言，任务划分的方式可分为两类，即域分解[58]（Domain Decomposition）划分和功能分解[59]（Functional Decomposition）划分。其中，域

分解指通过数据的并行性分析将任务执行过程中需要或产生的数据分解成若干块数据规模大致相等并且可被并行处理的小数据块，进而以此为依据将任务划分成若干个可被并行执行的子任务；而功能分解则是通过计算目的和过程的并行性分析将任务划分成若干个可被并行执行的子任务。对于域分解，需要考虑划分后各子任务间的通讯操作问题。若某个子任务需要使用另一子任务中的数据，则这两个子任务间便需要进行通讯操作。对于功能分解划分，则需要对划分后各子任务所需的数据进行分析。若各子任务所需的数据集互不相交或只存在少量交叠，则划分方案可行；若各子任务所需的数据集间存在过多的交叠，则需要重新进行功能分解，直至划分方案可行为止。

2) 通信分析，指对任务划分后各子任务间的关系进行分析，包括时序依赖、信息交换、同步等，以确定各子任务间的相互依赖关系。由于任务划分所生成的所有子任务一般不能都满足完全独立执行的要求，即部分子任务间需要进行数据交换和通信操作，因此需要进行通信分析以指导各子任务执行方式的设计和实现，进而保证任务的可执行性。此外，通信分析也进一步确定了各子任务间的并行性。若某两个子任务间存在着相互依赖关系，如需要进行数据通信操作，则这两个子任务不是完全独立并行的（即不能被完全并发执行），这也就限制了任务的并行性，也即，子任务间的通信需求限制了并行性。其中，任务间的通讯模式可根据通信性质分成四大类，即局部和全局通信模式、结构化和非结构化通信模式、静态和动态通信模式以及同步和异步通信模式。综上所述，通信分析过程是PCAM 方法学的重要阶段，对于任务的执行至关重要。

3) 任务组合，指根据通信分析过程所得到的各子任务间相互依赖关系的紧密程度以及各子任务具体实现过程的复杂程度，将联系紧密的多个子任务组合成一个大子任务，即对子任务进行优化组合处理。其目的是减小子任务总数，以降低子任务间的交互性和提高子任务间的并行性，进而简化编程实现过程和提高程序的执行效率。对于任务组合而言，若组合后的子任务数量与计算机系统中处理器数量是相等的，则很容易完成子任务至处理器的一一映射以及子任务的分配与执行。但在解决实用的并行性问题时，一般很难达到此理想目标。此外，虽然任务组合通过增加子任务的粒度和重复计算减小了子任务数和子任务间的通讯量，但其同时也增加了子任务的计算量。因此，应对组合后的子任务数和计算量进行折衷处理，使两者保持恰当的平衡性，以尽量减少算法的总运算时间和提高程序的执行效率。

4) 任务映射，指将子任务映射到计算机系统中的处理器上，即将各子任务定位到具体的处理器上进行执行。PCAM 方法学的前三个过程均是从程序的并行性设计角度来对任务进行划分和对子任务进行组合，未考虑最终的并行应用程序

在具体的并行计算硬件环境上的执行过程。由于目前的并行计算硬件环境有多种类别，各种并行计算硬件环境均拥有不同的体系结构和互连结构，直接影响和决定了任务的映射方式和执行方式，因此，在对子任务进行映射和部署前，需要对具体的并行计算硬件环境的特点及其物理资源进行分析，并对组合后的子任务在具体的物理处理器上的映射进行分析，以选择合适的任务映射方式，使其满足具体的并行计算硬件环境的物理资源的限制，从而优化子任务的物理执行性能。在任务映射过程中，一般情况下任务组合的子任务数均大于物理处理器数，即需要将多个子任务映射至同一处理器上，这使得各处理器存在负载不平衡问题和任务调度问题。为提高任务的执行效率，通常采用的方法是将相互间存在较多通信的子任务映射至同一处理器上，以降低处理器间的通信量和提高通信速度。

综上所述，任务映射需要根据子任务和具体的并行计算硬件环境进行权衡和负载平衡处理，这些问题属于 NP 完全问题[60] [61] [62]。为使各处理器达到近似平衡的负载，通常需采用一些负载平衡算法，如静态负载平衡算法、动态负载平衡算法、基于概率的负载平衡算法和基于域分解的负载平衡算法。

虽然 PCAM 方法学包含了四个按步进行且相对独立的过程，但在实际的并行程序设计过程中，这四个过程可同时进行考虑，甚至可以相互交叉的形式进行实现。此外，实际的并行程序设计过程往往很难通过一次性完成上述四个过程来实现，通常需要进行回溯反复的多次迭代来实现最终的设计方案。本文在进行天文交叉证认的并行算法设计时，也是按照 PCAM 的并行程序设计方法进行程序优化的。

2.3 MapReduce分布式并行模型

MapReduce[63]是Google在2004年提出的一个编程模型，并已于2010年年初正式申请获批该项技术的专利。它主要用以进行大规模数据集上的并行运算，其主要概念“Map（映射）”和“Reduce（规约）”最初借鉴于Lisp和其他函数式编程语言。Lisp在上世纪50年代就已诞生，Google可谓让这种古老的思想重新燃起了生命力，在Google内部，MapReduce得到广泛的应用，比如分布排序、Web连接图反转和Web访问日志分析等。除此之外，Yahoo!的搜索基础设施，Amazon的Elastic MapReduce[64]服务，IBM的M2平台等很多号称云计算、云服务的企业级应用都以MapReduce为核心技术。在2009年InfoWorld推出的“2009年十大新兴企业级技术排名”中，MapReduce成功地超越了跨平台移动应用开发、NoSQL数据库、重复数据删除及桌面虚拟化等技术，一举摘得了该项排名的第一名。

众所周知，对于大数据量的计算而言，并行计算是最常用、最自然的处理手

段，但其繁琐的编程方式使其对许多开发人员来说，还是一个比较遥远的东西。MapReduec一经推出之后即大获欢迎的一个原因就是它简化了并行计算的编程模式，让那些没有并行计算经验的非专业人士也可以开发高性能的并行应用。当然，MapReduce在商业上成功的另一个关键则得益于它使原本必须借助于非常高昂的超级计算机才能获得的计算能力可以在大量廉价机器上同样实现，大大地减少了开发成本、缩短了开发周期。

本文的研究内容之一就是基于MapReduce模型的分布式交叉证认，之所以选择了这一模型同样出于上面两点优点的考虑。对于天文数据处理人员来说，复杂的并行编程一直是困扰他们的一个难题，MapReduce所带来的简易并行编程方式正好解决了这一问题。本文希望可以借助于基于MapReduce的交叉证认这一特定应用的研究，为今后天文数据处理提供一种新的思路。另一方面，天文科学本身具有非盈利性特征，在各方经费并不十分充足的时候，这种利用大量廉价机器获得高性能计算资源的方式本身就具有现实意义。

2.3.1 MapReduce模型的基本原理

Google作为一个搜索引擎公司，每天都要处理堪称海量的网页原始数据及中间数据。诸如网页爬行、搜索请求日志、倒排索引、Web文档的图结构标示等计算任务在原理上都很简单，容易理解，然而考虑到其所面向的巨大数据量和有限的时间之间的矛盾，就必须要采取一些特殊的处理，通过利用成千上万的机器获得足够强大的计算力才可以完成目标。在这样的背景下，Google公司的Jeffery Dean——09年新当选的美国工程院院士，于2004年设计了一个新的并行计算模型，即MapReduce，将任务分发、任务调度、数据分布、容错处理、负载平衡等并行计算中不可避免的复杂控制细节隐藏于系统的运行时后台处理中，对程序设计人员完全透明。它的主要贡献就是通过简单而强大的接口在大量普通PC机上实现了一类问题的自动并行化和大规模分布式计算。

MapReduec的名字来源于它的两项基本操作：Map（映射）和Reduce（规约），这两个概念在函数式编程中非常常见。简单地说，Map是把最初的待处理输入数据一对一地映射为另外一组数据，即<key, value>二元组，其映射的规则由用户定义的一个函数进行具体指定。Reduce是对Map产生的中间二元组数据一组一组地进行规约操作，这个规约规则同样由用户定义的函数给定，以得到最终的计算结果。Map过程针对每个输入数据独立进行，因此可以实现高度的并行化，而其产生的结果因为产生于各个节点上，天然具备并行的分布特点，只是为了Reduce过程的高效性，在输出时会根据二元组中的key值进行重新的排序、分发，以保证各个Reduce任务可以独立地完成于各自的节点上，避免不必要的节点间数据通

信。图2-3形象地描述了这两个过程。

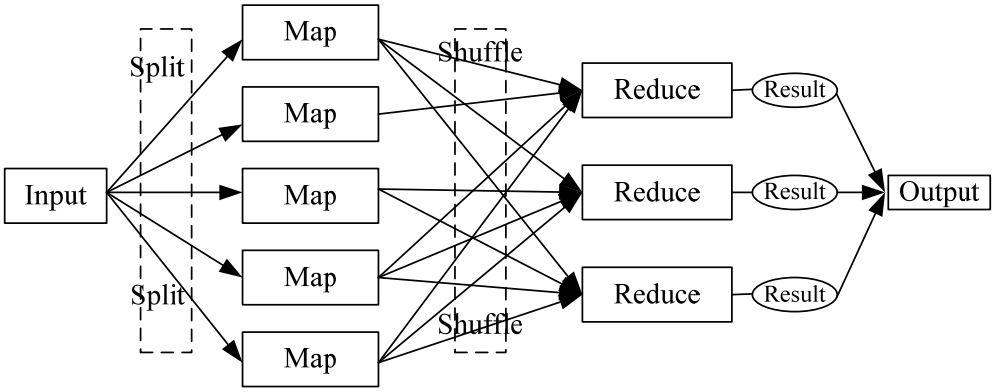


图 2-3 MapReduce 原理示意图

从上面的工作原理可以看出，MapReduce适合处理计算任务相对独立的数据密集型问题，与搜索相关的几个典型环节均可应用MapReduce实现，从中可以进一步了解MapReduce模型的功能：

✧ 分布式Grep操作：

Grep是Unix类系统中的常用工作程序，其功能是实现文件内的字符串查找，它在搜索引擎中是一个常用基础性操作。在Map步骤中，如果输入行与给定字符串相匹配，就输出这一行，Reduce函数完成中间数据向输出的复制。

✧ 计算URL访问频率：

搜索引擎的用户使用日志对搜索性能的优化至关重要，map函数处理web页面请求的记录，输出<URL，1>二元对集合。Reduce函数负责累加相同URL的value，产生<URL，记录总数>统计结果输出。

✧ 倒转网络连接图：

对网络连接图的链接统计是实现搜索中页面评分的关键一步。Map函数为每个链接输出（目标，源）对，其中目标指锚链接中的目标URL，源指该锚链接的来源页面。Reduce函数对相同的目标URL执行聚合操作，产生<目标，源列表>对。

✧ 倒排索引：

倒排索引是实现搜索高效的关键。Map函数读入每个文档，然后输出由<词，文档编码>二元组组成的序列。Reduce函数接收一个给定词的所有二元组，排序相应的文档编码，然后产生<词，文档编码列表>对。

除了上述简单计算之外，MapReduce在数据挖掘、机器学习等计算密集型应用中也有非常好的利用价值[65][66][67][68]。

2.3.2 MapReduece模型的开源实现

1. Hadoop:

Hadoop[69]是一个Java实现的支持MapReduce模型的分布式计算开源框架。它最早是Apache的Lucene[70]搜索项目中为支持其Nutch[71]搜索引擎的分布式并行处理而设计的。从Nutch的0.8.0版本开始，Apache将其中实现的NDFS和MapReduce剥离出来成立了一个新的开源项目——Hadoop，从此它不再只是支持分布式搜索的框架，更成了一个完整的分布式计算平台。它主要由两部分组成，其一是一个可以支持几千台机器的分布式文件系统，即Hadoop Distributed File System[72][73]，简称HDFS，类似于Google的GFS[74]；另一个是它的运行时环境系统，即实现MapReduce功能的核心引擎。

1） HDFS

HDFS是一个可以部署在廉价硬件设备之上的分布式文件系统，很适合于面向大量数据的应用，并且提供了对数据读写的高吞吐率。HDFS具有master/slave结构，通常情况下，master上运行着一个全局的Namenode，而每个slave上运行着各自的Datanode。Namenode是整个文件系统的管理员，维护着整个文件系统的元数据（metadata），对系统中任何文件的诸如建立、删除等的操作都是通过Namenode来控制的。Datanode是系统中文件的实质载体，它存储和维护着文件的各个片段。图2-4描述的HDFS的基本结构[75]。

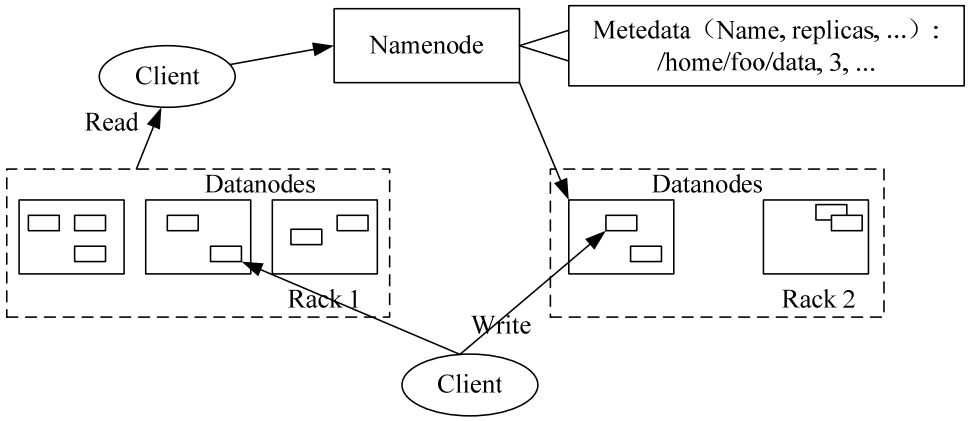


图 2-4 HDFS 基本结构

从图中可以看出，Namenode是该文件系统的逻辑层，隐藏了底层的真实数据存储情况。用户只需设定每个文件的名字、数据来源和副本数量等信息，HDFS就可以自动实现对大文件的分块，并按照副本数量的设定向各个Datanodes存放这些数据块，因为其元数据一直由Namenode进行维护，所以当用户提交一条命令

（读、写、删除等）时，Namenode就可以自动驱动各个Datanode上的实际操作。以写入操作为例，当Client要执行一条写入操作时，命令并非立刻发送给Namenode，Client首先在本地的临时文件夹中缓存这些待写入数据，当临时文件夹中的数据块达到了设定的块大小时，Client才会通知Namenode，Namenode便响应Client的RPC请求，将文件名插入文件系统层次中，并在Datanode中找到一块存放该数据的块，同时将该Datanode及对应的数据块信息告知Client，此时Client才将这些本地临时文件夹中的数据写入指定的数据节点中。

为了保证数据的可靠性、安全性和高效性，HDFS采用了副本策略。默认情况下，其副本数被设为3，即一个存放在本节点上，一个存放在同一机架中的另一个节点上，还有一个放在不同机架的另一个节点上。

2) MapReduce引擎——Job Tracker和Task Tracker

运行于Hadoop分布式文件系统之上的是MapReduce引擎，它是维护分布式程序运行的运行时系统，它的存在才使MapReduce模型中的一个核心功能得以实现，即向用户隐藏并行实现中任务分发、任务调度、数据分布、容错处理、负载平衡等复杂控制细节。MapReduce引擎包含两个核心概念：Job Tracker和Tast Tracker。Job Tracker是维护程序运行的主进程，它负责将处理任务分发给集群中处于空闲状态的Task Tracker，并且保证计算可以尽可能地与所需数据相接近。因为其下分布式文件系统的维护，Job Tracker可以获知哪些节点包含这些数据，哪些节点相对较为接近。比如说，如果执行任务无法安排于数据所在的节点上，则最优的选择是选取处于同一机柜的另一个节点。Job Tracker和Task Tracker联合起来形成了一个多层次的任务调度、维护系统。如果某个Task Tracker出现了问题，则处于它上的工作部分将被Job Tracker重新安排分配，如果Job Tracker发生错误，则全部正在运行的工作都将可能丢失。

一次MapReduce程序在Hadoop上的执行情况是：在正式执行之前，master节点的Namenode、SecondNamenode、JobTracker和各个slave节点的Datanode、TaskTracker先被启动起来。然后每一个执行任务（一个Map任务或Reduce任务）都是由JobClient通过远程过程调用提交作用，而TaskTracker在启动之后一直处于不断向JobTracker发送心跳的等待状态中。如果JobTracker的作业队列不为空，则TaskTracker发送过来的心跳将会获得其中一个任务。TaskTracker收到任务之后，就开始在其本地展开调度执行。图2-5形象地描述了这一基本过程。TaskTracker在分配了任务之后，JobTracker还起到执行时监控作用，这种监控同样通过心跳的机制进行，即每隔一小段时间，TaskTracker就像JobTracker发送状态信息，JobTracker接到心跳后，首先检查上一个心跳响应是否完成，是否要求启动或重启任务，如果一切正常，则会处理心跳，如果发现某个Task执行失

败或超时，就重新分配该块任务。

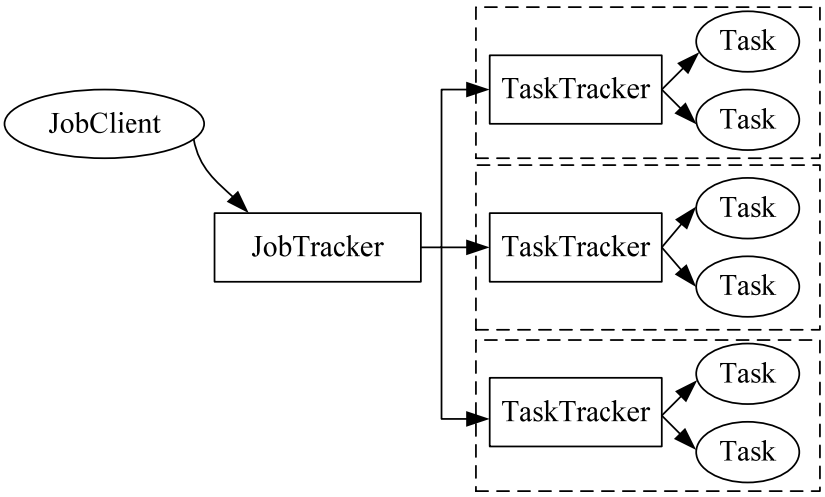


图 2-5 JobTracker 和 TaskTracker

从上面的描述可以看出，Hadoop分布式文件系统和Hadoop的MapReduce引擎是一个相互协作的整体，它们的相互配合达到了对硬盘和CPU的最大限度利用，非常适合于处理面向海量数据的计算任务。

2. Phoenix:

MapReduce模型另有一个基于共享内存体系结构的开源实现：Phoenix。它致力于多核平台上的并行程序高效实现，并将琐碎的运行管理工作交给后台的运行时系统处理，包括并发管理、资源管理和错误修复等。下面先介绍下Phoenix的运行时系统（即Phoenix中的MapReduce引擎）的工作流程。

如图2-6所示，调度器是核心控制单元，它负责建立和管理所有运行的Map线程和Reduce线程，同时也管理着用于任务间通信的缓冲区。初始化工作完成之后，调度器为每个处理器建立一个worker线程，Map和Reduce任务会动态地分配给这些线程。

开始时，调度器调用Splitter将用户输入数据划分为一些大小相等的单元，然后当Map任务调用Splitter时，就分配给它需要处理的数据单元。Map任务构建完成后动态地分配给各个worker线程。Partition函数对Map任务的输出数据执行排序再分配，以保证相同key的所有value都会转到同一单元，以便于Reduce任务的处理。调度器判断到所有Map任务都已返回后则开始Reduce任务。

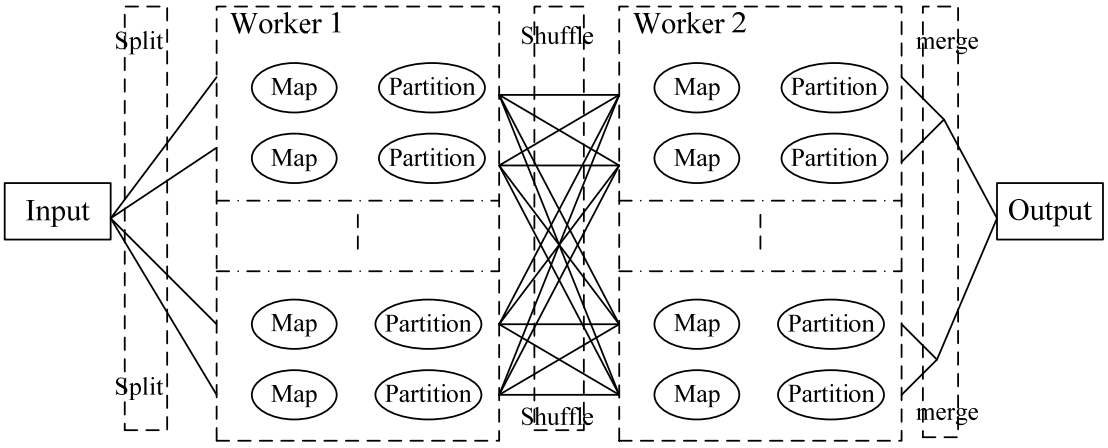


图2-6 Phoenix运行时系统基本工作方式

Reduce阶段同样采用了动态任务分配方式。因为Reduce必须在同一个任务中对相同key的所有value执行规约，所以各个worker可能会出现负载不平衡的情况，因此动态分配更为重要。

下面介绍Phoenix的容错机制。Phoenix以其它worker上执行类似任务所需的时间为标准检验正在执行的任务是否超时。当前Phoenix尚不能判断超时的出现导致的是完全的失败还是未完成的数据，也不能阻止有错误的任务污染共享内存区。当有错误发生时，运行时系统会重新分配该任务到另一个worker，由于原失败任务尚在进行，新启动的替代任务会获得新的输出缓冲区，否则新任务的地址空间会与原任务的发生冲突。运行时系统会采纳该两个任务中先执行完毕的任务，并把它的结果合并到该环节的输出数据中。在某个worker刚刚出现错误的初始阶段，调度器只假设这是一次意外，会继续向这个worker分配后续的任务。但如果此worker在很短时间内再次出现错误，或者在一段时间内频繁地出现错误，调度器则会认为此worker出现了某些延续性的严重错误，从而不会再向它分配任务了。

2.4 天文数据索引方法

2.4.1 索引在天文数据处理中的重要性

对于海量数据的处理，建立索引是提高其访问速度的必不可少的重要手段。以搜索引擎为例，用户之所以可以在零点几秒的时间内从浩瀚的网页海洋中得到相关的返回结果，其重要的原因就是已经对网页中的词语建立了倒排索引。本文的研究中，对于存储于数据库中的海量天文数据来说，索引更是不可或缺。天文数据的增长速度遵循着类摩尔定律，大概每两年就会翻一番，而计算机处理器的

发展当前还可以保持每 18 个月增长一倍的速度，与此同时，存储介质的技术发展也遵循着每两年或更短时间内，单位价格容量增一倍的规律。这就说明，在处理天文数据中，得益于计算机计算能力和存储能力的快速发展，天文数据的存储和计算的需求可以比较容易的得到满足。但与此呈鲜明对比的是计算机 I/O 的带宽和寻道时间的增长滞缓：每年大约只能增长 10%左右，远远小于天文数据规模的扩张速度。因此，数据的 I/O 处理是天文数据处理中最困难之处。这一结论与当前很多天文数据处理方面的文献中所给出的结论正好吻合。本文所研究的大规模交叉证认问题也同样面临着这样的困难。因此，要想提高天文数据访问、处理的效率，克服制约其性能提高的最主要瓶颈，就必须要依靠数据索引技术。

对存储于数据库中的天文数据建立索引，其实质就是通过对球面进行分割来过滤掉大量的不相关的数据，从而实现对所需数据的快速定位，已实现访问性能的提高。对于交叉证认这个具体的天文数据处理任务来说，建立索引对其又有着一些特殊的作用，具体来说包含以下几点：

1) 如果按照原始的交叉证认方法，两个星表间的每两点数据之间都要取出并进行一次距离计算，其复杂度高达$O(n^2)$，数据读取耗时、计算耗时都非常巨大。一种常用的方法就是要对数据进行划块处理，从而限定需要证认的范围，过滤掉大量无用的计算。而选择一种适当的索引方式正好可以实现数据的划分，可谓一举两得。

2) 交叉证认是典型的数据密集型计算，由于天文数据量巨大，内存空间无法容纳，所以在整个计算过程中，硬盘访问次数繁多。所以，对数据库中的数据建立索引是减少数据库I/O操作的关键步骤。

3) 对于交叉证认问题所面临的海量数据，并行计算技术非常重要。通过建立索引实现了数据的区域划分，是实现数据并行化处理的第一步。

4) 由于天文数据具有全球范围内的广泛分布性，大批量的传输数据到本地无论在存储空间还是在传输耗时上都不现实，所以多源数据间的交叉证认会向着分布式计算的方向发展。索引的建立自然的实现了数据块的划分，完成了多源间数据位置上的关联，便于数据的定位和传输，可见，索引是后继实现分布式交叉证认服务所必须的。

5) 建立索引后，可以快速地判断出各个区域的天体的密度。在并行交叉证认方法中，拥有密度信息有利于负载平衡的实现，也有利于对空白区域的预先过滤；

而事实上，索引对于交叉证认的意义还不止于此。在天文数据访问服务中，交叉证认是其中的一个核心模块，实际应用中它往往会和其他数据访问功能联合使用，在构建服务时交叉证认与其它服务也共同建立在同样的数据层上；另一方

面，交叉证认又是其它更为复杂的数据分析方法的前奏，它和后续的数据挖掘、机器学习、概率统计等分析又息息相关。所以说，考虑到应用背景，建立索引不仅提高了交叉证认过程中的效率，还在整个天文应用中起到了至关重要的作用[76]，具体包括以下几点：

1）好的索引算法，可以帮助提高一系列查询的效率，包括锥形查询、相邻区域查询、几何重叠区域查询以及和交叉证认相关的联合查询等，在虚拟天文台的整个数据访问服务模块中都必不可少。

2）经过索引，完成了数据的区域分割，可以方便的记录下各个星表中各个区域的天体密度，对虚拟天文台中的可视化服务至关重要。

3）索引实现了数据的分割，同时也实现了不同星表间区域位置上的关联，是多星表上数据挖掘的重要手段。

4）各种复杂天文数据处理都面临着效率的问题，并行计算技术是解决这一问题的重要途径，而通过索引实现的区域划分是实现并行计算的前提。

综上所述，无论是交叉证认计算本身还是与交叉证认相关的背景服务都离不开索引技术。根据天文观测数据在位置信息上所具有的球面二维特性，用于天文数据的常用索引方式可以分为一维索引、空间多维索引和伪二维索引三种。索引的选择直接关系到后续算法的设计、并行的实现等方方面面，本章后续几小结将分别给出球面空间索引的特点、各种索引实现方式及其优缺点，以及针对于交叉证认计算的适应性分析。

2.4.2 球面索引的特点

考虑天文数据的索引问题时，一定要考虑其所具有的一些特性。在天文研究上，通常都是把天体等研究对象置于一个名为天球的坐标上。如图 2-7 所示，天文学上就把图中这个以地球为中心，以无限大距离为半径，内表面分布着各种各样天体的球面称为天球[77]。因与地球相对应，所以命名为天球，相对于地球上的赤道、两极、经度、纬度等概念，天球上也有相应的天赤道、天极、赤经和赤纬。因为天体和观察者间的距离与观测者随地球在空间移动的距离相比要大得多，人的肉眼分辨不出天体的远近，所以看上去天体似乎都离我们一样远，这样，从地球的观测角度上看，全部的日月星辰好像都分布在天球的内表面上，而实际上我们看到的是天体在天球球面上的投影位置。

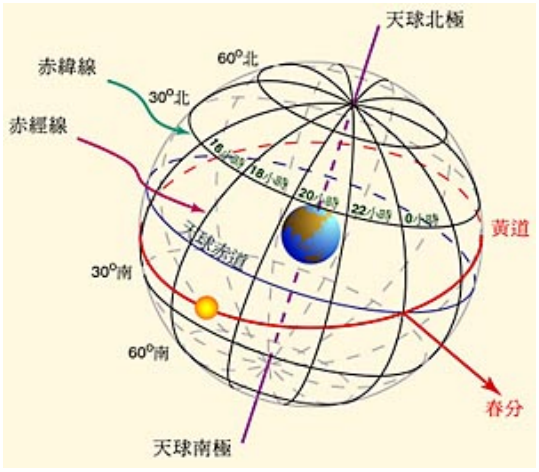


图 2-7 天球

因为球面上处理点和弧的关系，比在空间处理视线方向间的角度要简便得多，因此天文学的应用中大都用天体投影在天球上的点和点之间的大圆弧段来表示它们之间的距离关系。本文对交叉证认的研究中，两星体间的距离同样使用的这一计算方法。

球面的特性使对天文观测数据的索引有别于传统笛卡尔空间上的二维索引。归纳起来，球面空间的索引建立问题主要存在以下几个特点和难点：

1) 要对赤经和赤纬两维信息建立索引，而当今的二维空间索引技术尚不成熟，只有少数数据库管理系统提供了对多维空间索引的支持，而且其性能还不完善，效率上远远低于传统的一维 B-Tree[78]索引。

2) 即使多维空间索引方法的性能发展到可以容忍的地步，考虑到球面空间与笛卡尔空间的区别，直接应用这些球面索引也仍然存在着一些问题[79]。其一，球面坐标在赤纬这个维度上并不均匀，从赤道开始向两极逐渐发生变形，而到两极附近，变形变得非常明显。这种扭曲的存在导致直接应用传统笛卡尔空间二维索引方法时会出现各个区域面积不均等的情况，这将严重影响交叉证认中根据误差半径来确定证认范围的操作，同时也不利于并行程序实现时的负载平衡。其二，球面空间上的赤经坐标存在着圆形环绕的特点，既 360 度与 0 度相衔接，这种特性也需要加入额外的处理。

3) 在进行交叉证认计算时，因为数据本身存在着误差，所以对数据库的查询类型大多是区间查询，而非精确点值的匹配。因此，在二维索引的选择上最好能拥有良好的区间查询效率。

球面索引技术的这些特殊性的存在加大了天文数据处理的难度，多年来，天文学家和计算机学家都在致力于这方面的研究，并已提出了多种基于不同原理的索引技术。

2.4.3 一维B-Tree索引方法

由于球面索引存在的上述困难，一些天文数据处理中只针对赤经赤纬中的一维进行了索引，而考虑到赤经一维上存在的坐标环绕特性，又通常都选择了只对赤纬进行索引。这种方法下就可以利用当今最为成熟的索引技术——平衡二叉树（B-Tree）方法了，它与各种数据库管理系统中的查询优化器完美的结合在一起。大量的实验已经证明，B-Tree 索引具有非常好的综合性能，在建立、插入、删除、查找等各种操作上都具有很好的效率。与其它的空间索引方法相比（如 R-Tree、Kd-Tree、G-Tree 等），它的性能优势非常明显，并且它还能够保持良好的平衡结构，减少了对空间的过多消耗。

在数据库中，B-Tree 索引减少了定位记录时所经历的中间过程，从而加快了存取速度。B-Tree 索引所具备的优良性能使它成为了最为通用的数据库索引方法。而这种通用性又使得依赖 B-Tree 方法建立索引的服务可以具有广泛的可扩展性，因此在小规模数据集的交叉证认问题上，B-Tree 索引还具有一定的应用价值。本文在关于交叉证认的研究中也同样设计了赤纬单维 B-Tree 索引下的交叉证认计算算法，并在实验中给出了性能对比分析，这在本文的第三章中将有详细的描述。

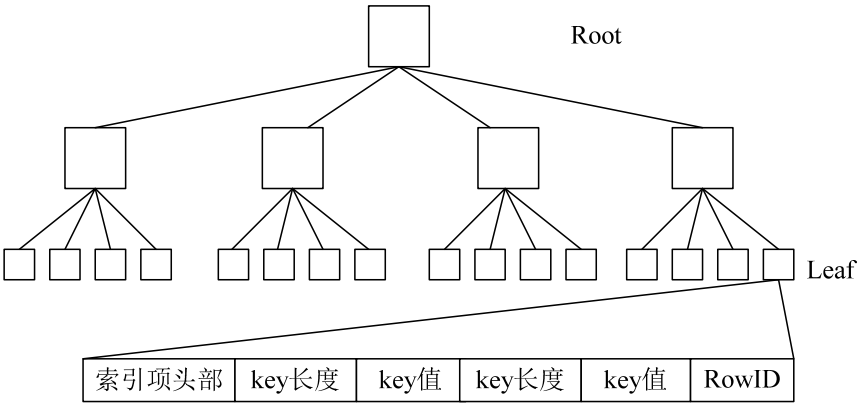


图 2-8 B-Tree 索引搜索过程

这种方式在简单数据查找中一般可以满足性能的要求，但随着数据量的增加，这种性能的隐患不可避免，而对于复杂的大规模联合查询，无论在效率上还是可扩展性上这种方法都存在着严重不足。这是因为，基于赤纬的单维索引只是快速地定位了球面上一个维度跨度上的一圈数据，远远大于查询时通常指定的一小块区域，所以从一圈数据到小块数据间的再次定位又需要逐条的顺序扫描，耗时可想而知。

2.4.4 空间多维索引方法

空间索引技术是数据库领域的一个热门的研究课题，近年来多种多维索引方法已被提出，如 R-Tree[80][81]、G-Tree[82]、Kd-Tree[83]、BANG file、BV-Tree、Cell Tree、Gridfile、hB-Tree、LSD-Tree、P-Tree，PK-Tree 等，但它们都存在着不同程度上的缺憾。这种性能的不理想其实是由多维空间的复杂本质所决定的。拿二维空间表面举例，从直观上讲，二维空间上的点不可能实现线性序列化，也就是很难完全保证位置上相近的点同属于同一父节点的目标，在分界位置上总是存在着跳跃性突变。在这些多维空间索引方法中，历史最悠久、应用最广泛的当属 R-Tree 方法。图 2-9 是对 R-Tree 方法原理的一个形象化的描述。

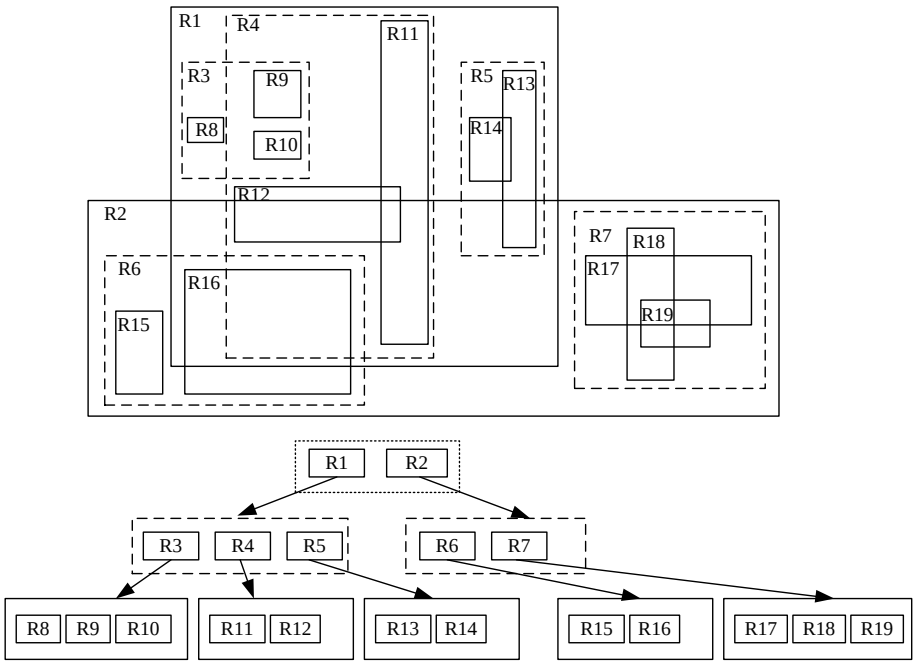


图 2-9 R-Tree 索引方法原理

从图中可以看出，它在分割思想上与 B-Tree 类似，不同的只是针对的是多维空间而非一维线性空间。它同样以层次递归的方式将空间逐级划分为有可能相互重叠的最小边界矩阵（minimum bounding rectangles, MBRs），并且其中的每一个节点可以具有不同数量的对象体。针对于天文交叉证认这个具体的应用来分析，考虑到前面所总结的天文数据索引问题所具有的特点，可以看出，直接应用 R-Tree 建立索引仍然存在一定的问题：

1）R-Tree 的划分几何结构呈矩形，不可避免的会把一部分落于误差圆圈之外的数据包括在内，从而增加了后续精确匹配的计算量。

2）从赤道向两极的过程中存在着不同程度的赤经扭曲现象，越是接近两极

就越严重，这种现象导致如果使用 R-Tree 索引方法，必须要考虑在不同纬度上对赤经进行修正，才能满足数据误差半径的要求，而这种修正又将导致耗时的三角函数的引入。

3）球面数据赤经坐标在 0 度存在环绕现象，需要在代码中进行额外的处理，而不能直接利用卡迪尔空间的 R-Tree 索引方法。

4）索引的建立会引入额外的数据存储量，R-Tree 所实现的空间划分所需的存储消耗无疑要多于 B-Tree 索引。

虽然 R-Tree 已被部分开源数据库所采用，如 PostgreSQL、MySQL、Informix 等，但它尚缺乏适用的并发控制机制来保证并发环境下数据的一致性，且无法做到在查询、插入、删除、查找及索引的建立等各个方面的综合性能优越。Kalpakis[84] 曾在他的文章中给出了使用 Informix 数据库的 R-tree 索引技术来为 USNO-A1.0 星表建立索引的实验，结果显示无论是数据加载还是索引的建立都非常耗时，其中为整个星表建立 R-Tree 二维索引居然用时 25 天。Clive Page 也对多种数据库的空间索引（主要是 R-Tree 索引）做了性能测试实验[85]，结果同样显示数据库空间索引的数据导入和索引建立过程均非常耗时，以 PostgreSQL 为例，其 R-Tree 索引后的数据表大小竟是原始数据大小的近 12 倍。有待提高的性能很大程度上制约了 R-Tree 索引方案在实际中的应用，这也是为什么至今大多数商用数据库系统并不支持基于 R-Tree 的并发处理的原因。

虽然 R-Tree 并不成熟，但它已经是空间多维索引方法中应用最为广泛的一种了，而其它各种方法在现今的数据库系统中更是少有支持。实现全球统一、可互操作的天文数据访问服务一定是最终的目标，所以只有具有广泛通用性的索引方案才会被采纳。因此，无论从适用性、效率还是通用性来考虑，现成的空间多维索引方法都不是最优的解决办法。下一小节中将着重介绍伪二维球面索引方法，它既可以实现对赤经、赤纬二维信息的联合索引，又可以直接借助数据库技术中最为成熟的 B-Tree 一维索引方法，从而既保证了较好的综合性能又保证了良好的通用性和可扩展性，现已被广泛地应用于天文数据处理中。

2.4.5 伪二维球面索引方法

所谓的伪二维球面索引就是通过某种几何球面划分方式将全天区分成许多小区域，这样每一个天体根据它的赤经、赤纬坐标就可以落入某一个小块区域内，从而在天体到小块之间建立起了一种多对一的映射关系。因为每一个小块都会对应一个唯一的整数型 ID 编码，这样就实现了赤经、赤纬二维空间到一维空间的映射，实现了赤经、赤纬的合二为一，从而可以直接应用所有数据库管理系统都支持的 B-Tree 索引技术完成索引的构建。

目前应用最广泛的两大伪空间索引方法是 HTM（Hierarchical Triangular Mesh）[86] [87]和 HEALPix（Hierarchical Equal Area isoLatitude Pixelisation）[88][89]，全球各大天文数据中心几乎都选择了这两种中的一种来构建自己的数据服务。而我国在诸如虚拟天文台、天文科学数据主题库的一些项目中，也需要在它们二者之间进行选择已决定如何构建数据访问服务。但当前，关于这两种索引方式的对比讨论还比较少，本文在后续讨论交叉证认算法和实验的章节中，将分别给出基于这两种索引方式的交叉证认算法，并通过理论和实验两个角度对它们针对交叉证认这一特定应用的性能给出对比和分析，以为我国正在开发的虚拟天文台、天文科学数据主题库等重要项目的数据访问服务的方案决策提供参考。下面就 HTM 和 HEALPix 的基本概念、编码方式、应用背景等方面进行简要的介绍。

HTM 由约翰·霍普金斯大学提出，最早服务于美国斯隆巡天数据，从诞生之日起，它的目的就是对球面上的点源进行分割。它的划分方式可以描述如下，见图 2-10，最开始它将全天区划分为 8 等份，南北各四个球面直角三角形，后续每一级的划分则以上一级划分所得的各个三角形为目标，取其每一边的中点为新的顶点从而将它划分为基本相等的四个新的子三角形区域。通过这种四叉树式的层次递归的划分方式，对应第 n 级的划分，全天区共包含 8×4^n 个索引小块。

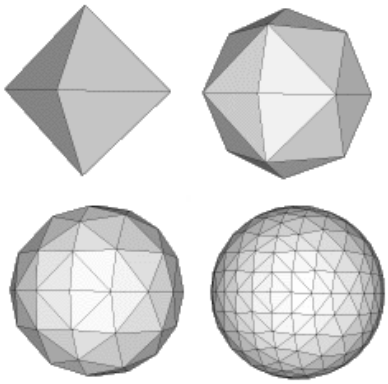


图 2-10 HTM 球面索引方法

HEALPix 索引的划分思想与 HTM 非常相似，同样具有四叉树式的层次划分规律，只是在几何拓扑结构上不以三角形为构成图形，而选择了四边形，如图 2-11 所示。HEALPix 在起源和发展上却与 HTM 非常不同，它最初的目的是为了对球面数据的快速数值分析提供支持，而并非针对于点源的。相比 HTM，它具有一个非常好的特性——等面积性，这使它更具强大的分析功能，如支持全局核函数和局部核函数的卷积运算、球面调和函数的傅里叶分析、功率谱测定和拓扑分析等，在诸如宇宙微波背景研究的数值分析中已有广泛的应用。近年来，它对海量数据的存储、访问方面的特性也逐渐被重视起来，一些虚拟天文台、数据中

心也开始选择 HEALPix 作为构建天文数据访问服务的基础，从这点功能上看，它与 HTM 又非常相似。

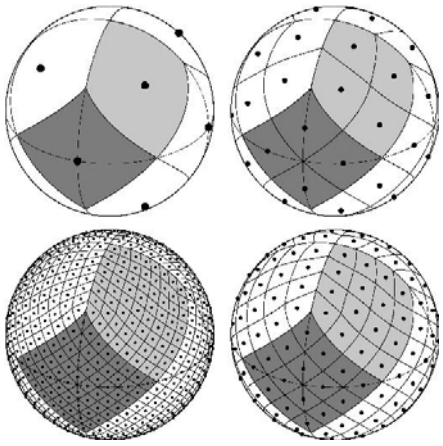


图 2-11 HEALPix 球面索引方法

HEALPix 和 HTM 在划分方式上非常相似，都遵循着逐层递归的方式，在首级划分（第 0 级）上也都将整个球面分成了几个基本区域，分别是 12 个和 8 个。在几何结构上，HEALPix 以四边形的拓扑结构逐层递归地对 12 个基本区域进行划分，在每个划分级别上，所有的四边形小区域面积完全相等；相比之下，HTM 则以三角形的几何结构进行球面切割，同样，它也具备逐层递归的划分形式，但每个划分级别上的各个小区域的面积只相互相近并不完全相等。通过这样的划分方式，对于第 n 级的索引划分，全天区在 HEALPix 和 HTM 下分别被分成了 12×4^n 和 8×4^n 块索引小块。从编码方式上来看，无论是 HTM 还是 HEALPix 都具有一种层次递归的方式[90] [91]，即父节点的编码在进行下一级划分时，被它的四个子节点继承为其编码的公共前缀，而在此前缀后只需额外添加两位二进制数即可区分它们在此级划分中各自所属的位置，如图 2-12、2-13 所示。HEALPix 在具有这种层次递归式编码方式的同时，也支持等纬度的环形方式以适用于数值分析方面的应用，如图 2-14 所示。

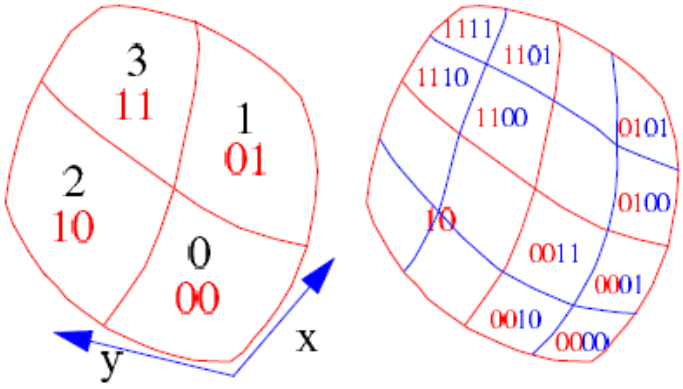


图 2-12 HEALPix 的层次递归编码方式

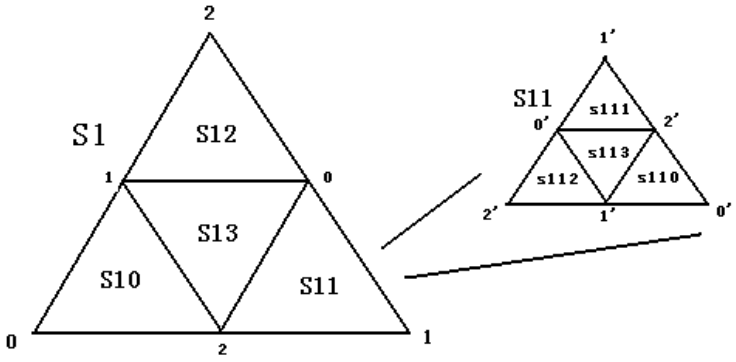


图 2-13 HTM 的层次递归编码方式

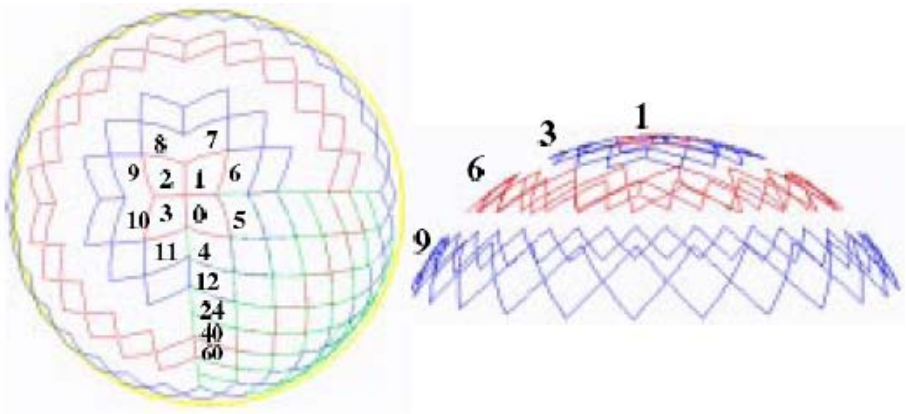


图 2-14 HEALPix 的环形编码方式

HTM 从诞生之日起，目的就是对海量天文数据的存储和搜索，它最初在斯隆巡天项目（SDSS）中的成功应用已成为后续很多天文数据管理方面的典范。比如欧洲空间局正在计划的天体测量卫星 GAIA[92]，它的数据处理方案就选择了 HTM。GAIA 以探索银河系的起源和形成为主要目的，计划 2012 年发射升空，将停留在离地球 150 万公里的第二拉格朗日点上。GAIA 卫星计划对银河系内星等亮于等于 20 星等的约十三亿个天体进行观测，在 5 年内对每个对象观测大约

70 次，其观测涵盖了角距测量、多波段测光和光谱观测，将得到微角秒精度的包括三维空间结构、三维速度场在内的六维银河结构资料，其产生的原始数据量将达到 10TB 之多。预计将发现数十万的新天体，如太阳系外行星和褐矮星等。GAIA 当前之所以选用 HTM 天区划分方式，主要是考虑到可以增加与其他同样使用了 HTM 进行索引的星表（如斯隆数据）的交互能力。而对于后续的全天区的统计分析工作，GAIA 也在考虑使用具有等面积优良属性的 HEALPix。当今天文界逐渐开始认识到选择 HEALPix 索引方式管理数据可能会带来更好的性能特性，所以越来越多的航天项目开始使用 HEALPix，而伴随着 HEALPix 在数据存储访问领域的应用越来越广泛，它所带来的交互性也必将越来越好。

2009 年 5 月 14 日于法属圭亚那库鲁航天中心发射升空的宇宙辐射探测器普朗克（Planck）在数据处理方面就使用了 HEALPix。Planck 是一个国际合作项目，由欧洲空间局（ESA）负责，有 15 个国家参与其中，其中也包括中国。它的目标是以空前的精度获取大爆炸的回声，进而提供幼年宇宙的清晰影响。普朗克将在不同的频率段观测产生 9 个完全的天区影射，初级原始数据约在 0.5TB 到 1TB 左右。早在普朗克发射升空之前，HEALPix 就已经被用于它的数据模拟了，而普朗克科学小组也将 HEALPix 制定为 Planck 数据地图的标准索引划分方式[93]。

本文经过对各种索引方式的综合分析后，最终选择了 HEALPix、HTM 这两种伪二维球面索引方法，它们在交叉证认研究中不仅起到了提高数据查询速度的作用，也是并行程序设计中实现数据划分的基本方法。归纳起来，使用伪空间索引方式的好处包括：

1) 利用它们的层次递归的划分方式和编码方式，只需进行一次高精度的索引编码就等于完成了多次各个精度级别的编码，这是因为，只需忽略掉此高精度编码末尾的若干位就可以得到其所属的父节点上的编码。HEALPix 和 HTM 的这种优点便于程序中根据不同的交叉证认对象星表所具有的不同误差半径来选取合适的索引粒度（即数据划分粒度），从而保证既可以满足证认精度的要求，又可以过滤掉绝大多数不再证认范围内的天体数据，以免过多地引入额外的计算量。

2) HEALPix 和 HTM 索引方式均可以保证划分形成的数据块大小相等或基本相等，这有利于并行程序设计中各节点间负载平衡的实现，有利于保证较好的并行加速效果，同时基本相等的数据块也使根据星表精度进行数据划分粒度选择的步骤变得比较容易。

3) 利用 HEALPix 和 HTM 两种编码方式的规则，可以实现高效的邻域编码计算算法，从而使块与块之间的关系可以在程序运行中及时地快速建立起来，避免了对预处理中加入额外数据行的方法的依赖，以保证不会对原始数据造成污

染。文中第六章将详细给出具体的编码计算算法。

4) 经过这样划分编码后，二维空间成功地映射到了一维线性空间，避免了数据库中尚不成熟的多维空间索引技术，这种伪二维空间索引方式的通用性很好地保证了本文所设计的交叉证认算法在不同星表数据集上的通用性。

5) 逐层递归的划分方式和编码方式很大程度上保持了原物理空间位置上的距离关系：空间位置上相近的点通常在编码上也具有相近的值。依赖这种编码方式，再利用数据库中最常用的 B-Tree 索引，最终使空间上相近的点其在磁盘上的存储位置仍可保持连续或相近，从而最大限度地保证了读取效率。

第三章 多核环境下并行交叉证认

近年来，随着多核技术的迅速发展、成熟，以及它在各方面所展现出来的无可替代的优势，使得当今的处理器芯片已经步入了多核时代。多核计算机毫无疑问已成为业界的主流，同时也将继续成为未来几年的必然发展趋势。这种在过去只被应用于超级计算中心的技术，现在已完全普及到千家万户的个人电脑中。对于天文界来说，当今各大天文中心、研究所所拥有的各种服务器设备也几乎无一例外的采用了多核体系结构，可以说，多核计算资源的普及所带来的强大的计算能力为天文学中很多大规模计算难题的解决提供了新的途径。然而，虽然多核计算机在当下已经唾手可得，但它的性能优势尚无法直接体现，必须要依赖设计合理的并行程序才能得以发挥[94] [95]。本章正是基于这种考虑，研究如何应用并行计算技术在多核单机环境下实现高效的交叉证认方法，并同时解决交叉证认中的常见问题：边界漏源问题，达到效率和精度上的双重提升。并期望借助于对本章中具体并行算法的实验测试和性能分析寻找到交叉证认计算中性能提升的关键因素，为后续更加全面彻底地突破这一海量数据处理难题打下基础。

本章 3.1 节重点解决了并行实现中的数据划分问题，在分析了交叉证认中划分应遵循的原则之后，提出了基于 HEALPix 的划分方式；3.2 节阐述了在该划分下边界漏源问题的解决办法；3.3 节研究并设计了具体的并行交叉证认方法；3.4 节论证了本章交叉证认方法在 HTM 索引下的适应性，并给出了具体的算法设计；3.5 节通过实验的方式验证了本章方法的有效性，并通过对 HEALPix 和 HTM 两种索引方式下的程序运行情况的测试，对比分析了它们在交叉证认这一应用中的具体性能，为虚拟天文台、天文数据主题库等项目提供了技术参考。

3.1 数据划分

第一章已经给出了基于位置信息的交叉证认的具体数学方法。整个证认计算过程看似非常简单，但若采用暴力证认算法，即对星表 A 中的每条记录计算其与星表 B 中的每条记录的球面角距离，其计算复杂度将高达 $O(n^2)$，对于急速增长的海量观察数据不可容忍，将严重影响后续天文数据的处理效率和应用价值。经过分析，发现只有距离小于一定程度的两个记录才可能匹配成功，因此并非每两条数据之间都有证认的必要，他们之中大多数的距离已经远远超过了公式 1-1

中的限定。所以为了提高效率，首先应去除大量无意义的计算量，减少计算复杂度。一种简单的思想即通过画框来限定需要证认的范围，高丹在其论文[29]中指出了证认时赤经赤纬应遵循的计算范围，如公式 3-1 和公式 3-2 所示。

$$\left|RA_A - RA_B\right| < \frac{\left|r_1\right| + \left|r_2\right|}{\cos((DEC_A + DEC_B)/2)} \qquad (3\text{-}1)$$

$$\left|DEC_A - DEC_B\right| < \left|r_1\right| + \left|r_2\right| \qquad (3\text{-}2)$$

由公式 3-1 可以看出，在确定赤纬的证认范围时，其计算包含了一个耗时的三角余弦函数，这是球面坐标的特性所决定的。球面坐标下，从赤道向两极的过程中，即赤纬绝对值递增的过程中，赤经逐渐发生扭曲变形，导致相同赤经数值跨度在不同纬度上的实际球面距离并不相等，且越接近两极，变形越明显。由于这种特性的存在，为使证认范围足以包含误差引起的浮动范围，必须要对此范围在赤经一维坐标上进行一次以赤纬为参数的余弦修正。如果预先对星表 A 中的每一条记录都划定证认范围，即按照公式 3-1、公式 3-2 来判断哪些数据在这个范围内，哪些数据不在这个范围内，则计算量可想而知，必然会导致整体效率的低下。所以，本文采取了天区划分的方式，以较大面积的区域划分代替对每一条记录的画框操作，然后令两星表中同属于一个划分块的数据进行两两间距离计算，这样只要划分可以保证绝大多数的证认都只需发生在同一块的块内部，再通过对少量边界数据的一些特殊处理就可以成功地过滤掉大多无意义的距离计算工作。

3.1.1 简单网格天区划分方式

一种最为简单的天区划分方式如图 3-1 所示，它对整个球面天区在赤经赤纬二维坐标上进行数值等间距切分，从而将全天区划分成很多球面四边形小格子。本文将这种简单的划分方法称为简单网格天区划分方式（Simple Grid Partitioning Function, SGPF）。采用这种划分方式，星表中的每条记录都可以根据它的赤经赤纬坐标上的数值而直接映射到它所属的格子内，而格子的编码方式也较为容易设计和计算。但是，通过下面的分析可以看出，简单网格天区划分方式对于交叉证认这一应用来说并不理想，它在很多方面并不能满足并行交叉证认计算的需要。

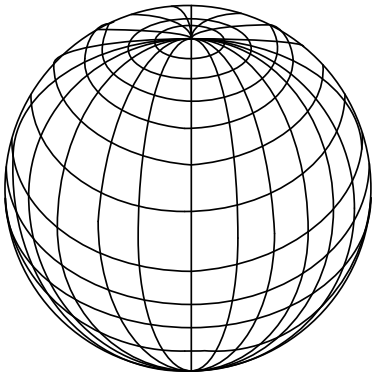


图 3-1 简单网格天区划分方式

数据划分的意义不仅在于实现了计算复杂度的降低，而且实现并行化的一个必备条件。按照 PCAM 并行程序设计模型，划分（Partitioning）是并行程序设计的第一步，并且它与后续的通信（Communication）、组合（Agglomeration）、映射（Mapping）阶段息息相关，划分方法是否合理对最终能否达到优秀的并行加速效果至关重要。

综合考虑数据划分在并行交叉证认实现中各方面的作用，本文认为针对多波段交叉证认这一特定的应用，一个设计良好的天区划分方式应该遵循以下原则：

1) 应该保证使绝大多数的交叉证认计算只发生在独立的块内，跨块的证认应尽可能避免，从而减少程序运行时的进程间通信量（针对共享存储环境，则是减少重叠数据查询量），从而增加程序的并行加速比和可扩展性；

2) 对任意指定块的块内数据应容易定位、提取，并应避免引入过于耗时的额外计算；

3) 划分小块的大小应基本相同，从而可以选择一个较为合适的划分粒度来满足公式 3-1、公式 3-2 中的证认范围，在保证不漏源的情况下尽量减少超出范围的无意义计算量。另一方面，在任务调度设计中，大小相当的数据划分有利于负载平衡的实现，提高并行加速效果。

4) 在此天区划分和块编码方式下，应存在快速、方便的邻接块编码推导算法，保证因解决边界漏源问题而需频繁执行的邻接块编码计算操作可以很高效地完成。边界漏源问题的描述和解决将在下节中给出。

5) 在块的编码方式上，位置相近的块其编码在数值上也应接近，这样通过对块编码建立 B-Tree 索引后就可以保证局部区域数据的编码及其在数据库中的存储都较为连续，从而获得较高的查询效率。

按照上面的几条原则可以很容易地分析出简单网格天区划分方式的不足之处。首先，如果希望 SGPF 方法满足原则 3，就需要引入复杂的三角函数计算来对块的赤经尺度进行修正，否则不同赤纬的小块间就会存在很大的面积差异，而

三角函数修正计算的引入又会违背原则 2 中的要求：保证在划分的实现中要尽量避免复杂的额外计算量的引入。此外，SGPF 方法的块编码方式很难保证原则 5 的要求，比如如果采用图 3-2 中的环形编码方式，则虽然原则 4 的要求——快速高效地推导邻接块的编码很容易实现，但却无法保证原则 5 中对“位置相近的块其编码在数值上也较为接近”的要求，从而导致在进行局部某区域的查询时无法充分利用 B-Tree 索引所擅长的高效区间查询所带来的优势性能。

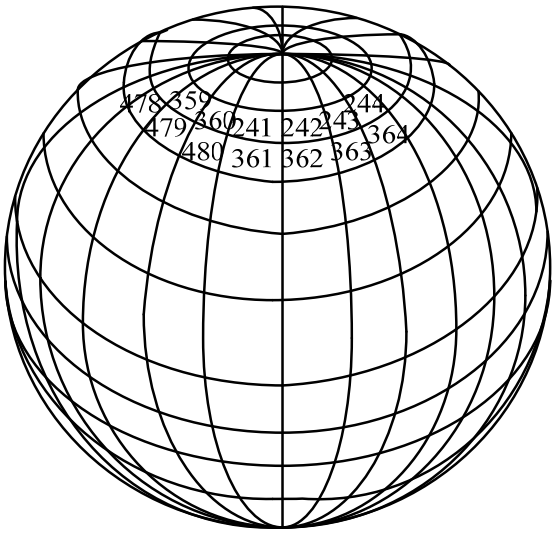


图 3-2 简单网格天区划分中的环形编码方式

因此，简单网格天区划分方式并不能有效地提高交叉证认的计算效率，根据上面总结的划分原则，提出了基于 HEALPix 球面索引算法的数据划分方式。分析发现，这种划分方式既可以有效地过滤无用的计算量又可以达到优秀的并行加速效果。

3.1.2 基于HEALPix索引的天区划分方式

正如 2.4 节所述，索引的建立对海量数据处理的效率至关重要，HEALPix 和 HTM 正是两种应用最广泛、性能最优越的天文数据索引方法。它们二者均通过球面划分的方式来实现二维球面空间到一维线性空间的映射，再利用关系数据库中当前最为成熟的 B-Tree 索引技术完成索引。HEALPix 和 HTM 两种索引方式不仅提高了数据库查询的效率，它们对全天区数据的划分也是实现并行化的关键一步。

HEALPix 划分所具有的等面积特性满足了前面原则 3 中的要求，HEALPix 工具包在为每条数据生成对应分块编码时同样包含了对赤经坐标的修正，但因为

这种包含了修正工作的编码生成只需在数据到来之时执行一次，其后无论进行大规模全天区证认还是实时性的小范围查询证认都无需重复编码，所以这一过程可被视为数据的预处理阶段，并不会给证认计算带来额外的计算耗时，正好切合了原则 2 中的要求。

继续分析 HEALPix 的编码方式发现它同样符合原则 1 和原则 5。正如本文 2.4 节中对 HEALPix 的介绍，它具有一种层级嵌套的划分方式和编码方式，即属于同一父块的所有子块具有相同的编码前缀，该编码前缀即为它们的公共父块的编码。此性质对于交叉证认中数据划分阶段而言有两个好处：

1) 执行一次高层级的 HEALPix 划分、编码工作等于就完成了低于该级别的各个级别的划分、编码，当需要选取一个较粗的划分粒度时，只需忽略掉表中各个 HEALPix 编码的末尾几位即可。因此，采用 HEALPix 索引方式，对所有星表数据库只需执行一次编码计算操作。该性质的另一个好处是更好地切合了原则 1 中的要求。这样，根据目标星表的星体密度的不同，就可以在任务分配时选择最为合理的数据划分粒度，以保证在该划分粒度下绝大多数的证认计算只发生在块的内部，从而确保较低的进程间通信量或重复数据查询量。

2) HEALPix 的层级嵌套划分方式和编码方式正好保证了原则 5 中强调的“位置相近的块其编码在数值上也较为接近”的要求。这样，每个区间的索引小块就会具有较多的连续编码，B-Tree 索引在区间查询上具备的优势保证了这种编码方式下区域查询的高效性。参照上一点的分析，按照星表密度的不同，往往会选择不同的划分粒度，其实质就是对原子索引小块进行区域合并，因此区域查询对整个交叉证认计算至关重要。

由此可见，HEALPix 划分方式很好的切合了原则 1、2、3、5 中的要求，而原则 4 所强调的“存在快速、方便的邻接块编码推导方式”，HEALPix 同样可以满足，本文在第五章设计了基于位运算的高效邻域编码计算算法。

在对星表数据建立 HEALPix 索引编码时，为了不破坏原来星表数据库结构，同时也为了异地星表间执行证认操作时可以较少地传输数据，所以该 HEALPix 编码列并非直接添加在原始数据表中，而是首先提取了与交叉证认计算相关的“天体 ID”、“赤经”、“赤纬”三列组成了该星表的证认数据表，再在该表中添加第四列“HEALPix_ID”。以 SDSS 的数据为例，表 3-1 给出了经过 HEALPix 编码标记后的 SDSS 证认数据表示例，此时 HEALPix 划分级别设为 13，即全天区被划分为12×4^{13}块索引小块。经过 HEALPix 索引标注之后，表中的天体记录与 HEALPix 编码之间形成了多对一的映射关系，这时只需利用数据库管理系统中的 B-Tree 索引对 HEALPIX_ID 字段建立索引即可大幅加快按照 HEALPIX 编码标示的指定区域的数据库查询速度。

表 3-1 HEALPix 编码后的 SDSS 证认数据表示例

OBJ_ID	RA	DEC	HEALPIX_ID
587722951693303820	236.22351359142900	-0.99498276718255996	512738094
587722951693303821	236.21882531110799	-1.00325496854867002	512738091
587722951693303822	236.22462092629101	-0.99496978765010202	512738095
587722951693303823	236.22060204651899	-0.99460964413498598	512738180
587722951693303824	236.22260639274199	-1.00537610660075005	512738089

3.2 边界漏源问题的解决方法

本节在解决边界漏源问题时，有两个重要概念：HEALPix 原子块和计算块。所谓 HEALPix 原子块就是通过 HEALPix 天区划分方式划分后形成的最小单元，即该划分形成的多层四叉树的所有叶子节点。计算块是距离计算的基本单元，位于同一计算块内的星表 A 和星表 B 中的数据两两间才发生距离计算，同时计算块也是并行程序中任务分配的基本单元，它由 HEALPix 原子块组成，并且正好是一些原子块的上层父亲节点块，因此它总是包含 4^m 个 HEALPix 原子块，而它的编码也正好是该 4^m 个 HEALPix 原子块的公共编码前缀。之所以在任务划分、调度、分配时不以 HEALPix 原子块为基本单元，而是选取较大尺度的数据块，是基于以下几方面的考虑：

1) 过细的计算块会给任务分配进程带来很大的压力，过于频繁的任务分配和新任务请求将会成为程序处理的性能瓶颈；

2) 过细的计算块在数据库查询时会造成一种灾难，数据库的 I/O 操作本身已是交叉证认问题的主要性能瓶颈，通过实验测定发现当划分深度超过 9 时，对所有小块的数据查询时间将成倍增长，以致导致整体计算性能非常低效。

3) 计算块划分的粒度越细，则边缘输入所占的比例将会越高，从而带来了更多的边缘块处理代价，而边缘块处理的最大耗时同样在数据库查询上。这一点在本节后续的边界漏源问题解决方法中做进一步阐述。

由上面的三点分析可以看出，减少计算块分块粒度，即增大每个计算块的大小，实质上就是以增加距离计算量的代价换取较小的通信耗时、数据库查询耗时。而究竟计算块的粒度选取何值时整体的交叉证认效率可以达到最优，将通过后面的实验加以测定。

只对处于同一计算块内的数据进行距离计算以推断它们是否属于同一个天体会导致大量的漏源现象的发生。如图 3-3 所示，因为不同的望远镜在观测时会

存在不同程度的位置误差，所以星表 A 中落于某 HEALPix 块边缘的一个天体，其在星表 B 中的天体很有可能超出了同位置的这个 HEALPix 块边界，而落于了它的邻接块内。因此，如果严格地对天区进行分块，且将距离计算限制在块内部，则不可避免地会导致一些处于块边缘的天体无法找到对应体，或者找到距离较远的错误的对应体。这种现象通常被称为边界漏源问题。

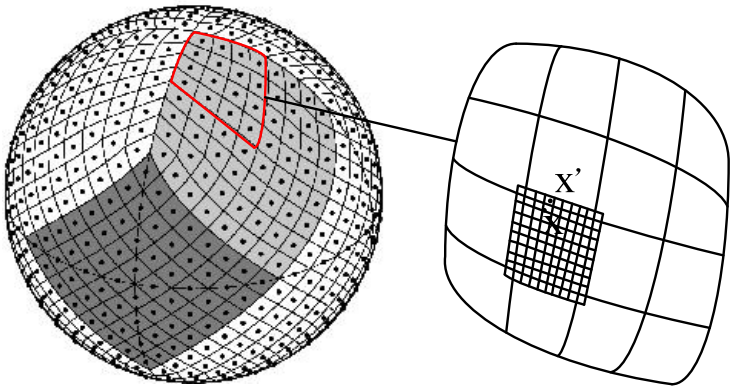


图 3-3 边界漏源问题

在通过划分的方式解决交叉证认问题时，如果不对边缘数据进行额外的处理，则边界漏源现象无可避免。为了简化程序，提高效率，很多交叉证认处理方法都忽略了这一问题，虽然获得了较好的效率但却造成了一定程度的精度损失。高丹在其基于 HTM 索引和 Kd-tree 的交叉证认方法中，也没有对边缘数据进行特殊处理。本文应用一定策略在保证计算效率的基础上解决了边界漏源问题，向天文学家提供更为可靠的结果数据。

为了保证交叉证认的准确性，防止漏源现象的出现，当计算进程从星表 A 中提取出了某一计算块（包含 4^m 个 HEALPix 原子块）的数据后，它不仅需要提取出星表 B 相同块的块内数据，还需要提取出星表 B 中该块周围一圈 HEALPix 原子块的数据，如图 3-3 所示。在本文的实验中，HEALPix 划分深度取值为 13，即全天区总共分为 12×4^{13} 块 HEALPix 原子块，之所以选择这样的划分深度，是因为此时的原子块宽度刚好可以满足精确度较高的 SDSS 数据源和 TwoMASS 数据源的可望证认范围的需要，从而保证在进行较高精度的星表证认时，星表 A 中的某块数据在星表 B 中的全部对应体都会落于包含了一圈邻接 HEALPix 原子块后的较大块范围内，同时该范围又不至于被扩张得过大从而引入过多的计算量。当然，当在精度较低、密度较为稀疏的星表中进行交叉证认计算时，只需在较粗的粒度上构成一圈边缘数据即可，而 HEAPix 天然具有的四叉树层次编码方式可以直接支持各个划分粒度上的小块编码的生成。

经过分析，本文认为为了在证认计算时快速地确定每个计算块周围的边界块数据可以有以下两种方案：

1. 对于星表 B，可以在最初对原始数据进行 HEALPix 编码后，再进行一次计算块块号标注，而此次标注采用打标签的方式，即对每条天体记录不一定仅标记一个它所属的计算块的 ID 号，而是先判断它是否属于某块计算块的边缘块，如果是，则同样标记该块计算块的 ID 号。如图 3-4 所示，处于四条边上的边缘数据一共会有两个计算块块号标记，而处于四角的数据则会有四个标记。事实上，这种所谓的打标签的方法在实现上就是以插入额外的行方式完成的。这样，当计算进程执行到某一个计算块的证认任务时，按照该计算块的块号在星表 B 中查找的数据就包含了该计算块位置周围一圈 HEALPix 原子小块，从而在计算距离时避免了漏源现象的发生。

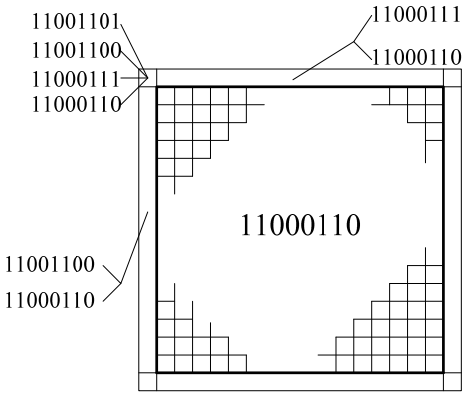


图 3-4 边界漏源问题解决方案之“打标签”方案

2. 不在数据库中增加额外的行，当计算进程执行到某一个计算块的证认任务时，程序快速计算出该块周边一圈 HEALPix 原子小块的 HEALPix 编码，程序根据这些编码查询星表 B，从而提取出边界数据，避免漏源现象。这种方法要求相邻边界块编码计算算法要具有很高的效率，才可避免对整体效率的拖累。

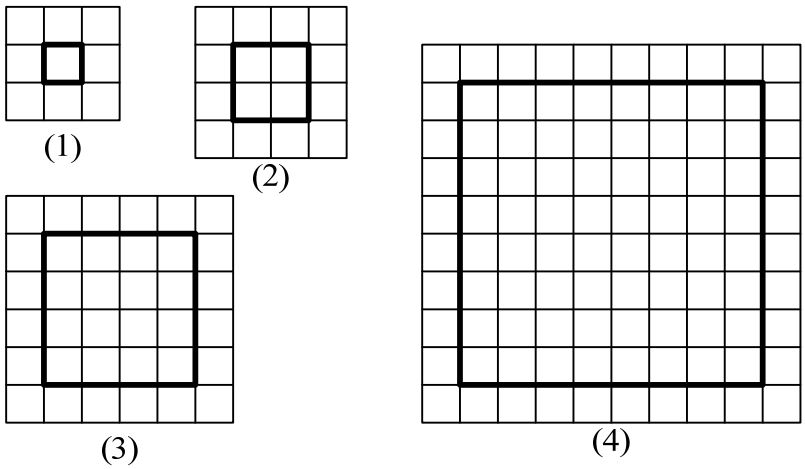
如果采用方法一，则在证认计算过程中避免了对边界数据编码的推算，节省了一定时间，但此方法因为要在数据库中加入额外的行，所以会对原始数据表造成一定程度的污染，致使经过这样标记后的数据表只能为交叉证认计算所用，而且这种边缘数据标记还与计算块划分的深度相关，所以灵活性差，当计算块划分需要改变时，只能重新生成。方法二不用加入重复的数据行，避免了对原始表结构的破坏，而且只需对所有数据执行一次 HEALPix 原子块的编码标记，与证认过程中选择的计算块粒度无关。这样既节省了数据存储空间，也避免了多次重新 HEALPix 标记带来的额外开销。正是基于这些优点的考虑，本章选择方案二来解决边界数据漏源问题。第六章给出了本文设计的基于位运算的快速邻接块

HEALPix 编码推导算法，将会看到，利用这种快速算法，方案二中对效率的忧虑可以完全排除。

3.3 并行程序设计

对交叉证认并行化，是解决海量数据量所带来的效率问题的根本方案。PCAM 并行程序设计模型是并行程序设计中最常采用的设计方法，本章并行程序设计中也遵循了这种思想，以确保并行加速的显著效果。

对于交叉证认这一应用，数据块划分粒度的选择关系到的是 PCAM 模型中的划分和组合两个步骤。考虑到交叉证认属于数据密集型计算任务，所以在划分这一步本方法主要采用域分解的方式，也叫数据划分。上一小节在介绍边缘数据问题时已经给出了计算块的概念，它是本算法中数据划分、任务调度的基本单位。良好的数据划分一般要遵循两个要点：其一，要力图避免数据和计算的复制，即尽量保证数据集和计算集互不相交；其二，如果可能的话，应使这些小的数据块尽可能大致相等。要点一主要是为了保证数据块间的非重叠性，从而减少计算中由于数据间相互关联所造成进程间的通信和额外的其他处理。对于很多并行程序，通信恰是影响并行加速的主要因素，而对于本程序，由于采用了共享存储的并行方式，数据通信量小，主要性能瓶颈来源于数据库的 I/O 读写操作，所以保证数据块间的非重叠性主要目的是为了减少重复的数据库查询。更为具体地说，这种重复数据查询主要源于边界数据问题，而基于“计算块”的数据划分方式刚好可以确保较好的数据非重叠性。从图 3-5 可以看出，计算块选取的越大则边缘数据量占总数据量的比例就越小，而基于 HEALPix 的层次划分方式和编码方式刚好保证了计算块大小选择的灵活性。在下一小节的实验中，也将针对“计算块划分粒度”这个参数进行效率测试，以找到它的最优值。同时，计算块粒度的确定也与 PCAM 模型中的组合一步相关，计算块本身可看作更小的 HEALPix 原子块的组合，图 3-5（1）中的划分粒度就是以 HEALPix 原子块为任务分配单位的情况，过多的边缘数据的比例导致了极差的数据独立性，从而造成了过多的额外处理代价，这也正是 PCAM 中需要进行组合一步的意义所在。



注：（1）计算块大小为4^0个 HEALPix 原子块时，边界数据量与原数据量比例为 8:1；（2）计算块大小为4^1个 HEALPix 原子块时，边界数据量与原数据量比例为 3:1；（3）计算块大小为4^2个 HEALPix 原子块时，边界数据量与原数据量比例为 4:5；（4）计算块大小为4^3个 HEALPix 原子块时，边界数据量与原数据量比例下降为为 9:16；

图 3-5 计算块大小与边界数据所占比例

通信是 PCAM 模型中另一个要考虑的因素。正如前面所述，由于采用了共享内存的编程模式，数据通信量很小，而任务通信所涉及的也只是进程间的少量控制指令，由于每一个计算块任务都可以由它的 HEALPix 编码唯一确定，所以这种控制也非常简洁。因此，通信对于本程序而言并不是最主要的考虑因素。

在数据划分、通信、组合之后， PCAM 设计步骤中的最后一步是映射，即讨论按照何种方法将数据、任务映射到特定的计算资源上面去。在任务调度方面，本章采用了主从式编程模式。所谓主从式编程模式又名播种模式，其思想是将一个待求解的任务分成一个主任务（即主进程）和一些从任务（即从进程）。主进程负责向从进程分配任务，并收集子进程的求解结果。对于本交叉证认算法而言，每个计算块的证认任务都可以以它的 HEALPix 编码唯一标识，所以主进程在向从进程分配任务时只需发送待解决的计算块的 HEALPix 编码即可。

主进程在进行任务分配时，主要需考虑的因素就是负载平衡问题。本方法为保证良好的负载平衡，采用了动态任务分配方法，即主进程向每个子进程每次只分配少量的计算块证认任务量，当子进程完成证认计算后再通知主进程，将证认上的结果传给主进程，并申请新的尚未证认的计算块块号。由于每次任务分配时主进程只需向子进程发送计算块块号，传输量极小，所以采用动态任务分配方式（每次分配几十个连续计算块编码）既可以最大限度地保证负载平衡又可以不带来额外的开销。

本方法的并行程序计算流程最终设计如下：如图 3-6 所示，一个主进程负责全局的任务分配、结果收集、结束控制；多个子进程（根据多核计算机的核数确

定子进程数），也就是计算进程，根据主进程分配的任务进行具体的数据加载、计算、判断工作。3.5 节通过实验对本节设计的并行方法加以验证。

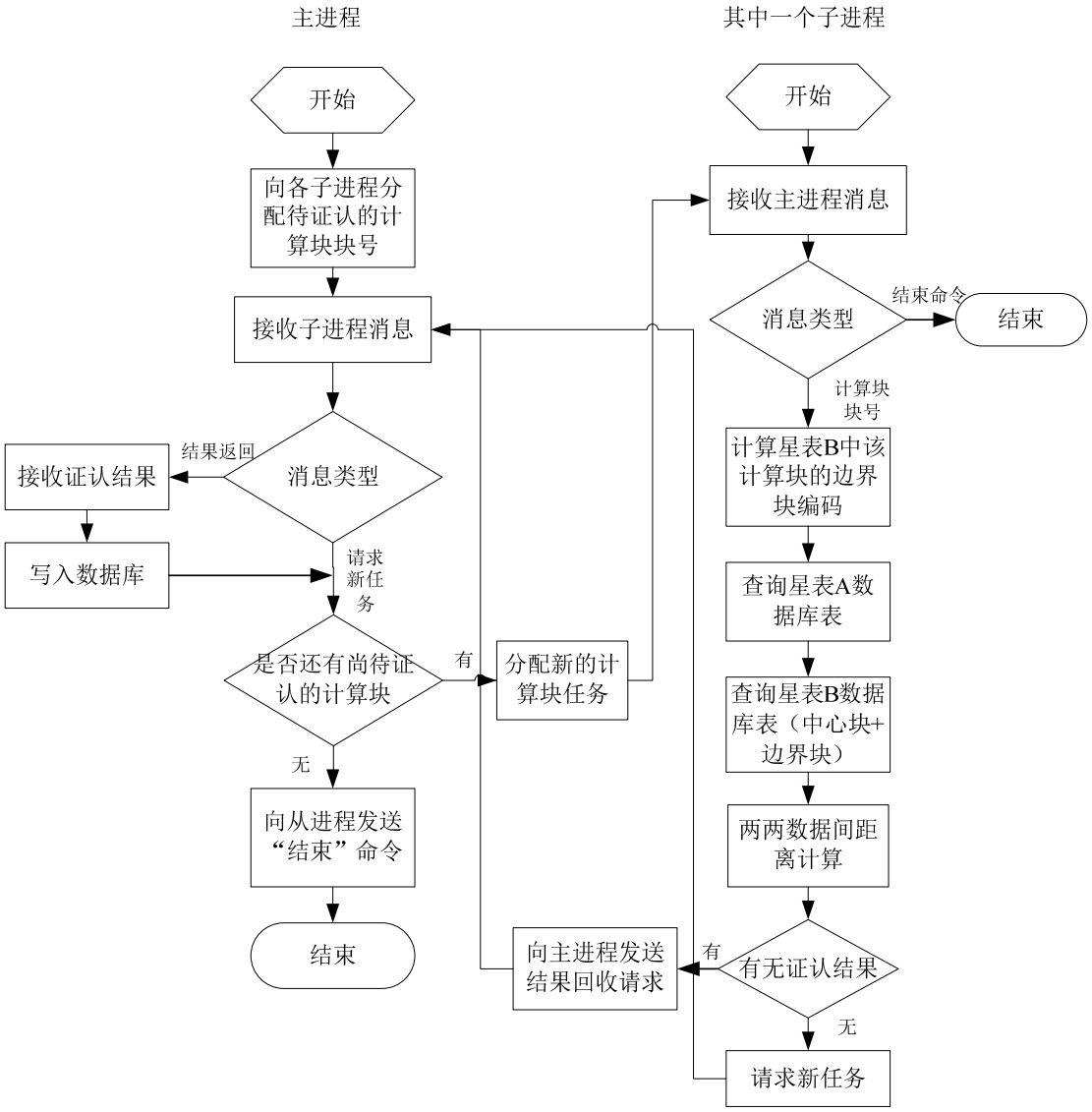


图 3-6 并行程序设计流程图

3.4 面向HTM索引的方法扩展

HTM 是 HEALPix 之外的另一种非常常用的球面索引方法，前面的 2.4 节已经就它的划分原理、编码方式及应用情况给予了简单介绍，本节目的是探究本章所提出的并行交叉证认方法在 HTM 索引下是否同样具有可行性，并从理论和实验两方面对这两种索引方式在交叉证认中的应用性能进行了对比分析。

在天文数据访问服务方面，HTM 球面索引已经有很长的应用历史了，诸如美国斯隆数字巡天项目等很多大型天文项目都以 HTM 作为它们的观测数据的索

引方式。所以，开发基于 HTM 的高效并行交叉证认方法同样有着至关重要的实际意义。甚至，从某种程度上来说，HTM 球面索引在数据访问服务方面较 HEALPix 的应用更为广泛一些，当然，这种广泛性主要源于一种称为“惯性”的因素，而并非出于它们二者性能差别上的考虑。当前，对这两种索引方式在各种天文应用中的性能对比分析还鲜有研究，本节对它们二者在交叉证认这一天文基础数据处理环节的实现和对比正是希望能为我国正在建设中的诸如天文数据主题库、中国虚拟天文台数据访问服务、LAMOST 数据服务等的几大天文数据项目提供一些基础性的技术参考，以便于在其二者之间进行抉择。另一方面，HTM 和 HEALPix 作为两种最为成功的虚拟二维球面索引方法，在它们二者之上进行对本章交叉证认思想的检验有着非同一般的意义，这将加深我们对解决交叉证认效率问题的关键的理解，同时也可以促进我们对本章方法在设计上存在的问题的发现，为今后对该问题的进一步研究和突破打下基础。

因为 HTM 索引在很多方面都与 HEALPix 索引非常类似，比如它同样具有逐层递归的四叉树式划分方式和编码方式，同样是通过将二维空间坐标映射到一维线性空间实现的虚拟二维索引，因此 HEALPix 方法中的很多设计思路同样适用于这里。所以，本节并不完整地阐述 HTM 下的并行交叉证认实现方法，只侧重于分析它们二者之间的异同之处。

3.4.1 基于HTM的数据划分方法

3.1.1 节已给出了适用于并行交叉证认的划分方式应该遵循的五点原则，这是在综合地考虑了划分在并行交叉证认中所起的重要作用后提出的，本节基于 HTM 的划分同样可以以此五点原则为理论上的检验标准。

1) 对于第一点“保证使绝大多数的交叉证认计算只发生在块内”，因为 HTM 同样具有逐级递归地四叉树式划分方式，所以只要选取合适的划分级别既可以保证分块的尺度明显大于误差半径的尺度，从而保证除边界数据外的大多数数据只需与其同块内的数据进行距离计算判断。但在这一点上，HTM 仍与 HEALPix 有所不同，因为它是以三角形为划分形状的，较 HEALPix 的方形更偏离于误差形成的本来形状：圆形。因此，为了使划分后的三角形块满足误差的需要，以保证绝大多数的交叉证认计算只发生在块内，处于边角位置，超出了证认范围的无用距离计算量其实较 HEALPix 更多。

2) 对于第二点“任意指定块的块内数据应该很容易定位、提取，并且应避免引入过于耗时的额外计算”的要求，HTM 索引也同样可以满足。所谓“过于耗时的额外计算”主要就是针对于两级周围存在的赤经扭曲现象进行的三角函数修正，与 HEALPix 类似，HTM 的编码工具包在为每条数据生成对应分块编码时

就已包含了对赤经坐标的修正，而这一较为耗时的步骤因为对所有数据只需执行一次，因此并不会影响证认过程中程序的性能。

3) HTM 在实现划分时因为已经包含了对赤经的修正，虽然不能像 HEALPix 一样保证面积的完全相等，但也具有面积接近的属性，因此原则 3 中的要求可以满足。这一属性的具备，使它既可以选择一个较为合适的划分粒度，使公式 3-1、公式 3-2 中的证认范围可以被满足，又可以保证并行任务分配时较为容易地实现动态负载平衡。

4) 按照第四点原则“存在快速、方便的邻接块编码推导方式”，本文根据 HTM 的编码规则同样设计了该编码下邻接块的编码快速推导算法，而对这一算法的具体阐述见于第五章。

5) HTM 与 HEALPix 一样具有逐级递归的编码方式，因此其同样可以在一定程度上使大多数数据满足原则中的第 5 点“位置相近的块其编码在数值上也较为接近”。但是无论 HEALPix 还是 HTM 在分块的边角处都存在着一定的编码跳跃现象，这是任何二维空间映射到一维空间时都无法避免的问题。在这一问题上，HTM 的三角形拓扑结构使它的边角数据所占比例更高，因此它所面临的编码局部跳跃现象也就更为严重。

由此可见，HTM 同样是一种值得期待的交叉证认划分方式。只是在第 1 点、第 5 点上，HTM 的三角形几何结构使其与 HEALPix 相比显得稍有不足。而这些细节上的差别是否会对最终的程序性能造成影响，将通过后续的对比实验进行进一步的分析。

3.4.2 基于HTM索引的边界漏源问题的解决

很显然，在使用 HTM 进行数据划分时，同样会存在着边界问题，如果不加以额外的处理，就会产生漏源现象。图 3-7 是 HTM 下边界问题的示意图。与 HEALPix 下的解决方法类似，同样对当前正在处理的计算块在星表 B 中要额外地加载该块周围一圈的 HTM 原子小块。从加载方式上来说，同样可以考虑按照两种方式进行设计：其一，对处于边界的数据进行多次 HTM 编码，并将这些重复编码后的额外的行预先加入到原始数据库表中；其二，设计高效的相邻边界小块 HTM 编码的推到算法，在程序的计算过程中及时地调用。经过分析后，我们认为如果采用第一种方案不仅会破坏原始表内数据，而且还会较 HEALPix 占用更大的存储空间。这是因为相比于 HEALPix 的方形块结构，HTM 的三角形块结构导致了更高比例的的边界块。基于这方面的考虑，本节同样通过设计出相邻边界小块的快速编码推算算法来避免这一问题，详细的算法将在第六章中给出。将会看到，因为 HTM 在编码规则上不同于 HEALPix，所以相应的相邻块编码算法

也有所不同，但与之相同的，它同样具有非常高效的性能，同样是本文设计的高效并行交叉证认算法的可操作性的重要保证。

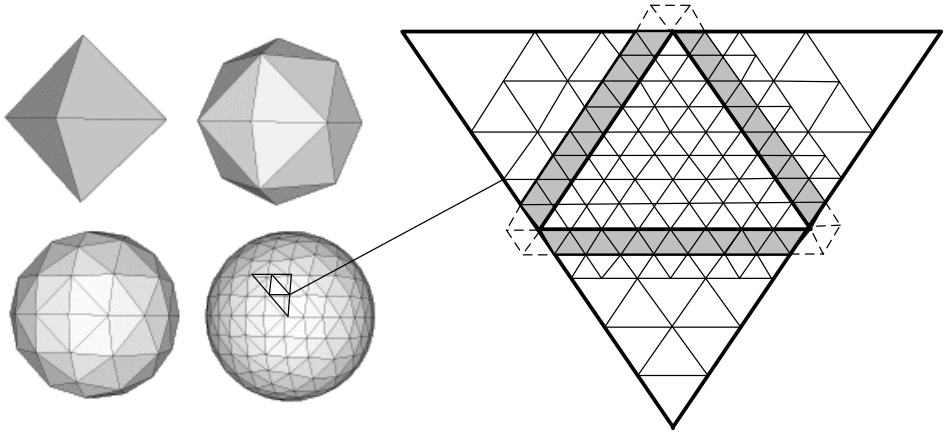


图 3-7 HTM 划分下的边界问题

3.5 实验结果及分析

为了测试本章方法的实际性能，首先在大数据集上进行了基于 HEALPix 索引的并行交叉证认方法的实验，并与高丹[29]提出的交叉证认方法及基于简单网格划分的交叉证认方法进行了全面的对比实验。此后，为了验证本章方法在 HTM 索引方法下的适用情况，继续进行了 HTM 索引下并行交叉证认的实验，通过实验结果的对比，分析了 HEALPix 和 HTM 两种索引方式在交叉证认应用中的优劣。

3.5.1 面向HEALPix索引的并行交叉证认方法实验

实验环境

本节实验采用了四核的 Dell PowerEdge T300 服务器，详细的环境参数如下：

- ✧ 处理器：Quad-core Xeon X3323
- ✧ 内存：2.0G 667MHz DDR2
- ✧ 操作系统：Linux Fedora 10（内核：2.6.23.1-42.fc8 SMP）
- ✧ 交换分区大小：2G
- ✧ 软件环境：C 语言+MPICH2[96]+MySQL

实验数据和参数

本节用于交叉证认实验的两个数据集分别是SDSS星表[1]和TwoMASS星表[2]，它们各自包含的数据量分别约为 1 亿天体记录和 4.7 亿天体记录。实验中在对数据计算HEALPix原子块编码时，选取的划分级数为 13。13 是使用C语言版本HEALPix 工具包的最高级别，在该划分级别下，全天区被划分为 12×4^{13}=805306368 个HEALPix原子块，经计算，这些小块的宽度约为 20 角秒，恰好刚刚大于最理想的边界数据块宽度，即SDSS与TwoMASS的误差半径之和。这样既可以保证证认结果的完全性，又可以避免过多的不必要计算量。表 3-2 给出的是HEALPix在各个划分级别下对应的划分数量和划分原子块大小。

表 3-2 HEALPix 划分级别、划分数量和原子块大小

k	$N_{side}=2^k$	$N_{pix}=12N_{side}^2$	$\theta=\Omega_{pix}^{1/2}$
0	1	12	58°.6
1	2	48	29°.3
2	4	192	14°.7
3	8	768	7°.33
4	16	3072	3°.66
5	32	12,288	1°.83
6	64	49,152	55′.0
7	128	196,608	27′.5
8	256	786,432	13′.7
9	512	3,145,728	6′.87
10	1024	12,582,912	3′.44
11	2048	50,331,648	1′.72
12	4096	201,326,592	51″.5
13	8192	805,306,368	25″.8
14	2^{14}	3.22×10^9	12″.9
15	2^{15}	1.29×10^{10}	6″.44

程序中另一个重要参数是计算块的划分数量，它是任务调度的基本单位。该参数的最优值已在下面的实验中加以测定。。

实验方法：

[1] SDSS：斯隆数字巡天项目，Sloan Digital Sky Survey，http://cas.sdss.org/dr6/en

[2] TwoMASS：二微米数字巡天项目，Two Micron All Sky Survey at IPAC，http://www.ipac.caltech.edu/2mass/

按照前面的设计，具体的交叉证认并行程序在实验中可以分为以下几步：

1) 在 SDSS 星表数据和 2MASS 星表数据中提取出与交叉证认相关的几列，包括天体 ID、赤经值、赤纬值，并根据每条记录的赤经、赤纬利用 HEALPix 工具包计算出该天体在划分级别为 13 时的的 HEALPix 编码，作为第四列写入。

2) 在 MySQL 数据库中建立两个表，分别名为 SDSS_xmatch 和 TwoMASS_xmatch，分别导入前面的四列数据。然后分别在两个表的 HEALPix 编码列上建立 B-tree 索引。

3) 运行按照上节中图 3-6 中设计编写的并行程序，设定主进程数为 1，从进程数为 3。将证认后得到的结果按照一对一、一对多、一对无、无对一分别写入各自的结果表。

实验结果：

Aladin 天文数据软件具有可视化的功能，将证认结果作为输入数据传给 Aladin 即可直观地检查证认方法的正确性。图 3-8 给出了局部一小块天区的证认结果。其中十字标记代表来自 SDSS 星表中的一个天体，圆点标记代表 TwoMASS 星表中的一个天体，较大的圆形框代表以 SDSS 为参照在该点找到了对应天体。从图中可以直观地看到，较大圆框覆盖的位置正好有两个分别来自 SDSS 和 TwoMASS 的天体记录相聚很近。由此可见，本节中的多核环境下的并行交叉证认方法确实可以识别出那些位置相聚很紧密的、具有很大可能性为同一天体的记录。

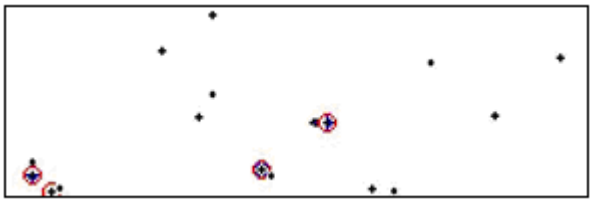


图 3-8 可视化证认结果

相对于正确性的验证，本实验更关心的是程序的执行效率。表 3-3 列出了当计算块划分级别选为 7、8、9 时最终的执行耗时。

从中可以看出，当划分级别选为 8 时，计算效率达到最高，总共耗时为 25 分钟。划分级别取得过低或过高时都会降低证认计算的总体效率，这是因为：当计算块被划分得过细时，边缘数据的比例升高，而边缘块的编码并不连续，所以数据库查询耗时相对明显高于大块的计算块数据的区间查询，这是 B-tree 索引的天然特性直接决定的；另一方面，如果划分得过粗，则需要进行两两距离计算的

计算范围变大，总体距离计算量明显上升，总体效率也会被削弱。

表 3-3 实验结果

计算块划分数量	星表 A 来源	星表 A 数据量	星表 B 来源	星表 B 数据量	运行总耗时
12×4^7	SDSS	100,106,811	2MASS	470,992,970	32 分钟
12×4^8	SDSS	100,106,811	2MASS	470,992,970	25 分钟
12×4^9	SDSS	100,106,811	2MASS	470,992,970	57 分钟

为了验证本节多核并行方法的优劣，实验中将与其他几种方法的效率进行对比。首先用来进行对比的是高丹[29]提出的基于 HTM 索引和 Kd-tree 的交叉证认方法。然后用于对比的方法还有第 3.1.1 节中介绍的简单网格天区划分方法，在该划分方法下又分别采用了按照网格号进行索引的索引方式和按照第 2.4.3 节中提到的赤纬单维 B-tree 索引方式，对这两种方法，这里采用了与本节多核并行程序相同的数据集和相同的软硬件环境，在并行实现方面同样也采取了四进程的 MPI[97]并行程序。表 3-4 给出了这三个对比实验各自的数据来源、数据量和运行时间。

表 3-4 其它对比实验

方法	星表 A 来源	星表 A 数据量	星表 B 来源	星表 B 数据量	运行总耗时
本节多核并行方案（计算块数量：12×4^8）	SDSS	100,106,811	2MASS	470,992,970	25 分钟
简单网格天区划分方法（对赤纬进行索引）	SDSS	100,106,811	2MASS	470,992,970	73 小时
简单网格天区划分方法（对网格编码进行索引）	SDSS	100,106,811	2MASS	470,992,970	78 分钟
高丹方法	Part of GSC 2.3	295,832	Generat from GSC 2.3	295,832	5.8min

由表中可以看出，对比于高丹的方法，本节基于 HEALPix 索引算法的多核并行算法在计算效率上有了很大的提高，并且真正意义上达到了适用于大规模数据量的目的。如果与赤纬单维索引的并行算法相比，在完全相同的环境与数据下，本节的方法的效率提高了 175 倍，这主要得益于数据库查询速度的显著提升。接

下来，当与基于简单网格划分+网格编码索引的算法进行对比时，本节的方法在效率上提高了大约 3 倍。相比前两个对比实验，此时提升效果好像并不非常显著，但如果考虑到 HEALPix 所具有的层次编码特性所带来的“一次编码、各分块粒度上灵活计算”的好处，将会发现本节的方法在小规模实时交叉证认中则具有更为明显的优势。这是因为，交叉证认速度较天体观测数据的下载速度还是有一定差距，对于每天收集的海量观测数据而言，并没有必要两两间进行证认后构建出完整的索引层数据，这样对计算资源存储资源来说都是一种浪费。一种更为可行的方案是当用户提交联合查询或交叉证认指令时，再实时地为之提供指定小区域的交叉证人服务。因为不同星表的精度、密度都有很大差别，所以星表间交叉证认要视证认对象的不同选取不同的划分粒度。而 HEALPix 的层次递归的划分方式和编码方式则可以保证“一次编码，多层次证认”的需求，从而可以把编码、索引这一步作为预处理阶段的一次性任务，更有利于高效的发挥。

3.5.2 面向HTM索引的并行交叉证认方法实验

实验所选用的软硬件环境、测试数据的来源与大小均与前面的 HEALPix 划分下的交叉证认实验完全相同。而数据预处理阶段对原始数据建立 HTM 索引编码时所选择的划分级数为 14，这同样是 32 位存储空间下的最高编码级数。下表列出了 HTM 与 HEALPix 在各个划分级数下的划分参数对比：

表 3-5 HTM、HEALPix 各级划分下参数对比

k	$N_{side}=2^k$	$N_{healpix\_pix}=12N_{side}^2$	$N_{htm\_pix}=8N_{side}^2$	PixArea_healpix	PixArea_htm
10	1,024	12,582,912	8,388,608	$1.18E1$	$1.77E1$
11	2,048	50,331,648	33,554,432	$2.95E0$	$4.43E0$
12	4,096	201,326,592	134,217,728	$7.38E-1$	$1.11E0$
13	8,192	805,306,368	536,870,912	$1.84E-1$	$2.77E-1$
14	16,384	3,221,225,472	2,147,483,648	$4.61E-2$	$6.92E-2$
15	32,768	12,884,901,888	8,589,934,592	$1.15E-2$	$1.73E-2$
16	65,536	51,539,607,552	34,359,738,368	$2.88E-3$	$4.32E-3$

表中已经标出了 HTM、HEALPix 在划分级数分别为 14 和 13 时各自的参数，这是 32 位整形所能容纳的最高划分索引级别。而从划分后原子小块的大小上来看，HTM 取$k=14$时刚好可以满足 SDSS、HEALPix 这样精度较高的星表的误差半径的要求，这样在解决边界数据问题时，只需要按照图 3-7 中示意的那样额

外选取计算块周围一圈宽度为 1 的原子块，则既可以保证漏源现象不会发生，又可以保证不引入过多的无意义计算量。

实验方法

在并行程序实现方面，这里仍然采用了主从模式、动态任务分配的方式，实验的具体步骤包括：

1) 在 SDSS 星表数据和 2MASS 星表数据中提取出与交叉证认相关的几列，包括天体 ID、赤经值、赤纬值，并根据每条记录的赤经、赤纬利用 HTM 工具包计算出该天体在划分级别为 14 时的 HTM 编码，作为第四列写入。
2) 在 MySQL 数据库中建立两个表，分别名为 SDSS_xmatch 和 TwoMASS_xmatch，分别导入前面的四列数据。然后分别在两个表的 HTM 编码列上建立 B-tree 索引。
3) 运行并行交叉证认程序，设定主进程数为 1，从进程数为 3。主进程将所有计算块证认任务按照编码顺序向从进程进行分发，并负责回收各个从进程证认后的结果数据，并将结果按照一对一、一对多、一对无、无对一进行分类，然后写入数据库。从进程则根据分配得到的计算块块号，分别执行 SDSS 数据库和 TwoMASS 数据库的数据查询工作，并根据调用相邻边界原子块的 HTM 编码推导算法所得的编码执行边界块数据的查询，然后执行两两间的距离计算、判断，以确定最终的匹配结果对。每完成一次分配的任务后，从进程向主进程发送数据并申请新的证认任务。

实验结果及对比分析

表 3-6 实验结果

计算块划分数量	星表 A 来源	星表 A 数据量	星表 B 来源	星表 B 数据量	运行总耗时
8×4^7	SDSS	100,106,811	2MASS	470,992,970	48 分钟
8×4^8	SDSS	100,106,811	2MASS	470,992,970	40 分钟
8×4^9	SDSS	100,106,811	2MASS	470,992,970	67 分钟

从表 3-6 中的实验结果可以看出，当计算块的划分数为8×4^8时，总体证认耗时达到最优：40 分钟。而当划分偏离这一粒度时，耗时都会上升，这同样是距离计算量与数据库查询量相互权衡的的结果。具体来说就是随着计算块划分数量的增加，数据库查询耗时呈上升趋势，这是 B-Tree 索引的原理造成的；但与此同时，证认范围因被限制得更小，所以总体距离计算量相应降低。而程序之所

以在划分数为8×4^8时达到最优，正是因为此时计算耗时与 I/O 查询耗时达到了一种平衡，也就是说当计算块数量继续上升时，I/O 查询将成为阻碍程序性能的最重要原因，而当计算块数量继续减少时，整体计算量又会付出过高的代价。

将此实验结果与表 3-4 中列出的几种其他交叉证认方法来看，此方法还是具有非常明显的优势的，它同样使海量数据下的大规模交叉证认具有了实际应用价值。因此可以说，本文所设计的多核环境下的并行交叉证认的处理策略在 HEALPix 和 HTM 两种划分、索引方式下均有很好的适用性。

但如果与 HEALPix 索引下的方法进行对比，可以看出，最优情况下的运行总时长还是较为明显地多于 HEALPix 版本程序下的 25 分钟的。这种性能的差异主要来源于 HTM 与 HEALPix 在几何结构上的差别。形象地来说，HEALPix 的方形划分结构较 HTM 的三角形划分结构更接近于圆形，而圆形才是点源观测误差形成的天然形状，因此，以 HEALPix 为划分、索引方式时，所引入的不必要距离计算量相对较少，而以 HTM 为划分、索引方式时，在其较为尖锐的三个顶角区域内则引入了较多的无用计算。另一方面，在边界数据的处理上，HTM 的索引方式会导致更多的边界数据比例。从理论上进行分析，很显然相同面积下正三角形的周长大于正方形的边长。如果从数值上进行精确的推断，更会发现 HTM 和 HEALPix 划分方式下，包含同样数量原子小块的计算块的周围一圈原子块的数量具有 1.5:1 的关系。

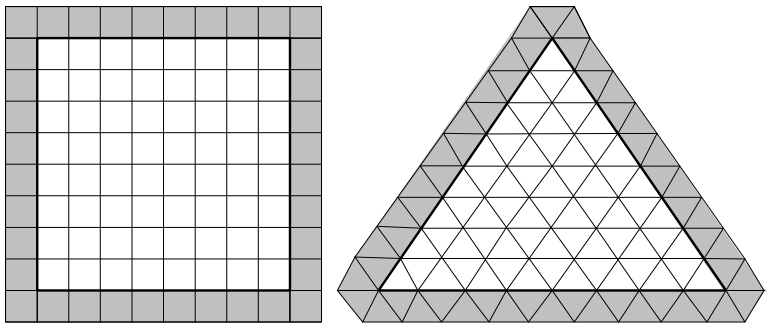


图 3-9 边界原子块比例推算

如图 3-9 所示，假设 HEALPix 和 HTM 里的计算块的边长都为2^k，则它们均包含4^k个原子块。则图中左边的 HEALPix 计算块周边一圈的原子块数量为$4\times(2^k+1)$；而图中右边的 HTM 计算块周边一圈的原子块数量为$3\times[(2^k+1)\times2]=6\times(2^k+1)$。所以，在相同划分级别下，以 HTM 为划分方式时所需要处理的边界数据量占全部数据量的比例是 HEALPix 为划分方式时的 1.5 倍。而根据前面的分析，数据库 I/O 访问耗时是交叉证认中的主要性能瓶颈，对于本文初始版本的交叉证认算法，边界块的数据库查询更是占用了大量的时间，

因此 HTM 索引下 1.5 倍的边界块查询量对整个程序的性能影响非常明显，这也就 HTM 版本程序较 HEALPix 版本程序效率略低的另一个原因。

3.6 本章小结

本章在分析了天文交叉证认并行化实现基本需要及任务划分应遵循的原则之后，提出了基于 HEALPix 索引方式的多核环境下的交叉证认算法。该算法在设计上参照了 PCAM 并行程序设计模型，通过保证其在划分、通信、组合和映射各阶段的良好性能以获得最终成功的并行加速效果。对于交叉证认中常见的边界漏源问题，本文提出了两种解决方案，在分析了二者的优缺点之后，最终选择了及时推算邻接块编码的方式，它不会引入额外的重复行，避免了对原数据表的污染。实验证明，本章方法相比前人的方法在效率上提升明显，使海量数据上的大规模交叉证认得以真正实现。为了使本章的并行方法更具通用性，继续设计了 HTM 索引下的交叉证认方法，它是当今天文界另一个广为应用的数据索引方式。实验显示本方案同样适用于 HTM 索引，只是效率略低于 HEALPix 索引下的程序，分析认为，这种差异主要源于 HEALPix 与 HTM 在几何划分结构上存在的天然差别。这一结论的得出对我国当前正在建设的虚拟天文台、天文数据主题库等项目的数据联合查询服务的建设有重要的参考意义。

考虑到交叉证认是天文多波段数据融合的基础，也是多种数据处理分析操作的必要前提，本节研究实现的并行方法具有较好的应用前景，同时对后续的进一步研究有着重要的参考价值。

第四章 基于限制生长模型的改进并行交叉证认

一般认为，交叉证认的效率瓶颈主要源于频繁的数据库读写操作，本章通过对上章程序中各部分耗时的分析进一步印证了这种观点。因此，优化数据 I/O 操作性能是继续攻破交叉证认的效率问题的关键所在。本章通过对上一章方法在实验中所体现出来的不足之处进行深入分析，提出了新的改进方法，通过重新设计并行程序中的数据加载计算流程及任务分配调度单元，重点优化了数据的 I/O 操作性能，力求将多核环境下并行交叉证认方法的效率提升到新的水平。

4.1 初始并行方法的不足分析

第三章提出的方法解决了交叉证认中的常见问题——边界漏源问题，并行化的实现又大幅度地提高了交叉证认计算的效率，使海量数据上的大规模交叉证认真正得以实现，且其方法具有易于理解，便于开发的优点，在各种天文数据访问系统的开发中具有广泛的实际应用价值。

分析上章方法的运行流程，可以看出其主要包含以下几个耗时环节：

1) 以计算块为单位的星表 A 的数据库读取操作
2) 以计算块为单位的星表 B 的数据库读取操作
3) 以 HEALPix 原子块为单位的星表 B 的边界块数据库读取操作
4) 球面距离计算和判断
5) 结果数据写入数据库
6) 其它耗时（任务分配通信等）

为了分析出制约总体效率的主要因素，在以 SDSS 和 TwoMASS 星表为实验数据的实验中，以上每一部分的耗时情况已被统计出来。图 4-1 中描述的就是 HEALPix 索引下计算块块数为12×4^8时的各部分程序用时的所占比例。

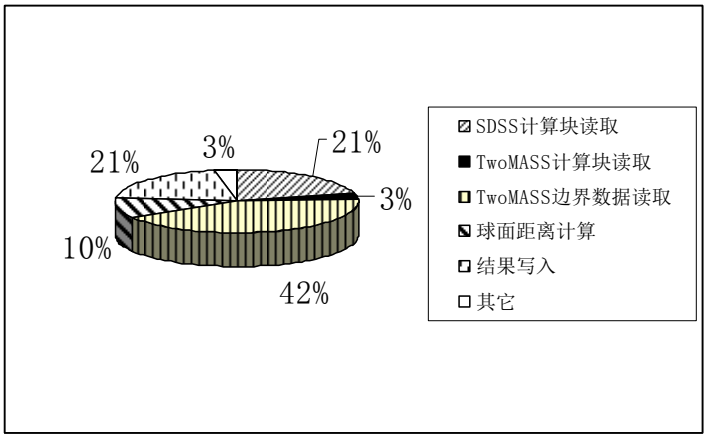


图 4-1 初始方法程序各部分耗时统计

从图 4-1 中可以看出，与数据库读写操作相关的包括四部分，即 SDSS 计算块读取、TwoMASS 计算块读取、TwoMASS 边界数据读取和结果写入，总共耗时占程序总耗时的 87%。在这四部分中，又以 TwoMASS 边界块数据读取耗时最为严重，占交叉证认程序总耗时的 42%，而占数据库读取操作总耗时的比例则接近 50%；SDSS 计算块读取耗时和结果写人耗时次之，都约占了总耗时的 21%；TwoMASS 计算块读取耗时最少，仅占了全部时间的 3%，这是因为 SDSS 数据在部分天区中较为稀疏，以计算块为任务调度单位时，存在很多空块，当判断到当前任务块的 SDSS 计算块为空时，程序即刻返回并开始下一块任务，而不再读取该块的 TwoMASS 数据和其边界数据。从这一现象也可以看出，如果 SDSS 数据在观测上不存在很多空白区域，则 TwoMASS 的数据库读取操作会更为耗时，尤其是数据库访问代价最为昂贵的 TwoMASS 的边界数据。因此，该方法存在的第一个问题是 TwoMASS 的边缘块读取耗时较多，边界漏源问题解决方法的效率仍有进一步优化的空间。

实验所用的两个数据集，无论是 SDSS 还是 TwoMASS 都存在着不同程度的稀疏问题。事实上，天文望远镜在进行观测时，很多都是有针对性地只对特定的天区位置进行观测，稀疏现象非常常见。因此，针对稀疏星表数据加以额外的处理，可以进一步提高程序的性能。

实验中反映出的另一个问题是证认结果向数据库的写入操作耗时相对较多。据统计，证认上的天体对条数总共约为 2000 万条，相当于输入 SDSS 星表记录数的五分之一，而其总共的数据库写入用时却与 SDSS 全部数据的计算块读取耗时相当。这种效率上的差异主要是因为逐条写人的低效造成的。这一问题比较容易解决，只要在程序里对写入操作稍加修改即可。所以，本节的改进方案并不针对这一问题。

4.2 边界数据处理方法的改进

正如上文的分析，前面的算法虽然解决了交叉证认中常见的边界漏源问题，但由此而产生的对边界数据小块的数据库读取操作却成为了该程序中最为耗时的部分，一定程度上影响了交叉证认效率的进一步提升。

实验中计算块划分数为12×4^8时程序效率达到了最优，此时每个计算块由4^5个 HEALPix 原子块组成，由此可以推算出处于计算块块外边界上的数据量与计算块块内数据量的比例约为：$4(2^5+1):4^5$，即约等于 1:7.8。再根据图 4-1 中的比例关系，可以算出对 TwoMASS 数据表的计算块块外边缘数据和边内数据的数据库查询耗时间的大致比值：$3\%:42\%=1:14$。由此可以继续估算出，计算块的查询效率大概是其边缘数据小块的查询效率的$(7.8/1)/(1/14)\approx109$倍，边缘块的查询效率之低可见一斑！经过实验和理论的分析，本文认为边缘小块查询效率之所以明显低于计算块大块主要源于以下原因：

采用 HEALPix 编码的星表记录数据保证了绝大多数在位置上相邻的数据在编码值上也相邻或邻近的特性。图 4-2 描述的是一个计算块（假定一个计算块包含4^3个 HEALPix 索引小块）和其周围一圈数据的编码方式的示意图。从图中可以看出，这个由 HEALPix 划分所得的计算块内的各个原子小块具有连续的编码值，具体来说就是：其内小块编码的最小值发生在该块最右下角的位置，最大值在该块最左上角的位置，而其中所有其他原子块的编码刚好介于此最小值和最大值之间，并按照图示的 Z 型方式依次递增。事实上，所有经 HEALPix 划分所得的计算块都具有这样的编码特性。由此可见，HEALPix 具有的 Z 型嵌套式递增编码方式保证了经 HEALPix 自然划分所得的某层计算块的全部 HEALPix 原子小块具有连续的编码值。该特性的一个好处是，当计算进程需要从数据库中读取某个计算块的块内数据时，它可以直接提交一个对 HEALPix 编码的区间查询。又因为该 HEALPix 编码列已经过 B-tree 索引，根据 B-tree 索引的原理，具有相近索引值的数据记录在存储位置也较为相近，因此区间查询较离散点源查询具有较高的效率。

下面再观察该计算块块外一圈边缘数据的编码，很明显它们彼此之间均不连续，而且存在很大的跳跃性。因此，当计算进程完成了计算块块内数据的加载后，他需要提交并不连续的确定值查询指令。当计算块数量取为12×4^8时，每个计算块包含4^5个 HEALPix 原子块，所以其块外边缘数据就包含了$4\times(2^5+1)=132$块 HEALPix 原子块。相比于中心计算块的整个区间查询，这种非连续值的定点查询将非常耗时。这正是为什么在对程序进行整体分析后，会发现边界原子块的数据库查询效率只相当于较大的计算块的查询效率的 1/109 的原因所在。

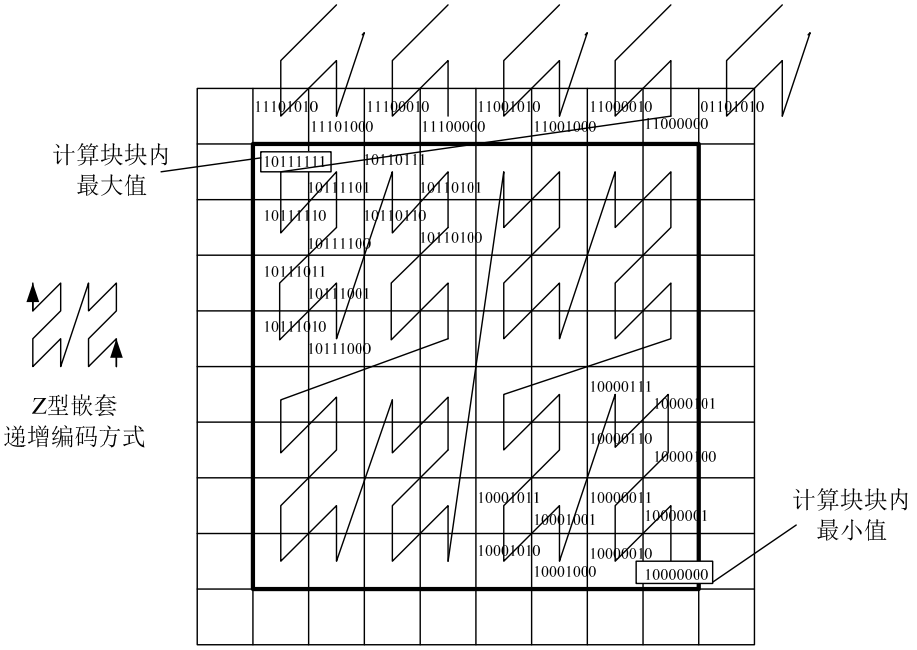


图 4-2 计算块与周围边缘块的编码特点

由此可见，边缘小块查询的低效现象是由 B-tree 索引的特性直接决定的。因此只有避免零散边界数据的查询操作才有可能大幅度地提高交叉证认的整体效率。一种直观的思路是扩大边缘数据的范围：将与某一计算块相邻的几个完整计算块都列为需要证认的边界数据范围，而不是只囊括围绕其一圈的 HEALPix 原子小块。图 4-3 形象地描绘了这种思想。

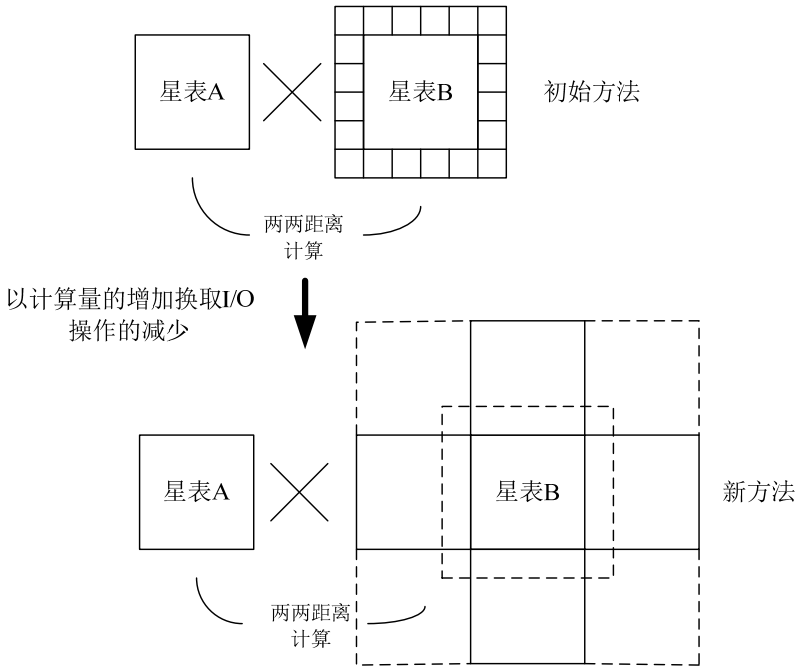


图 4-3 边界问题解决方式的转变

每个计算块周围共有 8 个与之邻接的同样大小的计算块，如果假设这些块内的记录密度基本相同，则这 8 块计算块的数据库查询耗时约为中心计算块耗时的 8 倍。很明显，这些边缘计算块的读取效率还是明显好于初始方法中的 109 倍。因此，经过这样的改进之后，良好的边界块数据加载效率是完全可以期待的。然而，很明显，在数据库访问量大幅下降的同时，程序中的球面距离计算量却明显增加了。因为此时星表 A 中的每个计算块将与星表 B 中的 9 个计算块进行两两天体距离计算，计算量约为初始方法的 9 倍。如果忽略掉四角上的四块，只考虑上下左右四块邻接计算块，则计算增加量降到了初始方法的 5 倍，因此时可能发生的计算精度损失很小，可暂且忽略不计。下面可以证明这种 5 倍的计算增加量完全可以通过增加计算块的划分级数的方式进行抵消。

当计算块的划分级数由 m 增加到 m+1 时，总计算块数量就由12×4^{m}增加到$12\times4^{m+1}$，所以每个计算块的面积大小就减少到了原来的 1/4。这样，平均情况下单位计算块块内的记录条数也减少到了原来的 1/4。如果设两条数据间发生的一次距离计算量为单位 1，单位 HEALPix 原子块内的平均天体数量为ρ，则在划分级数为 m 和 m+1 时，总距离计算量可以分别表示如下：

$$\text{总球面距离计算量}_{\text{计算块划分级数=m}}=12\times4^{m}\times(4^{13-m}\rho\times5\times4^{13-m}\rho)$$
$$=60\times4^{26-m}\rho^{2}$$

$$\text{总球面距离计算量}_{\text{计算块划分级数=m}}=12\times4^{m+1}\times(4^{13-m+1}\rho\times5\times4^{13-m+1}\rho)$$
$$=60\times4^{25-m}\rho^{2}$$

可见，计算块划分粒度每增加 1 级，总距离计算量可减少到原来的 1/4，所以边缘块处理方式的改变所带来的额外计算量就可以通过这种方式加以抵消。然而，另一个问题又随之产生了，正如前面 B-tree 索引特性的分析，在经过 B-tree 索引的编码列上对连续区间执行查询的效率较高，因此当计算块划分得过细时，其查询效率也会降低。幸运的是，实验发现当计算块数量由12×4^{8}增加到12×4^{9}时，计算块的数据库访问总耗时并未发生明显改变。图 4-4 表示的是原实验中在各个计算块划分级数下对 SDSS 所有计算块查询总用时的统计量。由此可以看出，划分级数从 5 增加到 9 的过程中，总计算块查询用时增长幅度很小，当划分级数继续增加的 10 时，查询用时才开始有了一定程度较为明显的上升，但此时上升幅度仍然不是很大，只有当划分级数超过 10 时，总查询时间才开始发生陡然的增加。

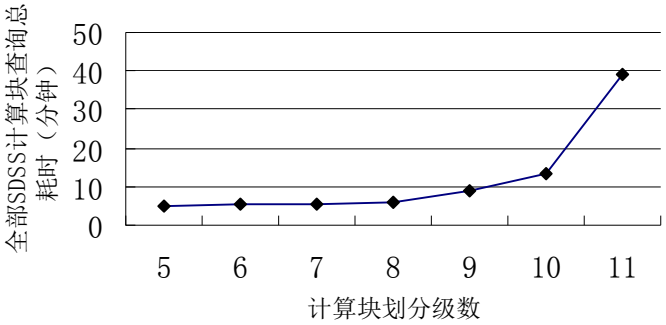


图 4-4 SDSS 中计算块查询总耗时随划分级数的变化

因此，可以认为，只要限定在一定范围内增加计算块的划分粒度，数据库查询速度就不会受到明显的影响，而与此同时所带来的好处却是距离计算量的迅速减少，从而可以抵消掉边界范围增大所带来的数据量的增加。针对不同划分粒度下计算用时和数据库查询用时所形成的此消彼长的态势，4.5 小节将通过实验找到可以使交叉证认总体效率达到最优的划分粒度。

4.3 基于限制生长模型的数据加载处理流程

本节继续围绕如何进一步降低数据库 I/O 操作用时展开研究。仔细分析上节中改进后的边界数据处理方法会发现，按照当前的数据加载、计算流程，对星表 B 的数据库查询量仍存在一定的降低空间。图 4-5 是某局部天区的计算块编码示意图，图中每个小块代表一个计算块单元，其上的二进制编码是该计算块的 HEALPix 编码，处于视觉上直观性考虑，括号中已注明该编码转换成十进制后的数值。此时如果采用根据计算块 HEALPix 编码数值由小到大依次处理的方式，则会出现星表 B 中的每个计算块都会被从数据库中加载 5 次的情况。这是因为，当证认任务执行到某个计算块时，按照边界数据的新方案，计算进程会加载星表 B 中的该块计算块的数据以及该块上向左右相邻的四块。然而每一块计算块都不仅是某一块计算块的相邻块，而是其周围四块计算块的相邻块。以图 4-5 为例，其中计算块的编码最小值为 35，最大值为 144，若按编码递增顺序加载数据、计算距离的话，则星表 B 中每一块计算块都要被从数据库中取出 5 次。以星表 B 中编码为“47”的块为例，当证认任务执行到块“45”时，该块作为块“45”的左邻接块会被取出执行两两距离计算。执行完毕之后，该块数据会从内存中释放，然而此时块“47”的证认任务并没有完全完成，当证认任务执行到块“46”时，它又以块“46”的上邻接块的身份继续被提取出来参与计算，而后又再次被丢掉。与此类似，此后在执行到块“58”、“133”时，“47”块仍然需要被加载、释放。

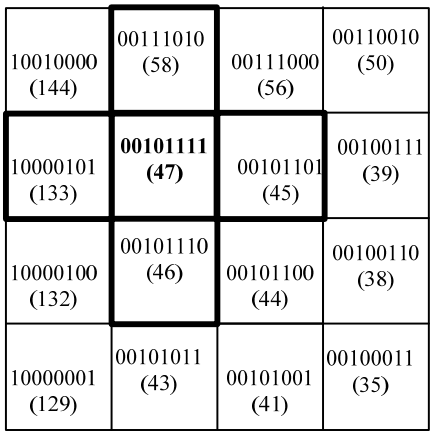


图 4-5 局部天区计算块编码示意图

对同一块数据的重复查询、释放显然是不明智的。但如果对某次证认任务中查询出来的星表 B 计算块数据不执行立即释放，而是继续将其保存于内存中直至全部与其相关的计算任务都已完成，又必将会占用大量的内存空间。而 HEALPix 的“Z 型”递归编码方式本身就存在着编码跳跃的问题。仍然以图 4-5 为例，“47”的相邻左边缘计算块的 ID 激增到 133，如果将“47”一直保存于内存中，直至“133”块任务完成时，则在她们之间将要经过 80 多个计算任务，而这 80 多个计算任务又将产生很多的需要继续保存于内存中的新邻接计算块，所需内存量可想而知。而且示意图只是设定了最高 4 级的编码，实际中编码深度明显大于此时，这种相近块的编码跳跃式增长情况更为频繁，且跳跃幅度也会更为明显，因此程序中将会发现内存空间很快就已占满。由此可见，在数据量很大的情况下，如果按照编码数值递增的顺序来安排证认任务将无法避免相同计算块的多次重复数据库查询。

基于这个原因，重新设计了一种名为“限制生长模型”的任务处理流程模型，从数据的角度来说，也就是重新设计了数据的加载、处理流，从而达到尽量避免对相同计算块的反复查询、减少多余数据库访问操作的目的，同时也避免了对内存的大量占用。

为了减少对内存的占用，同时又不反复加载、释放相同数据，就必须要尽块安排与已在内存中的计算块相关的计算任务，使这些数据都可以尽快地完成它们各自的历史使命以被尽快地清理出去，从而尽量地减少它们在内存中的持续时间。图 4-6 展示了“限制生长模型”的基本原理。

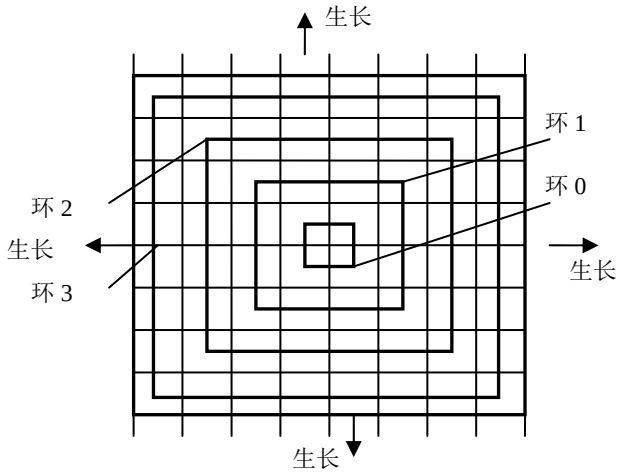


图 4-6 “限制生长模型”下的数据加载、处理流

该模型一个最显著的特点就是证认任务不再按照计算块编码由小到大的推进，而是从中心块逐渐向四周扩展，看起来就像自然界的很多生命体一样逐渐地增长、扩大。图中处于“环 0”位置上的四个计算块被视为“生长点”，它是该块区域中最先开始证认任务的计算块，其后的任务推进方式正是以该四块计算块为中心，呈环形地向外扩展。

- ✧ 步骤 1：计算进程从数据库中依次读取星表 A 中处于内环（即环 0）上的四个计算块的数据；
- ✧ 步骤 2：计算进程从数据库中读取星表 B 处于环 0 和环 1 上的计算块于内存中。之后，星表 A 环 0 上的四个数据块开始它们的全部证认计算任务，因它们各自的所有需要证认的星表 B 中的计算块已经被加载进内存，待证认计算完成之后，所有星表 A 环 0 上的数据将被永久释放，因为此时它们的全部证认任务已经完成。
- ✧ 步骤 3：继续加载数据库中星表 A 的环 1 和环 2 的计算块数据，此时内存中的计算块包括星表 A 的环 1、环 2，以及星表 B 的环 0、环 1。对于星表 B 的环 0 中的每一块而言，它们在星表 A 中的证认对象分别处于环 0 和环 1，处于环 0 上的证认任务显然已在上一步中完成，所以此轮中，星表 B 的环 0 上的数据可以与它余下的所有证认对象（处于星表 A 的环 1 上）完成证认；星表 B 的环 1 上的每一块在星表 A 中的证认对象分别处于环 0、环 1 和环 2 上，其中处于环 0 上的证认任务同样已在上一步中完成，因为此轮中已加载了星表 A 的环 1 和环 2，因此，星表 B 的环 1 上的计算块也可以在本轮中完成全部任务。所以，此步骤下，星表 B 中环 0 和环 1 上的每个计算块在完成它们各自的计算任务后都可以从内存中直接释放掉，且之后不再需要。

- ✧ 步骤 4：此时内存中的数据为：星表 A 的环 1 和环 2 上的计算块。同样道理，为了完成这部分计算块的全部计算任务，计算进程从星表 B 中查询环 2 和环 3 上的计算块。待计算完成之后，星表 A 的环 1 和环 2 上的计算块数据也已被释放，且永远不再需要。
- ✧ 步骤 5，步骤 6，步骤 7…以此类推。

从上面的详细加载、计算顺序叙述中可以看出，无论是来源于星表 A 的数据还是来源于星表 B 的数据，它们的每一环在读入内存之后的下一轮都执行了其对方星表的与之有关的环上的计算块的读取，从而保证了所有的数据都只在内存中停留了很短的时间即被释放。而一个计算进程的每一时刻，其内存中最多只存放了来自于星表 A 和星表 B 各两环的数据量。因此，此模型下的内存填充速度远远慢于按照计算块的 HEALPix 编码递增顺序展开证认的方式。然而即使这样，在处理海量数据时仍然可能会出现内存溢出的现象。这是因为，依照“限制生长模型”，在证认区域由内而外的增长过程中，其环总是在不断扩大的，即环上包含的计算块数量总是在一圈一圈地增加着，因此即使某一时刻内存中仅存储着最多 4 环数据，如果环增长得过大，或是数据过于密集都有可能会导致程序因内存溢出而崩溃。因此，这种生长范围也不应该无限地扩大，应该设法通过某种方式根据内存占用情况对它加以限制。这就是为什么本节将这种任务推进模型称为“限制生长模型”的原因所在。具体的限制方法将下一节中给出，将会看到对范围加以限制不仅保障了内存中的数据量，同时也使数据自然地划分成了一个个相对较小的生长块，从而为实现多进程间的并行处理打下了基础，而较小的生长块体积也更有利于进程间负载平衡的实现。

针对本节设计的图 4-6 中的生长块的形状，有人可能会提出以下的一些疑问，下面将对它们逐一进行分析，以更清楚地阐述本节所设计的“限制生长模型”在几何拓扑形状上的原因：

- ✧ 疑问一：既然由计算块组成的每一环的形状目的是要包围其内环的全部计算块，从而使其内环的全部证认任务可以尽快完成，那么为什么每一环都还要包含四角的计算块？既然每一个计算块仅与对方星表中该块及其上下左右的四块计算块执行证认操作，那是否可以按图 4-7 来构造近似菱形的生长形状呢？

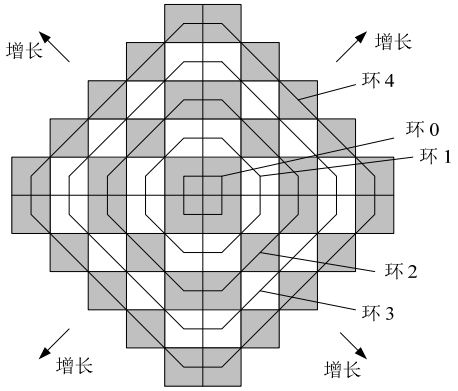


图 4-7 菱形生长模型

原因分析：这种近似菱形的生长模型看似也可以达到限制内存增长速度、避免重复查询相同计算块的目的，但其自然形成的接近于菱形的几何结构破坏了 HEALPix 自然划分下形成的结构。其弊端是，在这种情况下，此生长块不可以由某个 HEALPix 编码直接确定，不便于计算任务的分配和调度。因此，采用正方形的环形状因保存了 HEALPix 划分形成的天然形状，而更有利于程序中的划分、分配和调度。

✧ 疑问二：为什么该生长模型要选取四个计算块为中心生长点，而不直接选择某一个计算块，从而构成图 4-8 中的形态。

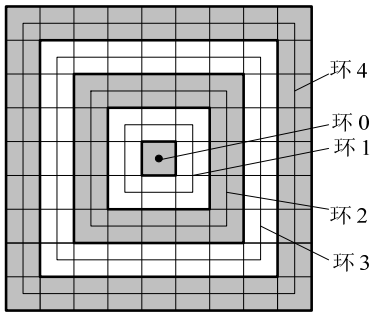


图 4-8 由一个计算块组成生长点的生长模型

原因分析：与上一个问题的原因基本相同，如果以中心的一个计算块为生长点，则生长出来的形状总是一个边长为奇数的正方形，同样破坏了 HEALPix 划分的固有形态。

从以上分析可以看出，本节设计的由四个计算块组成生长点的正方形生长模型既避免了相同数据的多次数据库重复查询操作，又可以通过生长大小的限制控制住内存的占用量，而其所具有的 2^k 的边长长度又完全符合 HEALPix 的自然划分，从而可以继续利用 HEALPix 编码的快速定位能力实现高效的任务分配。

然而，要保证本节提出的限制生长模型可以真正发挥价值，还有一个重要的

前提条件，即必须存在非常高效的“邻接块 HEALPix 编码推导算法”。第六章将详细介绍本文设计的基于位运算操作的快速邻接块 HEALPix 推导算法。

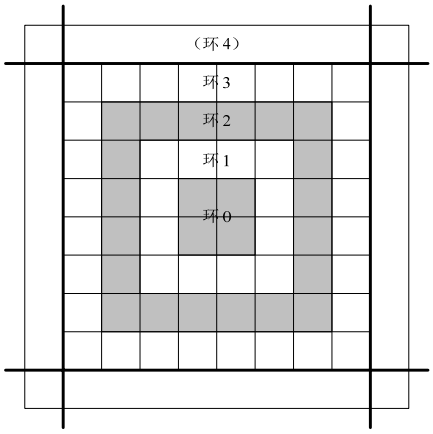
4.4 最大生长块的概念及其确定方法

上节在介绍“限制生长模型”时已经提到，各层环的大小是由内而外逐渐增大的，所以在处理海量数据时，必须要对生长块的大小加以限制，才能满足内存的有限性限制。这也正是该模型名称中“限制”二字的含义。经过限制后的一个个生长块将作为后续并行实现中任务分配和调度的基本单元。本节将给出具体的“限制”方法，即如何将全部计算块划分成一个个“限制生长块”。将会看到，“生长块”的划分方式不仅关系到任务分配时的性能好坏，而且还会关系到数据库的查询量的多少，以及当测试数据存在稀疏问题时的进一步性能优化。

4.4.1 生长块划分时的关键因素

将全部天区划分成一个个生长块时，同样会产生边界数据问题。这是因为，每个生长块的最外环数据块都需要与该生长块之外紧邻的一圈计算块进行证认计算才可以完成它的全部证认工作。如果不考虑这一额外的处理，同样会发生漏源问题。

现以边长为 8 的生长块为例，如图 4-9 所示。其中，粗直线是生长块的分割线，该图中心的生长块由 64 个计算块组成，从内到外共包含环 0、环 1、环 2、环 3 的四个生长环。因为每个生长块必须是 HEALPix 在某层级上划分而形成的自然块，所以其边长必为 $2^k (k = 1,2,3,...)$。该块右面的列表中给出了利用“限制生长模型”时，数据的加载、释放流程。其中，“Ax”的含义为星表 A 中的第 x 环的全部计算块。



数据加载、释放流

	加载	释放
第1轮	A0、B0、B1	——
第2轮	A1、A2	B0、B1
第3轮	B2、B3	A1、A2
第4轮	A3、(A4)	B2、B3、A3、A4

图 4-9 生长块划分时所产生的边界数据问题

从中可以看出，当该流程进行到第四轮的开始时，内存中包含了星表 B 的环 2 和环 3，然后又加载了星表 A 的环 3 和该生长块的外围一环（可看作星表 A 的环 4），所以此时星表 B 的环 2 和环 3 上的每个计算块都可以完成它们各自的全部证认任务。于是，在此轮结束时，除了星表 A 的环 3 以外，该生长块内的全部数据都已完成了它们各自的所有证认任务，也就是说在此之后，程序将不再需要再次读取它们。而星表 A 的环 3 之所以没有完成全部任务，是因为它还需要与星表 B 该生长块的外围一圈进一步进行计算，但在我们的设计中，程序并没有继续加载星表 B 的“环 4”数据。之所以这样设计的原因简述如下：

观察 4-9 中的数据加载、释放流列表可以发现，在奇数轮（$2m+1,(m=0,1,2,...)$）上所进行的数据加载工作总是针对于星表 B 的环“$2m$”，和环“$2m+1$”的，而释放工作则总是针对星表 A 的环“$2m-1$”和环“$2m$”的；而在偶数轮（$2m,(m=1,2,3,...)$）上，所进行的数据加载工作为星表 A 的环“$2m-1$”和环“$2m$”，而释放的数据则为星表 B 的环“$2m-2$”和环“$2m-1$”。又根据上文所述，每个生长块的边长必为 $2^k(k=1,2,3,...)$，其内所包含的总环数必为 $2^{k-1}(k=1,2,3,...)$，因此生长块的环数必为偶数。现排除边长=2 的这种情况，即规定生长块的边长至少为 4，也就是说一个生长块所包含的总环数必为大于等于 2 的偶数。现假设某个生长块由 $2m(m=1,2,3,...)$ 个生长环组成，则当它的数据加载、计算流程进行到第 $2m$ 轮时，所加载的数据必为星表 A 的环“$2m-1$”和环“$2m$” （环“$2m$”为该生长块的外围一圈），此时刚好可以使星表 B 的环“$2m-2$”和环“$2m-1$”完成全部证认任务。因此在此轮结束之后，仍然只有星表 A 的最外一环，即环“$2m-1$”没有完成全部的计算任务。因此按照这种数据加载、计算方式，每个生长块都只留下了其星表 A 的最外一环的计算块没有完成全部计算工作，而其尚需进行的证认工作正好是与其相邻的四个生长块的星表 B 的最外环的一条边间的证认。因这些证认工作刚好会在此四个生长块的证认任务进行时完成，所以按照本节设计的边界数据的处理方法，刚好可以保证所有的证认任务都可以完成，漏源现象并不会发生。

按照这种方法，边界漏源问题虽然得以解决，但每个生长块的星表 A 的最外环的计算块都需要从数据库中查询两次，这无形中再次增加了数据库 I/O 操作的耗时。而且很明显，当生长块越小时，最外层所包含的计算块数量占总计算块的数量就越大。当缩小到极端情况下时，即生长块小到边长为 4 时，16 个计算块中的 12 个都是外环数据块，所以“限制生长模型”所起的减少数据库重复查询的作用已经微乎其微。因此，在考虑生长块的划分时，第一个原则就是：应该避免将生长块划分得过细以至于影响限制生长模型在减少数据库 I/O 方面的作用。

生长块划分时应该考虑的第二个因素是：此划分方式应该可以根据不同区域的数据稀密情况进行自动调节，而不是预先静态的设定一个统一大小。这是因为星表中不同区域上的天体记录分布极不平均，有的位置上的星体可能非常密集，而另一些地方又极为稀疏，甚至基本空白。这种情况下，如果将所有生长块都设定为同一大小，则为了使最密集的地方的生长块所包含的数据不会造成内存溢出，就只能保守地将这一参数值设得很小。而根据上面的分析，过细粒度的生长块划分将导致“限制生长模型”的失效，因此这必然不是一种优秀方案。所以说，合理的划分方式应该可以根据不同区域的星体密度来自动地调节生长块的大小。

空白区域是数据记录最为稀疏的地方，上一点已经考虑了划分方式在数据密度方面应起到的作用，但星表中存在的观测盲区仍然有必要给予额外的考虑。在第三章提出的初始交叉证认算法中，至少对于星表 B 并不需要访问其全部计算块：当星表 A 的某个计算块为空时，随即直接跳到下一块证人任务。而按照本章提出的“限制生长模型”，无论是星表 A 还是星表 B，其几乎全部的计算块都会被查询，即使某个生长块中的大多数计算块都为空。因此，当数据集存在很多空白区域时，如果通过最初的生长块划分就可以过滤掉这些空白区域，则程序的整体效率必将被明显提高，这正是另一个需要考虑的因素。

另一方面，划分所形成的生长块的平衡程度还与并行实现时的负载平衡相关。这是并行程序设计的一个基本原则，过于不平衡的数据划分将导致任务调度时较难实现各进程间的负载平衡。因此，为了并行加速的效果，划分的平衡性同样应该给予一定的考虑。

最后，生长块的划分生成可以说是整个计算过程中的第一步，因此，它的效率非常重要。在这些生长块被确定之后，主进程才能根据各个计算进程的负载情况将对它们进行分配。因此，生长块的划分算法还要满足一定的效率要求。

综合上面各个因素的分析，现将在设计生长块的划分方法时应该考虑的几个基本原则归纳如下：

- ✧ 避免将生长块划分得过细以至于影响限制生长模型在减少数据库 I/O 方面的作用；
- ✧ 保证该划分方式可以根据不同区域的星体密度来自动地调节生长块的大小；
- ✧ 尽量过滤掉空白区域，从而进一步减少数据库的 I/O 操作量；
- ✧ 保证划分后的各个生长块所包含的数据量不会过于不平衡；
- ✧ 划分算法本身应具备较高的效率。

4.4.2 任务分配调度基本单元——最大生长块

本节将介绍一个重要概念——最大生长块，它是并行算法实现中任务分配调

度的基本单元。为了减少边界数据的比例，尽量保证大多数数据只需查询一次，所以生长块在构造上应该尽可能大一些。同时，为了在证认计算过程中减少在空白区域上浪费的额外无意义处理时间，在划定生长块这一步应该尽量过滤掉这些空白区域，于是，“最大生长块”的概念应运而生。

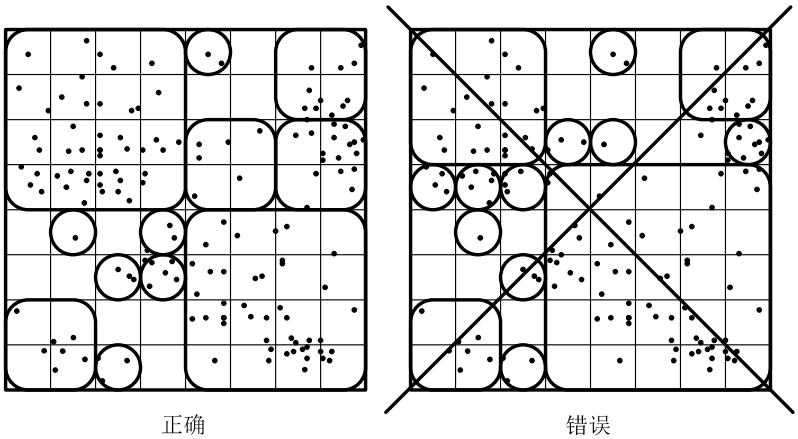


图 4-10 最大生长块的概念

首先选择一个合适的 HEALPix 划分粒度，以该划分下形成的小块为基本单位来判断其是否为空，然后在全天区球面上挖去这些空块，并在余下的球面“残骸”中按照图 4-10 左图中的方式找到一个个最大的 HEALPix 自然块，至此，所得的这些块就是用于后续任务分配的生长块了。在此，有两点需要注意，首先，之所以要在一个合适的划分粒度上去判断块是否为空，而不是简单地以计算块为单位，是因为如果在计算块粒度上判断是否为空，将会导致最终形成很多过于细小的生长块，其中最小的就是“计算块”，这并不符合前面在讨论“生长块划分时考虑的因素”时提到的规定：一个生长块至少由 16 个计算块构成，而且也违背了“避免将生长块划分得过细以至于影响限制生长模型在减少数据库 I/O 方面的作用”的原则。因此，在合理大小的数据块上进行空白与否的判断非常重要。下面的描述中，将选定的用于判断是否为空的判断单元称为 Empty_or_Not block，其大小可以通过实验进行测定。此外，还应注意的是所有最终形成的生长块都是 HEALPix 在某级的划分中所形成的自然块，这样做的好处是：首先，这些“最大生长块”因为是 HEALPix 划分自然形成的，所以它们各自都可以由一个唯一的 HEALPix 编码进行标定，这既便于了程序运行中的操作，同时，当块内数据过于大时，更可以利用 HEALPix 具有的嵌套划分方式容易地实现对其的继续划分。此外，下面在介绍“最大生长块的快速确定方法”时更会看到，按照 HEALPix 自然划分形成的最大生长块更有利于被快速找到，而在程序的并行实现中，具有 HEALPix 编码也使它们更便于进行任务分配。图 4-10 中的左右两图分别展示了

正确的最大生长块划分方式和错误的最大生长块划分方式。

从上面的最大生长块的确定方式可以看出，它已经考虑了前面所总结的生长块划分方式应遵循的原则中第 1 点和第 3 点，下面将继续考虑该原则中的第 2 点，即保证该划分方式可以根据不同区域的星体密度来自动地调节生长块的大小，从而可以在满足内存容量要求的前提下，使生长块尽可能的大。得益于本节设计的“最大生长块”是 HEALPix 划分中形成的自然块，这一条原则很容易被满足，只要对较大的生长块在数据库中执行“SELECT COUNT”指令即可判断出该块所包含的数据是否过多，如果过多，就对其执行拆分操作，即按照 HEALPix 的划分方式将其一分为四，并对这些子块重复执行这一“判断-拆分”工作，直至拆分后的所有子块均满足了预先规定的数据量要求为止。该“判断-拆分”过程可看作“最大生长块”确定后的调整完善步骤，其实质上正是根据不同区域数据的密度进行自适应的修改。经此过程之后的“最大生长块”即可满足原则 2 中的规定。另一方面，更可以看到，这一调整完善步骤将那些具有很大数据量的生长块进行了细分，所以可以认为，在此之后各个“生长块”所包含的数据量更为趋于接近了。由此可见，划分原则中的第 4 条其实已被实现。

确定“最大生长块”一步是本章在对并行交叉证认方法进行改进中所引入的额外代价，因此它的高效性非常重要。研究发现，按照 HEALPix 划分形成的四叉树，采用自顶向下的方式搜索最大生长块具有较高的效率。具体方法是：

1) 确定一个初始最大生长块大小（Initial_Biggest 块），这是因为在过大的区域上进行判断时搜索空间较大，且过大区域上的每一块都不为空的概率很小。

2) 选定合适的 Empty_or_Not 块大小，对每个 Initial_Biggest block 内的 Empty_or_Not 块进行是否为空的判断，即根据每个块的对应 HEALPix 编码向数据库中的两个表分别提交“SELECT COUNT”指令。如果两个表中的任一个返回结果为空，则此块的编码将被记录在内存中的“空表”（Empty Table）中。

3) 采用自顶向下的方法，从 HEALPix 四叉树的根节点（即初始最大生长块）开始判断。判断方式就是在空表中查询此区间内的空白块记录条数，如果为 0，则此块已是一个“最大生长块”，而无需向它的下层继续判断；如果结果正好等于该节点下的 Empty_or_Not 块的数量，则说明该块全部为空，所以直接将此节点完全过滤掉即可。如果查询结果介于 0 与总 Empty_or_Not 块数量之间，则继续判断其四个子节点，直至找到该空间上的全部“最大生长块”。

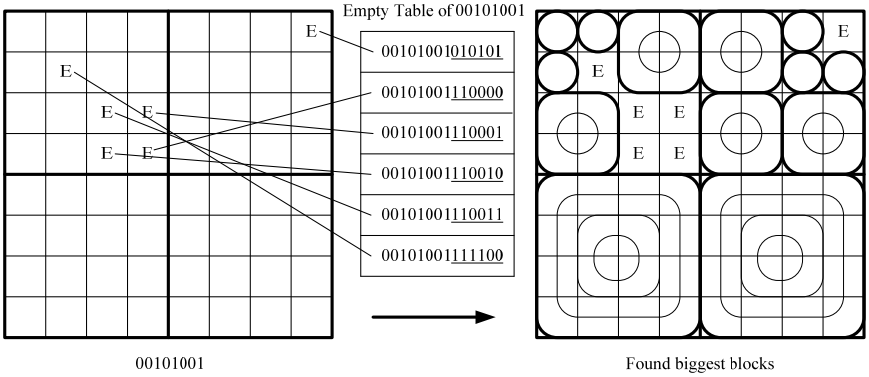


图 4-11 最大生长块的快速确定方式

为了更清楚地表达最大生长块的确定方法，下面以图 4-11 为例进行具体步骤说明。

假设 Initial_Biggest 块大小为4^{11}（即由4^{11}个 HEALPix 原子块组成），则全天区总共包含12×4^2个 Initial_Biggest 块，所以每个这样的块可以被表示为一个 8 位的二进制编码。图 4-11 中的大方块假定即为其中的一个 Initial_Biggest 块，且它的 HEALPix 编码为“00101001”。按照下面的步骤可以确定出其中所有的最大生长块：

1) 对每个 Empty_or_Not 块向星表 A 和星表 B 的数据库提交“SELECT COUNT”查询，将空块的编码记录于它的 Empty Table 内。
2) 在 Empty Table 内搜索以“00101001”为前缀的块编码，如果结果为空，则整个“00101001”块即为一个最大生长块，搜索过程结束。如果结果不为空，而是等于其内 Empty_or_Not 块的数量，则整块区域可以被排除，搜索过程结束。图 4-11 中，这两种情况都没有发生，所以继续在该块的四个子块（“0010100100”、 “0010100101”、“0010100110”和“0010100111”）内进行搜索。
3) 在 Empty_Table 内继续搜索以“0010100100”开头的块编码，结果为 0，说明此块中不包含任何空白块，因此“0010100100”就为一个最大生长块。
4) 在 Empty_Table 内继续搜索以“0010100101”开头的块编码，结果为 1，所以继续搜索“001010010100”、“001010010101”、“001010010110”、“001010010111”四块。
5) 在 Empty_Table 内继续搜索以“001010010100”开头的块编码…
6) 在 Empty_Table 内继续搜索以“001010010101”开头的块编码…
7) 在 Empty_Table 内继续搜索以“001010010110”开头的块编码…
8) 在 Empty_Table 内继续搜索以“001010010111”开头的块编码…

…

理论上，如果该最大生长块上的数据过于稀少，或者在判断到某一层的某个节点时，发现其内的数据量过于稀少时，可以在该局部空间上改为“自底向上”逐渐组合的策略。但实际测试发现，在当前分层规模下这种改进并不需要。

4.5 实验结果和性能分析

为了与第三章中的初始并行方法进行对比，本实验采用了与之相同的软硬件环境。证认部分的并行设计仍然采用了主从模式和动态任务分配的方式，而最大生长块的生成部分由于耗时较小，所以只作为主进程开始的第一步串行执行。整体并行程序流程图设计如下：

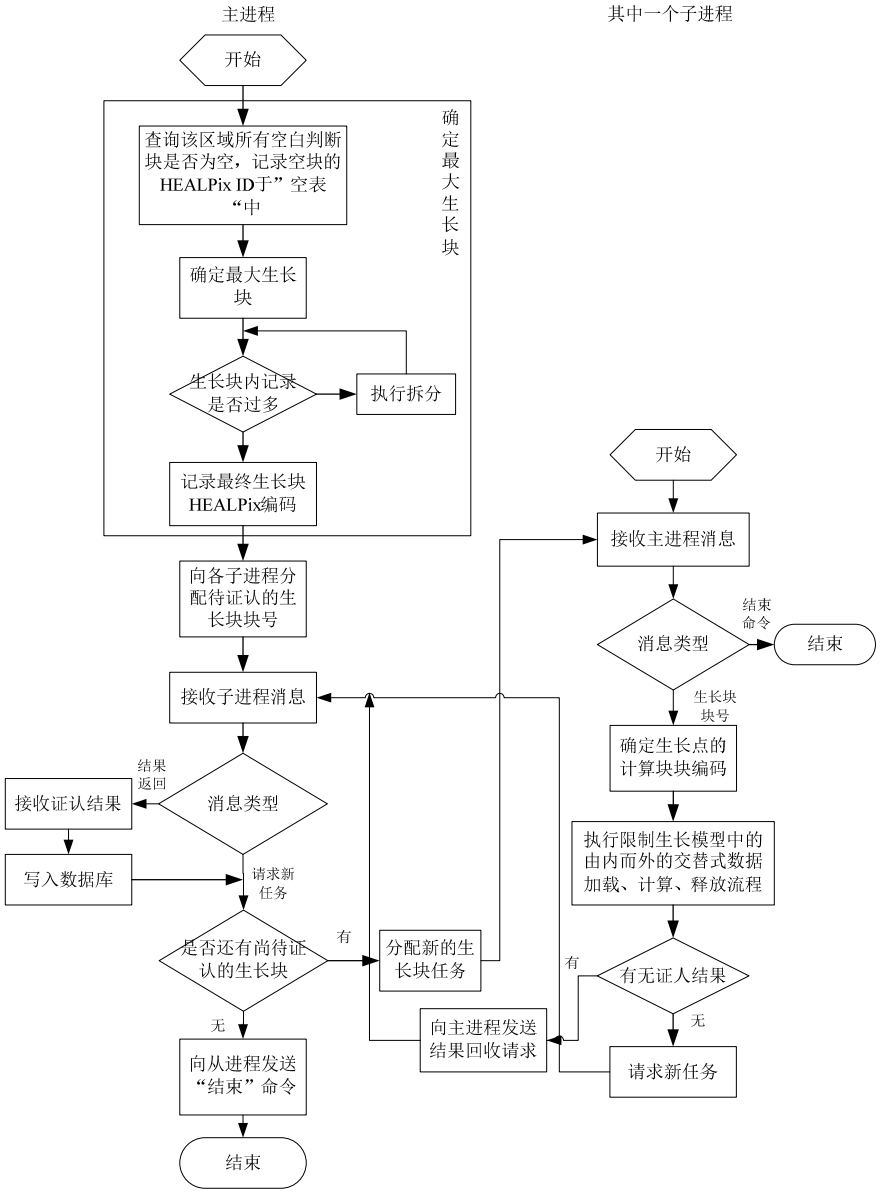


图 4-12 基于限制生长模型的并行交叉证认程序流程

实验结果及分析

为了全面测试本章基于限制生长模型的并行交叉证认方法的性能，在两种不同的数据集上分别做了实验。实验一，为了测试本方法在稀疏数据集上的适应性，采用了与第三章实验中一样的数据集，即 SDSS 和 2MASS 星表的全部数据，它们都存在着很多空白区域，是理想的天然稀疏数据集。实验二，利用 SDSS 和 2MASS 两个星表数据构造了一个实验用非稀疏数据集，通过测试程序在此数据集上的运行情况用以检验：排除对稀疏数据所做的优化带来的性能提升之后，本改进方法是否仍然具有良好的优化作用。

- 实验一：稀疏数据集上的交叉证认实验
  - ✧ 测试数据：SDSS DR6 星表（约 1 亿条数据）和 2MASS 星表（约 4.7 亿条数据）；
  - ✧ 进程数：1 主进程、3 从进程；
  - ✧ 结果对比：现将实验结果与第三章中的初始并行程序结果进行对比，为了清楚地分析影响程序性能的各方面原因，现已分别统计出程序各个部分的耗时情况。因本节所作优化并不针对证认结果向数据库的写入操作，为了更清楚地看到本文的优化对程序总体性能的影响，表 4-1 和表 4-2 中的总时间一项均未包含结果写入耗时。

表 4-1 初始并行方法在稀疏数据集上实验结果

计算块分块数量	SDSS数据库查询	2MASS数据库查询（中心块）	2MASS数据库查询(边界块)	距离计算	其他	总用时
12×4^7	307s	59s	335s	580s	40s	1321s
12×4^8	317s	40s	639s	151s	44s	1191s
12×4^9	427s	54s	1177s	51s	72s	1781s

表 4-2 基于限制生长模型的新方法在稀疏数据集上实验结果

计算块分块数量	SDSS数据库查询	2MASS数据库查询	距离计算	其他	总用时
12×4^7	120s	78s	2489s	48s	2735s
12×4^8	127s	79s	690s	58s	954s
12×4^9	191s	102s	199s	57s	549s
12×4^{10}	374s	239s	58s	67s	738s

对比以上实验结果可以看出以下几点明显改进：首先，在本节的改进方法中，因为对测试数据中大量存在的空块数据执行了过滤操作，消耗于 SDSS 星表数据库的读取时间从 317 秒下降到了 191 秒；然后，得益于 4.2 节中对边缘数据解决办法的改进，原始程序中最为耗时的部分——2MASS 星表的边界数据块数据库查询操作已被省略掉，这也是为什么在表 4-2 中只有一列和 2MASS 星表查询相关的操作。因这一操作原本用时高达 639 秒，占到总时间的 53%，所以改进后新程序的最优总时间的 549 秒相比初始方法下的最优时间 1191 秒减少量高达 53.9%。

- 实验二：非稀疏集上的交叉证认
  - ✧ 测试数据：利用 SDSS、2MASS 星表构建非稀疏数据集，具体方式为选取那些在两个星表中都较为密集的共同区域。最后得到的 SDSS 数据量为 47949212 条记录，2MASS 数据量为 35476377 条记录。
  - ✧ 进程数：1 主进程、3 从进程；
  - ✧ 结果对比：为了有效检验本节改进方法在非稀疏数据集上的应用效果，现同样在此非稀疏小规模数据集上重复了第三章的初始方法，并保证新旧对比实验中的运行软硬件环境完全相同。

表 4-3 初始版本并行方法在非稀疏数据集上实验结果

计算块分块数	SDSS数据库查询	2MASS数据库查询 （中心块）	2MASS数据库查询(边界块)	距离计算	其他	总用时
12×4^7	33s	17s	124s	96s	16s	286s
12×4^8	33s	19s	191s	24s	16s	283s
12×4^9	43s	28s	403s	11s	22s	507s

表 4-4 改进后新方法在非稀疏数据集上实验结果

计算块分块数	SDSS数据库查询	2MASS数据库查询	距离计算	其他	总用时
12×4^7	32s	19s	421s	27s	499s
12×4^8	36s	20s	130s	27s	213s
12×4^9	46s	27s	39s	31s	143s
12×4^{10}	107s	60s	11s	32s	210s

从对比实验结果可以看出，此次实验中，改进后的新方法在12×4^9的计算块划分粒度下取得了最好的性能：143 秒，这一耗时较对比实验中的最好情况下的 283 秒约减少了 50%。下面对这一性能提升的原因作出具体分析。鉴于本实验采

取了非稀疏数据集，SDSS 数据库查询时间无法通过过滤空白块的方式得到削减，所以这部分实验耗时在新程序中并未减少，相反，在计算块划分数量为12×4^8、12×4^9时，这部分反而有略微的上升，这是因为处于每个生长块的最外环的计算块总是要被访问两次造成的。然而，因为此实验数据不同于实验一中的稀疏数据，所以在稀疏集上应用初始方法时，因为 SDSS 的数据块为空而及时省略掉的大量 2MASS 数据读取耗时的情况在本实验中并不会发生，因此在此数据集上应用初始方法时，2MASS 的读取耗时所占的比例明显较高，具体数值为：从实验一表 4-1 给出的 2MASS 读取操作耗时在初始程序中总耗时比例的（40+639）/1191=57% 增加到表 4-3 中的比例（19+191）/283=74%。事实上，这一现象反而增加了程序性能的改进空间，尤其是 2MASS 边界小块的大量耗时，这是因为本节改进方法所针对的重要改进因素之一就是最为耗时的边界数据块读取操作。表 4-4 中的数据正好验证了这一推测，由于成功地应用了 4.2 节中的边界数据处理改进方法以及 4.3 节中的限制增长模型，避免了零散的边界小块的查询，同时又尽量减少了计算块的重复查询，使得 2MASS 查询部分明显减少，同时也使交叉证认整体效率有了显著的提升。这也就回答了为什么在非稀疏数据集上，虽然无法依赖过滤空白块所带来的性能提升，总体交叉证认程序仍然取得了很好的加速的原因。

4.6 面向HTM索引的适应性分析

本章提出的改进算法因优化了数据库的 I/O 操作、解决了稀疏数据集上的空白区域问题，大大地节省了原始算法的运行耗时。而按照 3.5.2 节中的分析，HTM 索引下的交叉证认因需要处理更多的边界数据块，所需的数据库查询耗时更为高昂。因此，如果能成功应用基于限制生长模型的改进方法，将对降低证认总耗时起到重要作用。本节的目的即是分析基于限制生长模型的改进策略在 HTM 索引上的可行性。

4.6.1 面向HTM索引的边界问题的解决方法改进

经过对 3.4 节中方法进行实验测试分析，我们发现计算块周边的原子小块的数据库查询操作仍是本程序中最为耗时的步骤，所以本节同样以扩大证认范围的方式来减少这种零散的数据库查询操作，即以计算量的增多为代价来换取数据库查询量的减少。具体做法如图 4-13 所示。

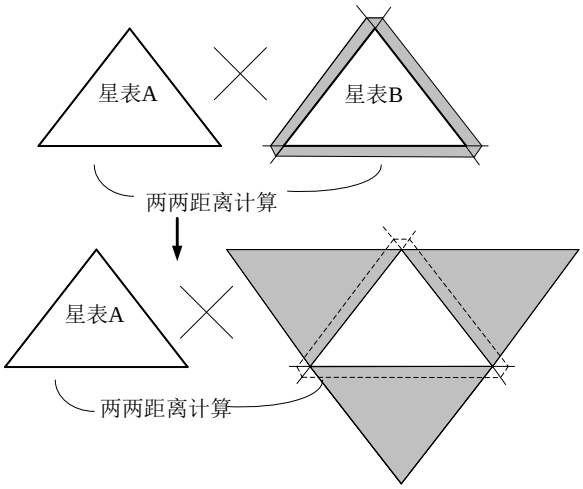


图 4-13 边界问题解决方式的转变

与前面 HEALPix 索引下的处理方法相似，在改进后的边界处理方法中，对每一个计算块仍然只考虑其三条边上相邻的计算块，而没有考虑三角形三个顶点周围相邻的几个计算块。这是综合考虑计算量和计算精度后的一种折中做法，严格意义上讲，每个计算块周围一圈共有 12 个计算块，但与其共顶点的 9 个计算块只含有很少量的数据在证认范围之内，尤其是当计算块尺度远大于误差半径时，因此而发生的漏源可以忽略不计，但省去的计算量却高达 3/4。

4.6.2 面向HTM索引的限制生长模型

这里的设计思想仍然是使新加载到内存中的计算块可以尽快地完成它的全部计算任务然后执行释放，既保证绝大多数的计算块只需被查询一次，又保证在此过程中所占用的内存量尽可能少。图 4-14 是限制生长模型的基本形态。

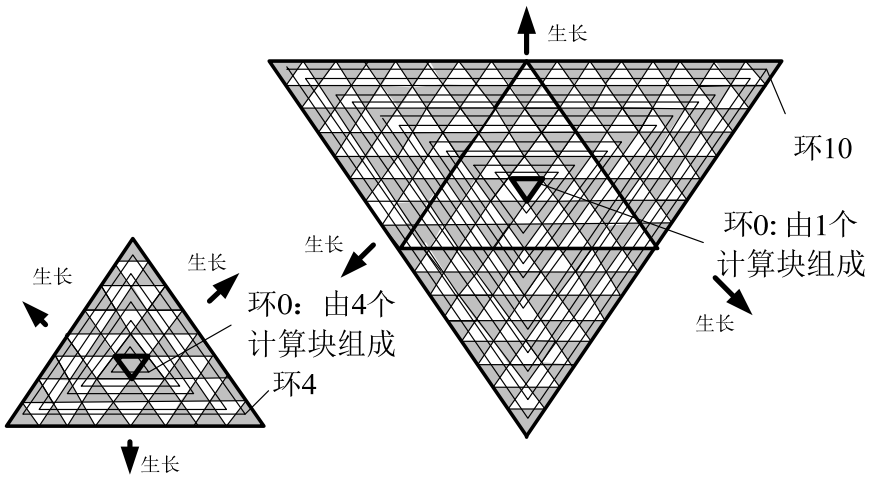


图 4-14 HTM 索引下的限制生长模型

从图中可以看出，HTM 索引下的生长模型就是以 HTM 的基本几何形状——三角形为生长环进行生长的，这种思路与 HEALPIx 索引下选取的四边形为生长环形状的思想一脉相承，但 HTM 下的模型仍然存在两点主要的不同之处。

第一点不同之处是 HTM 的最内环的构成方式与 HEALPix 不同，它在不同情况下有两种构成方式。为了阐述的清晰，本节先定义了两个概念：正三角形和倒三角形。下面将会看到，根据生长块属于正三角形还是倒三角形的不同，其最内环（也可以成为生长点）的构成方式也有所不同。图 4-14 中左侧的三角形即为倒三角形，这与它在图中摆放的方式似乎正好相反。事实上，三角形倒正的定义与其摆放的方位无关，而与它最中心的那块三角形相关。所谓最中心的三角形，就是对该块生长块执行多层四叉树划分时，每次都取其中心一块，直至划分到计算块所在的层次位置，最后所取到的那个中心的三角形即为这个生长块最中心的三角形。图 4-14 中左右两个大小不等的生长块都已用加粗的方式标记出了其最中心的三角形。如果从 HTM 的编码方式上来讲，因为每级的划分后处于中间位置的那块都会在其父块编码的末尾追加“11”两位，因此一个生长块中最中心的三角形就是编码末尾具有最多个连续出现的“11”的那个三角形。定义了最中心三角形，就可以对生长块的倒正进行判断了，其规则就是如果生长块的三角形方向与其最中心的三角形的方向正好一致，则该生长块为正三角形，反之则为倒三角形。基于这种规则就可以看出，图 4-14 中左侧的较小的三角形生长块就为倒三角形，而其右侧的较大的三角形生长块就为正三角形。

根据生长块是属于正三角形还是属于倒三角形，其最内环的构成方式有所不同，具体规定为：如果生长块为正三角形，则其最内环（即环 0）就是最中心的三角形；如果生长块为倒三角形，则其最内环（即环 0）则由中心位置的 4 个三角形构成。这种不同的方式主要目的是保证无论生长块的大小为多少，都包含奇数个生长环。继续以图 4-14 为例，其中左边的较小的三角形就是一个具有倒三角形形状的生长块，所以其最内层的环 0 就是由 4 个处于中心位置的计算块组成的，也就是说在第一轮计算任务时，会加载星表 A 的环 0（4 个中央计算块）的数据，以及星表 B 的环 0（4 个中央计算块）和环 1（4 个中央计算块周围的 9 个白色计算块）的数据。这样从内而外一环一环地、两星表相互交替地对数据执行加载、计算、释放操作，直至进行到该生长块的最外层一环——环 4。因为此生长块中一共包含了 5 个（奇数个）生长环，所以当进行到最后一轮（第 5 轮）开始时，内存中存放的是星表 A 的环 3 和环 4，此时将从星表 B 中加载环 4 以及块外一圈邻接计算块（可看作环 5），因此在这一轮执行到最后时，星表 A 的环 3 和环 4 已完成了全部计算任务，整个生长块内只有星表 B 的环 4 未完成全

部证认任务，事实上它只差与星表 A 的环 5（块外一圈）执行证认。而这种证认除了可以通过在此块任务内通过调入星表 A 的环 5 完成外，还可以在其边缘块证认任务进行到最后时调用星表 B 块外一圈数据（即本块中的环 4）来完成。因此，只要保证每个生长块都具有奇数个生长环，则当它们进行到最后一轮加载计算任务时，都会首先调用环外一圈的星表 B 计算块，这也就是本节之所以要根据生长块属于正三角形还是倒三角形来决定其最内环是由 1 个计算块组成还是由 4 个计算块组成的原因所在。

第二点不同之处是：HTM 下的生长模型中相邻的两环生长环是以互相交错的方式连接的，其接口处呈锯齿形状而非 HEALPix 下的直线，这是三角形划分几何结构直接决定的。此外，每环生长环在构造它的下一环生长环时，不仅选择了与它有邻接边的几个三角形，还额外加入了三个顶点处的对接三角形，这与 HEALPix 下生长环构造思路类似，仍然是保持 HTM 的自然划分形状的需要，如若不然就会形成图 4-15 中的六边形形状，其结果就是无法利用经索引后的 HTM 编码的快速定位能力来实现后续高效的任务分配。

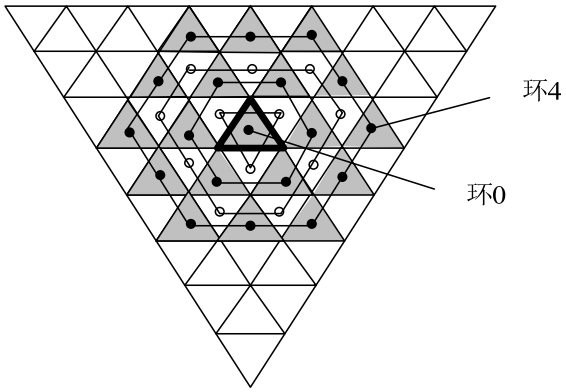


图 4-15 错误的生长块形状

从以上的分析可以看出，只需要在生长块的构造上加入一些额外的设计，就可以使限制生长模型完全适用于 HTM 索引下的天文数据集。下面将继续给出最大生长块的生成方式。正如 4-4 节中的分析，以最大生长块为并行计算下的任务分配单元，不仅可以将数据的重复查询操作降到最低，还可以过滤掉大多数空白区域，从而节省大量的调度、查询开销。事实上，HTM 下限制生长模型中的最大生长块的构造方式与 HEALPix 非常类似，同样包含下面两个主要步骤：一，通过“SELECT COUNT”查询判断每个“空白判断块”是否为空并进行记录；二，按照 HTM 的天然层级划分结构自顶向下地找到最大生长块。其具体做法与 4-4 节中的叙述基本相同。

综上所述，得益于 HTM 索引与 HEALPix 索引划分方式上的相似，本章的

改进思想仍然可以应用于 HTM 索引下的天文数据集的交叉证认。经过这样的改进之后，一方面去除了初始设计中最为耗时的大量边界原子块的数据库查询操作，一方面在高效的预先处理中过滤掉了大量的空白区域。在 3.5.2 节最后的实验部分，已经分析出了：同等划分粒度下，HTM 索引下需要处理的边界原子块数量约为 HEALPix 下的 1.5 倍，因此本节对边界块的处理方式的改进对 HTM 索引的天文数据集同样适用。

4.7 本章小结

本章首先分析了初始并行交叉证认方法的不足之处，包括程序的主要性能瓶颈——过于耗时的数据库 I/O 操作，尤其是边界数据的数据库查询操作、以及未对数据稀疏问题给予额外处理造成不必要耗时等问题，这些正是本章改进方法设计的出发点。针对这些问题，提出了一系列的改进方法，其中包括：解决边界数据问题的新方法，它可以去除边界零散小块的数据库查询，以计算量的有限增加换取了 I/O 访问量的明显下降；基于限制生长模型的全新的数据加载、处理流程，通过它既可以最大限度地避免对相同数据的多次重复数据库查询操作，也可以突破内存的限制；基于“最大生长块”概念的生长块划分方式，既避免了过细划分粒度下高昂的数据库 I/O 操作，也满足了可以根据天然星表中不同区域上的密度的不同进行灵活大小调整的需要，同时也实现了对空白区域的预先过滤。而在并行实现方面，该划分方式所产生的相对平衡的划分大小也使任务分配时的进程间负载平衡较易实现。随后本节继续提出了“最大生长块”的快速确定方法，它使本节的改进方法更具实际操作性，也进一步保证了总体交叉证认算法的高效性。4.6 节在稀疏数据集和非稀疏数据集上所做的两个实验全面地证明了本章提出的基于限制生长模型的改进方法的有效性。在此之后，本章继续证明了此改进并行交叉证认方法在 HTM 索引方式下的可行性，从而拓展了此方法的应用范围。

第五章 基于MapReduce模型的分布式交叉证认

天文数据的海量性和急速增长性已使依赖单一机器的数据处理变得越来越不现实，可以说基于局域网乃至广域网的分布式天文数据处理必将成为未来发展的主流。而交叉证认又是这些诸多天文数据处理的基础性步骤，且从其所涉及的数据量、计算量上来看，它又是一种非常典型的数据密集型天文数据处理难题，因此，研究如何在分布式环境下突破交叉证认的效率问题对天文数据处理领域的发展有着重要参考意义。本章正是基于这些考虑，同时又看到了 MapReduce 这一并行计算模型在处理海量数据上的天然优势，提出了基于 MapReduce 模型的分布式交叉证认方法，希望借助于这种全新的思路为交叉证认这一问题寻求到新的性能突破口，并向实现在线实时交叉证认服务的更高目标迈出关键一步。

5.1 MapReduce模型的适应性分析及算法设计要点

MapReduce 是 Google 提出的一个并行框架，是 Google 搜索技术三大核心技术之一，适合用来处理海量数据上的分布式运算，同时也是当今热门概念云计算[98] [99]的最重要核心技术。当前 MapReduce 已在很多对实时性要求很高的互联网商业应用中获得了很大的成功，究其原因主要是因为它具有以下几个优点：

- ✧ 适合于 TB 级乃至 PB 级海量数据集的处理，尤其适合待处理数据集可以分解成可以单独并行处理的许多小的数据集的情况；
- ✧ 可以利用大量通用机器组成超级集群，在获得强大的计算能力的同时，又保证了成本的相对廉价；
- ✧ 编程方式简单，用户只需考虑如何实现 Map 和 Reduce 函数以满足功能需求，即使完全没有编程经验的非专业人员也可以轻松掌握；
- ✧ 可以将应用程序运行于数千个节点组成的大型集群上，任务调度、节点通信等复杂繁琐的工作已由系统自动完成，而其完善的错误容灾机制更保证了运行的稳定性和可靠性。

基于 MapReduce 的以上特点，经过对天文交叉证认这一计算问题的深入研究，本文认为 MapReduce 模型正好切合了天文交叉证认这一课题的需求。首先，交叉证认是一个典型的数据密集型任务，这是由天文数据的海量性直接决定的，前面在对多核环境下的并行交叉证认的研究中，也已经看到数据 I/O 操作方面的耗时是其性能提高的最主要障碍，而 MapReduce 模型最擅长的正是这种海量数据的处理，它以分布式文件数据库作为海量数据的存储、处理方案，其效率远远

高于当前的任何一种关系数据库；其次，交叉证认计算总是可以通过数据的区域划分来降低复杂度，这是因为只有在一定区域范围内的天体才有可能证认上，并且如果使数据块足够大，就可以保证证认计算在很大程度上只独立地发生在的块的内部，这正好切合了 MapReduce 模型在数据分割方面的要求；再者，天文事业的非盈利性以及观测数据的急速增长，使得它对计算设备、存储设备的成本控制要求很高，而 MapReduce 模型正是这样一个可以充分挖掘大量廉价设备中所蕴含的强大计算能力的分布式模型；此外，从编程角度上讲，不能要求天文专家同计算机专业人员一样具备很多并发处理技巧或分布式系统开发经验，而 MapReduce 模型并不同于普通的并行编程，它在很高层次上进行抽象，从而使编程人员只需关注对具体数据、计算任务的处理，而其强有力的运行时系统更使运行过程中的调度、通信、容错等问题无需经由用户的过多处理，大大简化了程序设计难度。这一模型的很多成功应用已经显示：一个基于 MapReduce 的简单并行处理程序可能仅仅需要几十行代码就可以完成复杂而高效的分布式计算工作。最后，一些支持 MapReduce 模型的较为成熟的开源软件（如 Hadoop）的出现也使得其上应用的开发变得更为容易和可靠。

综上所述，MapReduce 模型的很多特点都切合了天文交叉证认这一应用的实际需要，事实上，该模型与技术在天文研究领域已经开始得到关注，华盛顿大学（University of Washington）所提出的“Scaling the Sky with MapReduce/Hadoop”项目在 2008 年就已获得了美国 NSF 以及 IBM、Google（ACCI）的联合资助，它的主要研究内容是借助于 MapReduce 模型进行天文图像的索引、访问和分析。此外，ACCI 还资助了多项基于 MapReduce 的云计算环境中海量数据处理的课题。因此可以说，本节对基于 MapReduce 模型的天文交叉证认这一课题的研究，不仅能够获得针对大规模交叉证认计算的完整解决方案，也将为未来天文数据/计算中心的高性能数据查询提供一种可行的方案，为其他天文计算的分布式并行化提供参考。

为了充分地利用 MapReduce 模型在大规模数据处理方面的优势，本章在给出具体的设计方案之前，首先分析了 MapReduce 模型下分布式程序的设计要点。

搜索引擎是 MapReduce 模型应用最早也是至今为止应用最为成功的领域，它解决了搜索中数据的海量性和查询的实时性之间的矛盾，而我们的最终目标—高效天文交叉证认查询软件的实现与搜索的过程也是不谋而合的。因此，在设计交叉证认程序时也同样应该参考搜索引擎的高性能架构。

在分布式搜索引擎中，为了达到搜索时的高效性，其前期预处理阶段都要经过网页爬行和索引建立两个阶段。参考网页爬行阶段，其工作目的主要是提取出网页中的有效信息并进行结构化存储，为后续处理做好准备，且因其要面向海量

的互联网数据，这一爬行的过程同样要分布式地并行进行。而在交叉证认中，面向的同样是海量的天文观测数据，因此在初始阶段也需要一个高效的提取相关数据并进行格式转换的分布式并行存储过程。如果把一次交叉证认任务请求看成搜索引擎中的一次搜索过程，则为提高这一搜索过程的效率，前期同样要建立分布式索引，而且该索引在存放上也应具有搜索引擎中索引存放的优越性能，要保证一定的语意相关性，从而才能降低查询语句提交之后的网络通信量。在交叉证认问题中，这一语义相关性其实质就是所属天区位置上的相近性，也就是要保证经索引后分布式存放的数据已存放在了它将要发生证认计算的本地，从而尽可能地避免证认计算过程中的节点间的数据通信。同时，较低的节点间通信量也是获得良好加速比的必要条件，尤其对 MapReduce 模型来说，只有能成功地运行于成百上千的节点才能够真正发挥其在海量数据处理上的优越性。因此，在本章的算法设计中，遵循了这样的原则：要将全部天文数据记录中用于证认的信息字段（包括赤经、赤纬位置信息、星体 ID、索引编码等信息）分成很多小块，从而充分地利用大规模计算集群的计算资源，并且在这些数据块的分发过程中设法保证绝大多数的证认计算可以在单独的块内完成，从而尽量避免证认计算过程中的节点间通信量。

5.2 数据划分及边界数据问题的解决方案

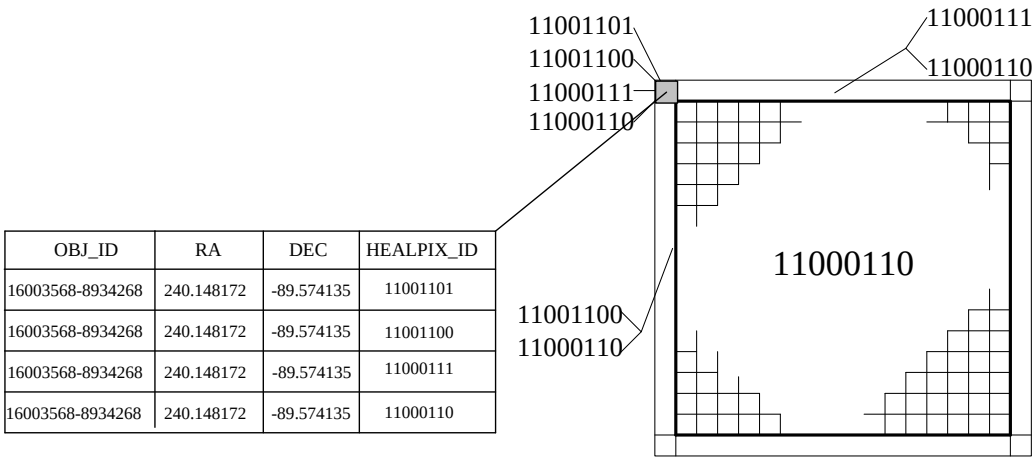
考虑到第三章中在 HEALPix 和 HTM 两种索引方式下所做的综合分析和实验，同时又考虑到 MapReduce 模型自身的特点，本节最终采用了 HEALPix 索引方式，现将理由阐述如下：

- ✧ HEALPix 球面索引方式在划分上具有绝对的等面积性，避免了对赤经坐标的额外修正，也便于选取一个合适的划分级别来满足证认范围的要求。同时相对平衡的数据量划分有利于程序执行过程中负载平衡的保证。
- ✧ HEALPix 与 HTM 在编码方式上都是逐级递归的，所以其编码数值均具有较好的保持局域性的能力，即一定程度上可以保证编码数值较为接近的块其所处的位置也较为接近。但相比之下，HEALPix 近似正方形的几何划分形状较 HTM 的三角形更接近圆形，因此它对局域性的保证也更好一些。
- ✧ 相同的划分粒度下，HEALPix 划分下所产生的边界小块的比例相对较小，下面将会看到基于 MapReduce 模型的交叉证认要通过向原始数据中增加额外的数据存储行的方式解决边界问题，因此采用 HEALPix 划分方式可以减少额外数据的引入量，节省存储资源。

正是基于以上几方面的考虑，本文选取 HEALPix 球面索引方式进行球面数

据划分。首先，选取一个较高的级别对全部数据建立索引，这样每条记录都将根据它的赤经、赤纬两维信息唯一地确定一个 HEALPix 编码。然后选取较低一些的级别来划分出计算块（每个计算块包含 4^n HEALPix 原子块），处于同一计算块内的分别来自星表 A 和星表 B 的数据将进行两两记录间的球面距离计算。计算块所具有的 HEALPix 编码将作为<key, value>二元组对中 key 值，以此为依据完成数据的排序和向集群中各结点的分发，从而最大限度地提升证认阶段的效率。

只要对数据执行划分就不可避免地会存在“边界数据问题”，显然，MapReduce 下的分布式程序也不能幸免。本章在这一问题的解决上采取了不同的方法，不再使用邻接块编码快速计算算法在数据的处理过程中逐块地加载其边界数据，而是在数据的预处理阶段就对处于边界上的数据执行多次打标签操作。其实质上相当于不再使用严格地、一刀切式的划分方法，而是采用弹性边界机制，使星表 B 中的计算块之间存在边界重叠。图 5-1 给出了这一解决方法的示例。



OBJ_ID	RA	DEC	HEALPIX_ID
16003568-8934268	240.148172	-89.574135	11001101
16003568-8934268	240.148172	-89.574135	11001100
16003568-8934268	240.148172	-89.574135	11000111
16003568-8934268	240.148172	-89.574135	11000110

图 5-1 MapReduce 方案中边界数据问题解决方法

从图中可以看出，使用这种解决方式，对处于边和角位置上的天体记录都要执行多次 HEALPix 计算块编码标注，且每一次标注对应的就是数据表中的一行。以图中计算块“11000110”外的左上角的 HEALPix 原子块为例，它本身属于“11001101”计算块内的一个原子块，但由于它刚好处于了四个计算块相交汇的位置，因此同样要参与“11001100”、“11000111”、“11000110”块的证认计算。当执行到 OBJ_ID 为“16003568-8934268”的一条记录时，因为发现其刚好落于此原子块内，所以此条天体记录会被额外复制 3 次存于数据表中，并分别标记上不同的 HEALPix 计算块编码。

采用这样的方式解决边界数据漏源问题，是 MapReduce 模型自身的特点直接决定的。前面已经分析了 MapReduce 模型下的程序设计要点，最重要的就是

保证数据间划分的独立性，减少计算过程中不同节点间的数据交换，从而才能利用大量的计算节点实现具有优越加速比的高性能并行程序，如若不然就不能发挥 MapReduce 真正的价值。然而，这种方式虽然可以获得较好的加速性能，但也会引入很多冗余的数据量，增加数据存储压力，而且计算块划分得越细所需要的额外存储量也就越大。下一节的分析设法解决存储量与计算效率间的平衡问题。

5.3 效率与存储间的平衡

计算块分块数量直接决定了程序中的两个关键因素：球面距离计算量的多少和为了解决边缘数据问题所需的副本数据的额外存储量的多少。所以，为了在存储允许的条件下，使计算效率达到最优，本节将从理论上对各方因素进行分析，5.5 节将针对这一问题给出在各个分块粒度下的实验结果，从而使本文的方法及结论在具有更大数据量的实际应用中更具参考价值。

从理论上很容易推算出总体球面距离计算量随计算块总数的增加而减少。在数据访问服务中，HEALPix 索引的级数一般选为 13，本文也选取了这个级数，即全天区被划分为12×4^{13}个 HEALPix 原子块。经过计算，此时每个原子块的边长大概为 20 arcsec，其范围已经略大于公式 3-1、公式 3-2 确定的需证认范围，所以程序中计算块的划分数量12×4^{m}中的 m 可以取值小于或等于 13 的任意整数。因为星表 A 和星表 B 中同属于同一计算块的两两记录间要进行一次球面距离计算，如果设两个索引小块间全部数据的距离计算量为单位 1，则全天区的球面距离计算量可以从理论上推算如下：

$$\begin{aligned} \text{总球面距离计算量} &= \text{计算块块数}\times\text{每块计算量} \\ &= 12\times4^{m}\times[4^{13-m}\times(2^{13-m}+2)^{2}] \\ &= 12\times4^{13}\times(4^{13-m}+4\times2^{13-m}+4) \end{aligned} \qquad (5\text{-}1)$$

其中，m 为计算块的分块级数，即将全天区分为12×4^{m}块，每个计算块包含4^{13-m}个 HEALPix 原子块。所以，从公式 5-1 可以看出，全部球面距离计算量随计算块总数的增加而减少。

当然，效率问题不仅由球面距离计算量决定，根据交叉证认问题的各方文献，大多情况下其主要性能瓶颈在 I/O 处理方面。经过实验测定，本章基于 MapReduce 的方法中，I/O 的耗时也大概占到了全部耗时的一半左右。但根据 MapReduce 模型，在建立索引之后，全部数据已被分成指定数量的文件小块，且此数量不随计算块分块数量的变化而变化，所以 I/O 的读取次数不会随计算块的增多而增多，

但这部分耗时确实会削弱球面距离计算部分对整个程序运行效率的影响力。

另一个问题是计算块划分过细将直接导致边缘数据所占比例的大幅上升，直接的结果就是为了存储这些边缘数据的副本数据所需的额外存储量的增加。比方来说，假设全天区被划分为12×4^{13}个 HEALPix 原子块，如果对计算块的分块粒度最大化，即以每个 HEALPix 原子块为计算单元，则每个原子块周围的 8 个相邻原子块都是它的邻接边界数据，因此星表的存储量将达到原始数据的 9 倍。而当计算块的大小设为 4 个原子块时，存储量就下降到原始数据的 4 倍。显然，单位计算块取值越大，边缘块所占的比例也就越少，所需存储量也就越少。不同计算块分块粒度下所需存储量的大小可以表示为下式：

$$\frac{\text{星表}B\text{的所需数据存储量}}{\text{星表}B\text{原始数据量}}=\frac{(2^{13-m}+2)^2}{4^{13-m}}=1+2^{m-11}+4^{m-12} \qquad (5\text{-}2)$$

由上面的分析可以看出，随着计算块的划分粒度的加大，计算量和存储量正好有着相反的变化趋势，如果将这两个变化量按照公式 5-1 和公式 5-2 描绘在一张图中，就可以得到如下的一张图。其中，总球面距离计算量的描绘是把计算块划分粒度达到最大值时（即12×4^{13}）的计算量作为 1 的，其它粒度下的球面距离计算量表示的是与此时计算量的比例关系。

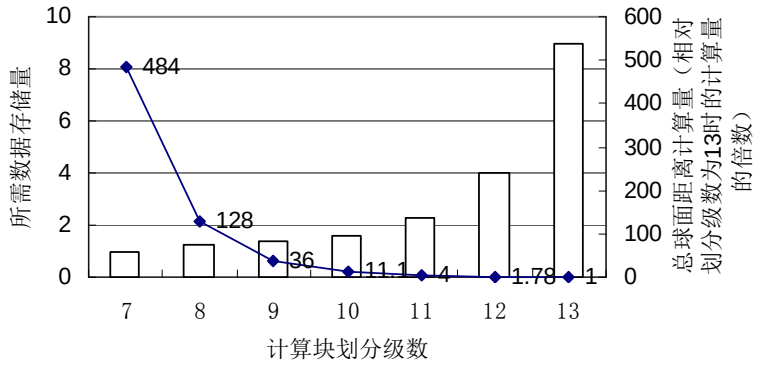


图 5-2 计算量和数据存储量随计算块分块数的变化规律

正如前面分析的，球面距离计算量只是决定效率的一个因素，而 I/O 读取部分的耗时、随着副本数量增多带来的额外的数据处理的耗时都会平抑计算块数增多所带来的球面距离计算量的减少的影响。所以，如何选取一个最优分块粒度，使在不引入过多存储量的同时获得较好性能还需要后续实验加以测定。

5.4 具体算法设计

为了使证认计算部分的效率达到最优，将整个交叉证认的计算过程分成了两个单独的 Map-Reduce 过程：第一个 MapReduce 过程实现了全部数据按照计算块的块号进行排序和向不同节点的分布式存放，所以经过这一过程，具有相同计算块块号的数据已经被传送到同一个计算节点上。这一过程在构建交叉证认服务中，对所有星表只需执行一次，因此可看作预处理过程；第二个 MapReduce 过程实现的是证认计算，由于彼此间需要进行距离计算的数据已经存储于同一计算节点上，不再需要排序和数据传输，所以实际上这一过程只有 Map，没有 Reduce。把能在 Map 中完成的任务尽可能地解决在 Map 中从而省掉包含排序操作的低效 Reduce，也是 MapReduce 并行程序设计中的一个基本原则。

具体的并行交叉证认算法如图 5-3 所示。

1) 证认数据的分布式存放：

- ✧ MapReduce 对输入数据进行自动文件切分，并将数据分块分发到各个节点；
- ✧ 分布式地执行 Map 过程，每次读入文件中的一行，将该记录中的计算块块号+代表星表来源的标志位作为 key，将其他字段的信息以空格为间隔相连接作为 value，从而构成<key, value>元组作为 Map 过程的输出；
- ✧ 按照 key 值（即计算块块号+代表星表来源的标志位）进行排序、分区，并分发给各个节点。这样具有相同计算块块号的记录已经连续存放，可以看作一个计算块组。而在组内，相同来源的数据也是连续存放的。此过程的数据负载平衡已由 MapReduce 自动实现；
- ✧ 各节点执行 Reduce 过程，对每个计算块组读入，统计各组的基本信息，包括该组内来自星表 A 的数据条数，来自星表 B 的数据条数等，作为该数据块组的头文件输出。

2) 证认计算，只包含 Map 过程：

各节点分布式地执行 Map 过程，读入每个文件的数据，此时具有相同计算块块号的数据已经连续排放，对每一个计算块组，先读入头文件，如果来自星表 A 或星表 B 一方的数据量为 0，则对此块数据不执行任何证认。否则，将星表 A 和星表 B 的数据分布读入两个数组，对两数组间的每两条数据记录执行两两球面距离计算，如果距离小于特定值则输出<ID_in_星表 A +ID_in_星表 B, 距离>。

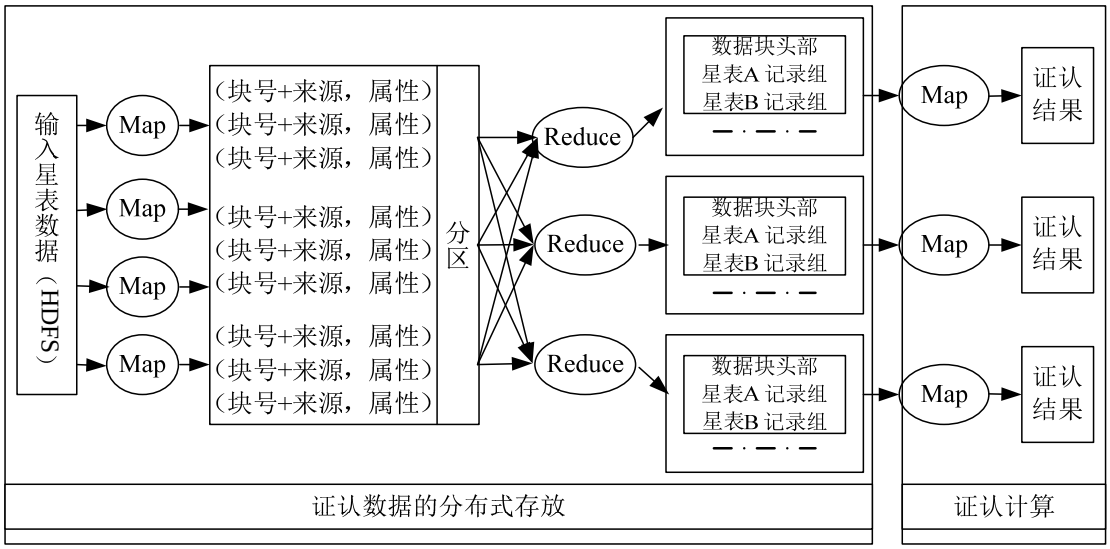


图 5-3 基于 MapReduce 模型的并行交叉证认设计

5.5 实验及结果分析

本文采用 Hadoop 作为支持 MapReduce 程序的中间件，它是由 Lucene Apache 实现的类似 Google 的 GFS 分布式文件系统和 MapReduce 并行模型的开源软件框架，支持运行在由通用计算机组成的大型集群上处理海量数据的分布式应用。

为了测试证认计算部分耗时随机器台数的增加的加速比，在由 64 台普通 PC 组成的集群系统上进行了实验。其中每台 PC 机配置如下：

- ✧ CPU：奔腾双核 E2160，1800mhz，内存：2G
- ✧ 操作系统：Linux Ubuntu 9.04; Swap：2G
- ✧ 编程环境：JAVA

测试数据选择的仍然是美国斯隆数字巡天 SDSS DR6 的星表数据和两微米巡天计划 TwoMass 的星表数据，其数据量分别约为 1 亿条和 4.7 亿条。

实验一：最优分块数量的选择：

5.3 节已经对分块粒度与球面距离计算量间的关系进行了理论上的推导分析，图 5-4 中给出的是在 64 节点机器上证认计算部分在不同分块数下的性能测试结果，这也是本章中最为关心的部分，它直接决定了在更大数据集上实现实时性交叉证认服务的可能性。

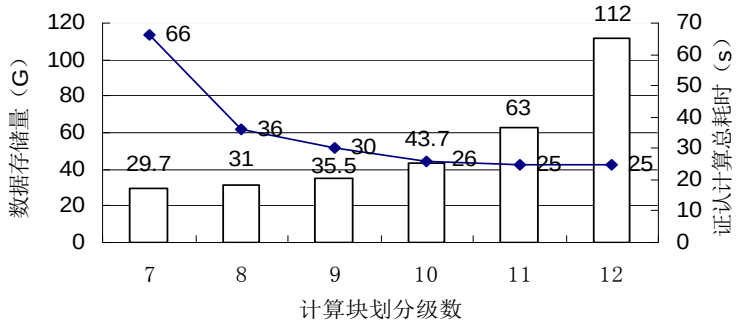


图 5-4 证认计算耗时和数据存储量随分块数量的变化

实验结果证明了 5.3 节中的分析，随着计算块块数的增多证认计算量逐渐减少，由此带来了证认总耗时的减少；而正如前文的分析，球面距离计算量只是决定效率的一个主要因素，I/O 读取部分的耗时、随着副本数量增多带来的额外的数据处理的耗时都会平抑计算块数增多所带来的球面距离计算量减少的影响，所以实验结果中总耗时曲线的下降幅度明显小于图 5-2 中理论上球面距离计算量的下降幅度。在存储量方面，实验中存储的数据在各分块粒度上的变化规律与前面理论上分析的结果基本一致。由此可知，在当前的数据规模下，分块粒度取到 12×4^{10} 或 12×4^{11} 比较合理，在没有引入过多存储量的同时，证认效率可以基本达到最高。

实验二：随节点个数的增加证认效率加速比的测试：

对于并行程序，加速比的高低决定了算法是否具有良好的规模扩展性。本章之所以将整个交叉证认过程分成数据的分布式存放和证认计算两个层次，也是为了尽量将影响加速比提升的数据通信部分尽可能地解决在数据的分布式存放这一预处理部分，以求获得最好证认计算性能，这也是最终实现实时交叉证认服务的基础。而本章的 5.2 节也提到，为了减少证认计算部分节点间的通信，以打标记、复制数据的方法解决了边界数据问题，以存储量的增加换取了计算过程中通信需要的减少，也同样是为了获得更好的加速比。本文分别在 4 节点、8 节点、16 节点、32 节点、64 节点下测试了 12×4^{10} 分块粒度下证认计算部分的性能，结果如图 5-5 所示。

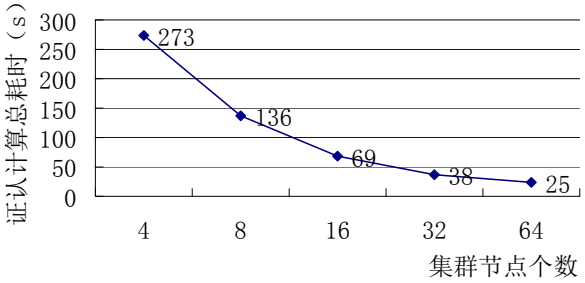


图 5-5 不同节点个数下证认计算总耗时的变化情况

由此可以看出，计算效率从 4 节点上升到 32 节点的过程中，基本上达到了线性加速，而当节点数继续增加到 64 时，效率则提高了 30%，这与 Hadoop 的 MapReduce 本身有一定的启动时间有关，当证认时间缩减到一定程度时，启动时间占有的比例相应提高，所以导致整体性能加速比减少，可以期待的是，如果继续扩大数据规模，应该可以看到更好的加速比。总而言之，经过实验测试，此方法已经具有了接近线性的加速比，这使今后向更大的数据集、更大的集群规模进行扩展成为了可能。

5.6 本章小结

本章提出了一种基于 MapReduce 的分布式天文交叉证认计算方法，将整个交叉证认过程分成了数据分发、证认计算两个独立的 MapReduce 过程。其中数据分发阶段是对数据的预处理，在交叉证认平台中只需对新加入的数据执行一次，所以设计上这部分尽量地包含了全部和通讯相关的工作，从而使证认计算部分的各个子任务可以不需通信地独立完成，最大限度地利用了 MapReduce 模型的特性，保证了证认过程的高效性。实验证明，在上亿条的数据量级上此方法可以快速有效地完成交叉证认工作，其性能远远优于多核环境下基于数据库的并行交叉证认算法，并且随着集群节点个数的增多，其性能表现出了接近于线性的加速比。可见，利用 MapReduce 分布式模型确实可以在普通机器组成的大规模集群上获得强大计算能力，而文件数据库在海量数据处理中确实较关系数据库更具性能优势。因此可以认为，基于分布式文件系统的 MapReduce 模型是天文数据处理中的一种新的可行方案，本章方法的提出为实现实时大规模交叉证认平台打下了基础，对今后提高天文学家对当今海量天文数据的利用效率方面具有重要的参考意义。

第六章 面向HEALPix和HTM索引的邻域编码快速计算算法

本章关于 HEALPix 和 HTM 索引下的邻接块编码快速计算算法的研究既是对前几章研究的必要效率支持，同时也是很多天文数据处理应用中的重要基础。无论是第三章中提出的多核环境下的初始算法，还是第四章中提出的基于限制生长模型的改进算法，亦或是第五章介绍的基于 MapReduce 模型的分布式算法，都无可避免的要反反复复地求解计算块周围相邻块的编码。前面各章的实验中都已分别证明了各个方法的高效性，而这些高效并行算法能够成功实现的一个原因就是得益于本章研究的基于位运算的高效邻接块编码计算算法。此高效算法不仅在交叉证认这一应用上必不可少，也在诸如锥形检索等其它众多天文数据处理上有着广阔的应用空间。因此，通过深度剖析 HEALPix 索引和 HTM 索引的划分、编码方式，研究出具有通用性的快速邻接块编码计算算法具有非常重要的实际意义。

本章将全面地给出 HEALPix 索引和 HTM 索引下的邻接块编码计算算法。6.1 节中将首先对前面各个交叉证认算法所需的邻接编码计算算法的具体需求进行深入的分析，以便后续可以更有针对性地解决这一问题；在此基础上，6.2 节和 6.3 节分别针对于 HEALPix 索引下和 HTM 索引下的的编码计算算法展开深入研究，以全面地解决这一问题。

6.1 邻接块编码推导之详细需求

从前面几章的并行交叉证认算法设计中可以看出，HEALPix 和 HTM 两种天文数据索引方式所具有的层级递归的划分和编码方式在数据处理中有着诸多的优势，可以说它们对高效交叉证认算法的实现起到了至关重要的作用。然而，另一方面，与 3.1.2 节中介绍的简单网格天区划分方式相比，这些复杂的划分方式在为程序带来了诸多优越性能的同时也造成了一些困难，其中之一就是邻接块编码的计算算法变得不再直观，必须要通过对它们的划分规则进行深入的分析和理解后才能实现。而且，因为邻接块编码计算算法在很多应用中都是一个常用的基础性操作，频繁的使用率增加了对它效率的要求。不同应用对邻接块编码推算算法有着不同的需求，或针对于同等划分级别上的邻接块，亦或针对于更高划分级别下的大量邻接块；或只针对于紧邻的一圈邻接块，亦或针对于周围的多圈邻接

块。

邻接块编码计算算法的应用：

1)　多核环境下初始版本并行交叉证认方法

从第三章的详细方法介绍中可以看出，当计算进程每执行一块计算块证认任务时，都要设法计算出该计算块周边一圈原子小块的编码 ID，以通过它们向星表 B 的数据库递交数据查询指令。因此，此交叉证认方法中所需的邻接块编码计算算法针对的是更高划分级别下的邻接块的。而且，考虑到不同星表所具有的误差各不相同，针对不同数据目标此方法有可能需要查询计算块周围多圈原子块的编码 ID。

更高划分级别下的邻接块编码计算方式可以转换成两个原子操作的组合：

- ✧ 同等划分级别下的邻接块编码计算
- ✧ 块内边界小块的编码计算

以图 6-1 中 HEALPix 索引下的更高划分级别的邻接块推导为例，按图中的两种方案组合上面的两个原子操作均可以实现计算块周边原子块的编码计算，很显然，方案一较方案二节省了 $n-1$ 次同等划分级别下邻接块编码计算，因此多核环境下的初始版本方法中的相邻块推导问题可以按照方案一转化为同等划分级别下的邻接块编码计算和块内边界小块的编码计算。HTM 索引下的该问题也可以按照这一较为优化的转化方法进行转化，如图 6-2 所示。本章的第 2 节和第 3 节分别给出了 HEALPix 索引和 HTM 索引中针对这两种原子编码计算的计算算法。

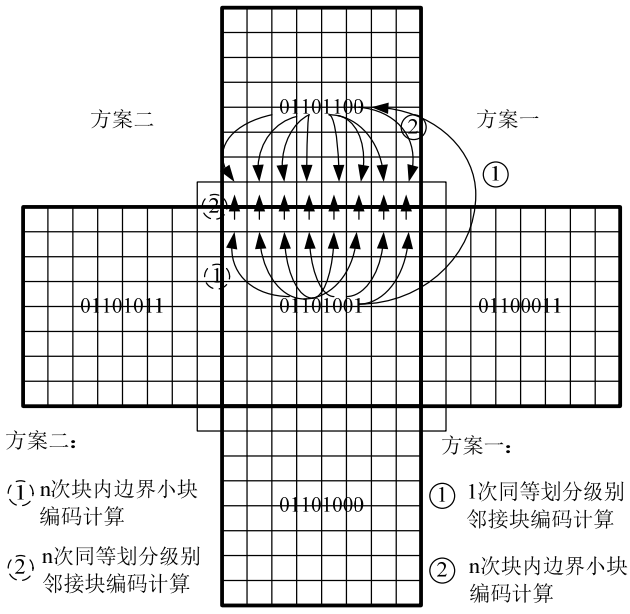


图 6-1 HEALPix 索引下更高划分级别下的邻接块编码计算的转化方案

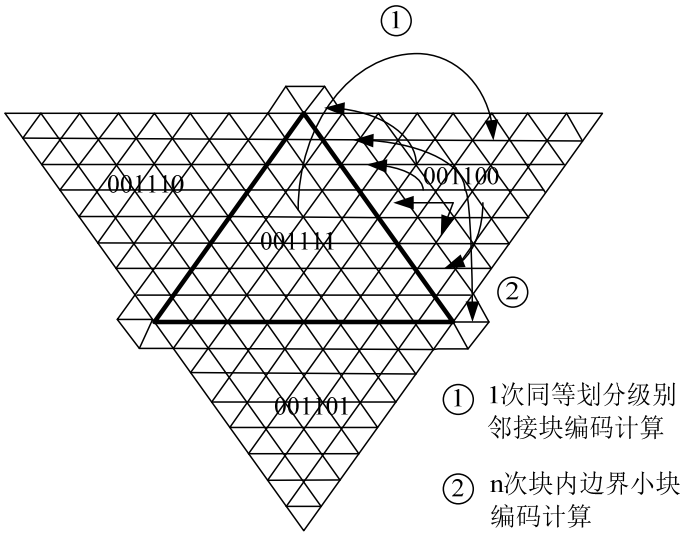


图 6-2 HTM 索引下更高划分级别下的邻接块编码计算的转化方案

2) 基于限制生长模型的改进版并行交叉证认方法

按照前面的详细算法设计，在这种改进后的方案中只需计算每个计算块周围的几个同等级别的计算块的编码即可。以 HEALPix 为例，如果按照计算量与证认精度的折中思想，只考虑边上邻接的几个计算块，则只需要计算其上下左右四个计算块的编码，如果采取严格方案，则其四个顶点处对应的相邻计算块的编码也需计算，而这种编码的计算可以转换成两次边邻接块的编码计算，如图 6-3 所示。HTM 下的邻接块编码计算也有类似的转换方式。因此，在基于限制生长模型的改进方法中，只需要同等划分级别下的边邻接块编码计算算法即可。

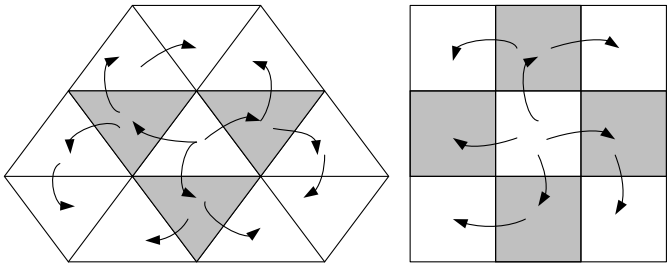


图 6-3 基于限制生长模型的改进方法中邻接块编码计算

3) 基于 MapReduce 模型的分布式交叉证认方法

正如第四章中的分析，按照 MapReduce 模型下的并行程序的设计要点，为了尽量减少计算时的节点间数据传输，采用了加入冗余数据量的方式解决边界数据问题，出于节省存储空间的考虑，这里所需的边界数据块是计算块周围一圈的原子小块。这与多核环境下的初始并行方法一致，需要计算的也是更高划分级别下的一圈邻接小块，因此同样可以采用图 6-1 和图 6-2 中的转换方法，实质上所

需的仍然是同等划分级别下的邻接块编码计算和块内边界小块的编码计算。今后如若根据证认目标星表的精度，可能需要调整邻接小块的宽度，6.2 节、6.3 节的具体算法研究中将会看到，得益于 HEALPix、HTM 的层次划分机制，这一调整很容易实现。

综上所述，同等划分级别下的邻接块编码计算算法和块内边界小块的编码计算算法是本节研究的关键。

6.2 面向HEALPix索引的邻接块编码计算方法

6.2.1 同等划分级别下邻接块编码计算算法

按照上节的分析，同等划分级别下的 4 个共边邻接块的编码是编码计算中的一个关键。下面将详细地给出 HEALPix 索引下此 4 个位置上的邻接块编码计算算法。将会看到，本小节设计的算法充分地利用了 HEALPix 的嵌套递归的编码规则，使每个邻接块的编码计算过程都转化成了 1 到 2 步的异或位运算，保证了计算过程的高效性。

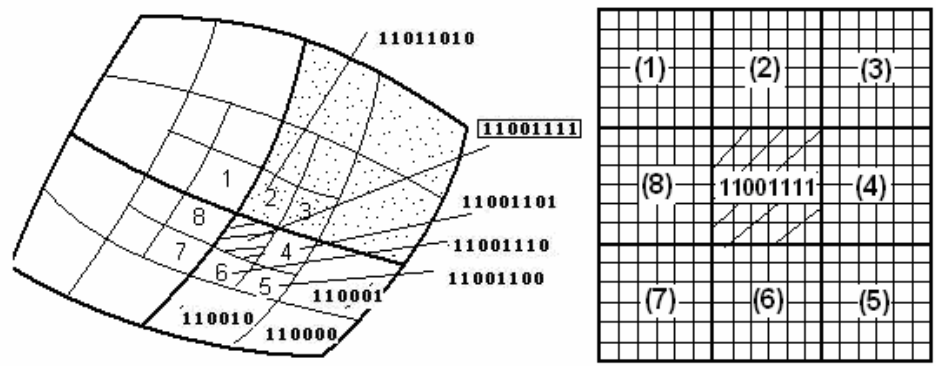


图 6-4 HEALPix 索引下邻接块编码计算算法

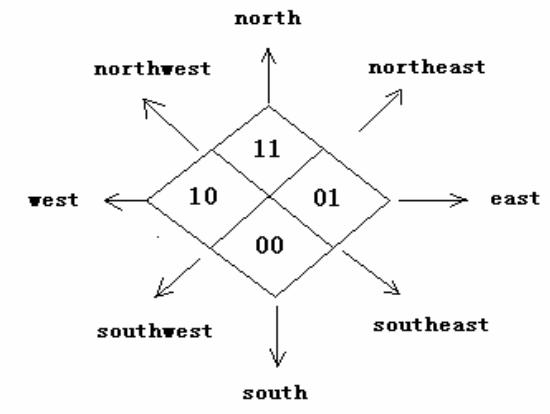


图 6-5 编码规则和方向约定

以图 6-4 中的块“11001111”为例，最终目标是推算出该计算块周围的标记为“2”、“4”、“6”、“8”的 4 个块的 HEALPix 编码。为了表述的方便，图 6-5 给出了方向上的约定。首先推导标记为“2”的 HEALPix 块的编码：

首先观察“11001111”的末尾两位，为“11”，这说明在最后一轮的 HEALPix 划分中该块被划分在了北边方向位置上，所以在倒数第二次划分中，“11001111”块与其东北方的共边邻接块并未被划分在同一块内。因此，只转换“11001111”的末两位不能够得到该块的编码。按照这样的分析，首先应该做的是对编码“11001111”从低位向高位寻找第一次出现的“00”或“10”两位，这样，通过将“00”转换为“01”或是将“10”转换为“11”就可以确定出目标块编码的前几位。对于此例中的“11001111”，首先找到的就是中间的两位“00”，现将此二位“00”替换为“01”，并让更高两位的“11”保持不变，就可确定目标块编码的最高 4 位，即为“1101”。实际上“1101”所代表的就是图 6-4 中左边图中布满点的大块区域。下面继续缩小该区域的范围，通过确定末 4 位的编码将其具体定位到目标块的位置上。

然后再观察“11001111”编码中“00”两位后面的编码，即它的后 4 位“1111”。显然，在最后的两轮划分中，该块都处于正北方向的位置上。这说明标记为 2 的目标块在它所在的父级大块“1101”中的最后两轮划分应该朝正西的方向，因此按照图 6-5 中的编码规则，它的末四位应该是“1010”。

至此，“11001111”正北方向的共边邻接块的编码已被确定为“11011010”。经过分析可以发现，上述复杂的推导过程究其本质可以抽象为一次“11001111”与“00010101”之间异或位运算。异或运算符左边的第一操作数显然就是原始块的 HEALPix 编码，而异或运算符右边的第二操作数则是如下求得的：对“11001111”从低位向高位找到第一次出现的“00”或“10”的位置，从该位开始直到最后一位间的每两位均变成“01”，而更高位上均为“0”。

按照类似的方法，“11001111”块周边“4”、“6”、“8”块都可以通过这样的异或位运算计算出来，只是在求异或位运算符右边的操作数的方法上可能会略有不同，先将这种规律归纳如下：

1) 如果最终目标是求东北方向的共边邻接块，即图中标志为“2”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“00”或“10”，从该位开始直到最后一位间的每两位均变成“01”，而更高位上均为“0”；

2) 如果最终目标是求西南方向的共边邻接块，即图中标志为“6”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“00”或“01”，从该位开始直到最后一位间的每两位均变成“01”，

而更高位上均为“0”；

3) 如果最终目标是求东南方向的共边邻接块，即图中标志为“4”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“11”或“10”，从该位开始直到最后一位间的每两位均变成“10”，而更高位上均为“0”；

4) 如果最终目标是求西北方向的共边邻接块，即图中标志为“8”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“00”或“01”，从该位开始直到最后一位间的每两位均变成“10”，而更高位上均为“0”；

得到了异或右侧操作数之后，就可以通过异或操作完成图中块“2”、块“4”、块“6”和块“8”的编码计算：

- ✧ 块“2”编码：$11001111 \oplus 00010101 = 11011010$
- ✧ 块“4”编码：$11001111 \oplus 00000010 = 11001101$
- ✧ 块“6”编码：$11001111 \oplus 00000001 = 11001110$
- ✧ 块“8”编码：$11001111 \oplus 00101010 = 11100101$

而“1”、“3”、“5”和“7”块则只需再进行一次这样的异或位运算：

- ✧ 块“1”编码：$11001111 \oplus 00010101 = 11011010$
  $11011010 \oplus 00101010 = 11110000$
- ✧ 块“3”编码：$11001111 \oplus 00010101 = 11011010$
  $11011010 \oplus 00000010 = 11011000$
- ✧ 块“5”编码：$11001111 \oplus 00000010 = 11001101$
  $11001101 \oplus 00000001 = 11001100$
- ✧ 块“7”编码：$11001111 \oplus 00000001 = 11001110$
  $11001110 \oplus 00101010 = 11100100$

6.2.2 块内边界子块编码计算算法

上节已经通过高效的异或运算解决了同等划分级别下的邻接块计算，因此，基于限制生长模型的并行交叉证认方法中的邻域编码推算问题已经解决，按照6.1节的分析，初始多核环境下的并行交叉证认方法和第五章基于MapReduce的分布式交叉证认方法都需要对更高划分级别上的邻接小块进行快速编码推导，而这就需要块内边界小块编码计算算法，根据待证认星表精度的不同，所需编码推导的边界小块的宽度也有可能等于1或大于1。经过本节的分析将会看到，利用HEALPix的编码规则这些问题均可快速解决。

仍以6.1节的图6-1中计算“01101001”块的上方邻接HEALPix原子小块为

例，当前应用 6.2.2 节中的基于位运算的同等划分级别下的邻接块计算算法已计算出这些小块所属的大计算块编码为“01101100”，继续利用 HEALPix 的编码规则，就可以计算出该大块块内的下边界原子小块的编码。

如果最终需要的边界数据宽度为 1，则只需要计算 01101100 块的最下方一行原子块的编码。根据 HEALPix 的编码规则，这些沿着下边缘的原子小块的后缀编码必定全部由“00”和“10”组成，这是因为只有在每一层的划分中都被分于下方的位置，最终才可能处于下边缘的位置上。如果该计算块的大小为 4^k 个 HEALPix 原子小块，即原子块与计算块之间相差了 k 个划分级别，则这些紧邻下边缘的原子块的编码的后 14 位就是任意“00”或“10”组合 7 次所得。根据排列组合原理，共可以组成 2^7 个不同编码，正好为该计算块的边长。

如果最终需要的边界数据宽度大于 1，并且为 $2^m(m=1,2,3,...)$，则可把该问题转化为在比原子块划分层次低 m 层上的宽度为 1 的紧邻边界的块的编码推导问题；如果，宽度大于 1，且不为 $2^m(m=1,2,3,...)$，也可以通过 HEALPix 的编码规则推算出每一个原子小块的编码，出于计算简便的考虑，也可以适当加宽边界宽度到 $2^m(m=1,2,3,...)$，只是会引入少量的证认计算量。

6.2.3 实验结果

第三章和第四章的三种交叉证认方法的实验已经显示了本节提出的两个 HEALPix 索引下编码计算算法的高效性，为了更清楚地测试它们的效率，现专门对编码计算操作进行了独立实验。实验环境与第三章多核环境下并行交叉证认实验完全相同。

实验内容为计算 12×4^8 个 HEALPix 计算块中的每个计算块周围一圈的 2^5 个邻接 HEALPix 原子块的全部 HEALPix 编码，该计算过程既包含了 12×4^8 次“同等划分级别下的邻接块编码计算”和 12×4^9 次“块内边界小块编码计算”。最终程序运行结果显示：总共耗时为 0.82 秒。

由此可见，本节设计的基于异或位运算的邻域编码计算算法确实具有很好的效率，完全可以满足当前高效交叉证认的需要，即使考虑到未来实时交叉证认实现的需要，该算法的性能也可基本满足要求。

6.3 面向HTM索引的邻接块编码计算方法

鉴于 HTM 索引在当今天文数据访问服务中的广泛应用，其上邻域编码的高效计算算法有着至关重要的意义，本节将围绕这一问题展开研究。6.1 节中已经分析了该算法在本章三种交叉证认方法中的具体用途和详细需求，与 HEALPix

索引一样，最终同样归结为对两个基本算法的研究：同等划分级别下的邻接块编码计算方法的研究和块内边界小块编码计算方法的研究。这两个基本算法不仅是实现高效天文交叉证认的需要，在其它多种天文数据处理中也是最为常用的基础性操作，因此它们能否被高效实现意义重大。

6.3.1 同等划分级别下邻接块编码计算算法

HTM 索引的三角形几何划分结构致使它的编码规则具有很多独特之处，与 HEALPix 不同，它在各层上的划分小块并非完全同向的，而是具有正三角形和倒三角形之分。而且，在讨论邻接块编码计算算法时，也无法以上下左右或东南西北来标识方向，而要视它的编码进行额外标识。为了本节算法表述的方便，下面首先给出一些自定义的约定。

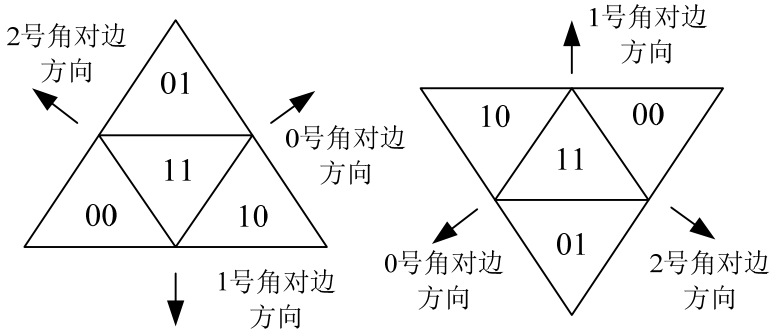


图 6-6 HTM 编码规则和方向约定

图 6-6 给出了正反两种方向三角形的编码规则，从中可以看出，左边的三角形中编码为“01”的子块处于该三角形上部的位置，而右边的三角形中编码为“01”的子块却正好相反地处于它的下部位置；同样两个三角形中的“00”块和“10”块也都有类似的方向相反的性质。因此，为了编码计算算法中统一规律的总结必须依照编码规则定义方向，而不能以这些 HTM 块所属天区物理上的方位来定义，否则就要分别推导正反两种三角形方位上的邻接块编码计算算法。本节规定：某个 HTM 块的下一级划分形成的四个子块中，与编码为“00”的子块的对边垂直指向三角形外的方向为“0 号角对边方向”，与编码为“01”的子块的对边垂直指向三角形外的方向为“1 号角对边方向”，与编码为“10”的子块的对边垂直指向三角形外的方向为“2 号角对边方向”。

下面以图 6-7 中的标记为灰色的计算块的邻接块推导过程为例，阐述该算法的具体执行步骤。

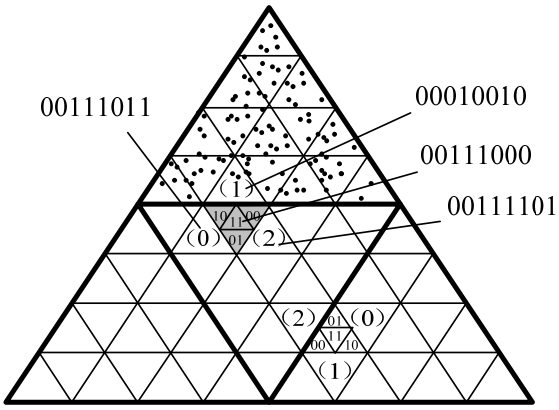


图 6-7 HTM 索引下邻接块编码计算算法

最终的目标是计算出图中块“00111000”周边的三个共边邻接块的 HTM 编码。这三个邻接块分别处于它的 0 号角对边方向、1 号角对边方向、2 号角对边方向，后续为了表述清晰，已在图中分别对这三个块进行了数字标记，方法即是：0 号角对边方向的邻接块标记为“0”，1 号角对边方向的邻接块标记为“1”，2 号角对边方向的邻接块标记为“2”。现首先推导“1”号块的编码计算方法。

推导思路上与上一节中 HEALPix 索引下的同等划分级别下的邻接块编码计算算法非常类似。为了找到“00111000”块 1 号角对边方向上的邻接块编码，就必须先找到它在哪一层的划分中被分向了编码为“11”的子块或“01”的子块，因为只有在这一层划分上“00111000”块与计算对象块“1”才在同一个父亲块内，才可以通过编码变换确定出块“1”的开头几位。于“00111000”块做法就是从右向左两位两位地搜索编码中第一个“01”或“11”，发现当搜索到位置 3 和位置 4 时刚好找到了第一个符合要求的编码对，为“11”。此时说明在这一次划分中它被分到了中间的“11”子块中，又由于这两位右侧不存在“01”或“11”，即其后续的每次划分都被分到了“10”或“00”的位置，也就是正好是 1 号角相对的方向。这就说明“00111000”块必紧邻其父块“0011”的 1 号角对边，因此，目的块必属于“0011”块的 1 号角对边方向的邻接块“0001”，即为图中圆点背景区域。这样，目的块编码的前 4 位已被确定为“0001”

下面继续推导后几位编码，即从“0001”这个大三角形逐步定位为标记为“1”的块位置。这是一个细化缩小的过程，且此过程正好与“0011”块细化缩小到原始块“00111000”块相对应，即在每一次划分中，如果原始块被分到“00”位置，则目标块应向“10”位置细化，如果原始块被分到“10”位置，则目标块则相反应向“00”位置细化。因此根据原始块“00111000”的末尾四位“1000”就可以推导出目标块的末四位应为“0010”。至此，块“1”的编码已被解析为“00010010”。

经过分析，上述推导过程同样可以抽象为一次异或位运算：

00111000⊕00101010=00010010。此位运算中的第二操作数的确定方式也类似 HEALPix 中的方法，根据所求邻接块所处的方向的不同而不同，现将这一规则总结如下：

1) 如果最终目标是求 1 号角对边方向的邻接三角形编码，即标记为“1”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“01”或“11”位，如果找到的是“01”，则从该位开始直到最后一位间的每两位均为“11”，如果找到的是“11”，则从该位开始直到最后一位间的每两位均为“10”，而更高位上均为“0”；

2) 如果最终目标是求 0 号角对边方向的邻接三角形编码，即标记为“0”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“00”或“11”位，无论找到的是“00”还是“11”，都从该位开始直到最后一位间的每两位均设定为“11”，而更高位上均为“0”；

3) 如果最终目标是求 2 号角对边方向的邻接三角形编码，即标记为“2”的邻接块，则其异或运算符右侧的第二操作数的确定方式为：对原块编码从低位向高位寻找第一次出现的“10”或“11”位，无论找到的是“10”还是“11”，都从该位开始直到最后一位间的每两位均设定为“01”，而更高位上均为“0”；

得到了异或右侧操作数之后，就可以通过异或操作完成图中块“0”、块“1”和块“2”的编码计算了：

- ✧ 块“0”编码：00111000⊕00000011 = 00111011
- ✧ 块“1”编码：00111000⊕00101010 = 00010010
- ✧ 块“2”编码：00111000⊕00000101 = 00111101

根据 6.1 节中的分析，其它相邻块的编码计算均可以看成这三个原子操作的组合，所以不再详述。

6.3.2 块内边界子块编码计算算法

初始多核环境下的并行交叉证认方法和第四章基于 MapReduce 的分布式交叉证认方法都需要对更高划分级别上的邻接小块进行快速编码推导，而这就需要块内边界小块编码计算算法。

仍以计算“00111000”块的 1 号角对边方向的邻接 HTM 原子小块为例，前面已通过基于位运算的同等划分级别下的邻接块计算算法计算出了这些小块所属的大计算块的编码为“00010010”，现利用 HTM 的编码规则推导其内紧邻下边缘的一行原子小块的 HTM 编码。如图 6-8 所示，其中的大三角形即为已推算出来的编码为“00010010”的 1 号块。

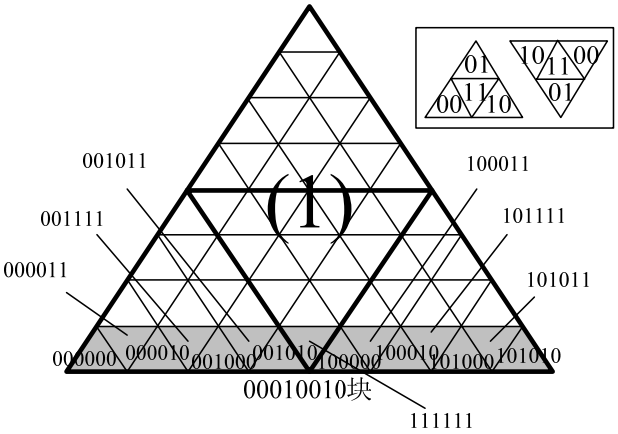


图 6-8 HTM 索引下块内边界小块编码计算

如果最终需要的边界数据宽度为 1，则只需要计算 00010010 块的最下方一行原子块的编码，即图中的灰色区域。根据 HTM 的编码规则，这些沿着下边缘的原子小块的后缀编码具有这样的规律：每两位可以任由“00”、“10”、“11”中的任一个组成，但是一旦为“11”后面的位也必须为“11”。

如果最终需要的边界数据宽度大于 1，并且为$2^m(m=1,2,3,...)$，则可把该问题转化为在比原子块划分层次低 m 层上的宽度为 1 的紧邻边界的块的编码推导问题；如果，宽度大于 1，且不为$2^m(m=1,2,3,...)$，也可以通过 HTM 的编码规则推算出每一个原子小块的编码，出于计算简便的考虑，也可以适当加宽边界宽度到$2^m(m=1,2,3,...)$，只是会引入较多的证认计算量。

6.3.3 实验结果

此实验选用的软硬件环境与 6.2.3 节中 HEALPix 邻域编码计算算法的实验环境完全相同。

实验内容为计算全天区8×4^8个 HTM 计算块中的每个计算块周围一圈的2^6个邻接 HTM 原子块的全部 HTM 编码，该计算过程实质上是次“同等划分级别下的邻接块编码计算”和$8\times4^8\times3$次“块内边界小块编码计算”的组合。最终程序运行结果显示：总共耗时 1.23 秒。

由此可见，本节设计的基于异或位运算的 HTM 索引下邻域编码计算算法确实能够很好地解决这一问题的效率问题，是高效交叉证认方法的重要保障。

6.4 本章小结

HEALPix 和 HTM 两种索引划分下相邻区域编码的计算算法是本文所提出的并行交叉证认方法成立的基础，同时它们也是诸如锥形查询等其它天文数据查询和天文数据处理工作的基础性操作。因此，这一操作能否具有很高的效率关系到很多天文数据服务的实现性能。本章在详细分析多核环境下并行交叉证认方法和 MapReduce 分布式集群环境下交叉证认方法对邻域编码计算的各自的具体需求后，归纳出了两个关键的原子操作：同等划分级别下的邻接块编码计算和块内边缘子块编码计算。然后在深入探究 HEALPix 和 HTM 两种球面索引的划分、编码机制的基础上，分别推导出了此二者的计算算法，并通过对推导过程的本质分析将较为复杂的同等划分级别下的邻接块编码计算算法转换成具有极高效率的基于异或位运算的编码计算算法。实验结果证明本章提出的 HEALPix、HTM 两种索引下的邻域编码计算算法均具有很高的效率，为高效交叉证认方法的实现奠定了基础，同时也在多种海量数据下的天文数据处理中有着重要的应用价值。

第七章 总结与展望

天文多波段交叉证认是天文数据联合查询的核心技术，同时也是应用数据挖掘、统计分析等方法进行天文数据分析前不可或缺的一步。经过交叉证认后形成的多波段或全波段数据蕴含了更多的可揭示天体物理本质的信息，是孕育天文新发现、新理论的摇篮。伴随着主动光学技术、薄镜面拼接技术、自适应光学技术等崭新技术的引入，天文数据正在急速地增长，研究如何在海量数据上实现高效率的交叉证认至关重要，它关系到当今大量尖端科技观测设备的巨额投入能否快速有效地转化为科学成果，关系到天文科学能否持续快速地发展。

本文所研究的基于位置信息的交叉证认是一种基础性交叉证认方法，其在各种数据集上的广泛适用性使之成为当前唯一可以在海量数据上实现的规模化自动证认方法，也是各国虚拟天文台构建数据访问服务时普遍采用的方式。本文在前人研究的基础上，针对多核处理器环境、大规模集群环境分别研究并实现了高效的交叉证认方法，并在解决交叉证认这一问题的主要性能瓶颈——频繁的数据I/O 操作方面取得了关键突破，真正使海量数据上的大规模交叉证认成为现实。

在多核环境下的交叉证认研究中，本文首先应用 HEALPix 伪二维球面索引方式在加快数据查询速度的同时实现了数据的区域划分，降低了证认计算的时间复杂度，然后应用本文设计的邻域编码快速计算算法全面地解决了交叉证认的常见问题——边界漏源问题，保证了证认结果的完全性。实验表明，与前人的做法相比，该方法对交叉证认计算的效率提升明显。此后，又针对该方法的最主要耗时环节——数据库查询操作，进行了优化，提出了可大量减少数据重复查询的全新的数据加载、计算流程——限制生长模型，并在该模型下提出了最大生长块的概念，它是并行程序任务分配调度的基本单元，既保证了较低水平的数据重复读取率，又全面地过滤了天文观测数据中大量存在的空白区域，从而使交叉证认的效率继续提高了 50%左右。

在此之后，为了继续突破关系数据库在处理海量数据中的性能瓶颈，同时也为了满足海量天文观测数据的存储需求，本文继续提出了基于 MapReduce 分布式并行计算模型的交叉证认方法，以文件数据库取代了关系数据库，使数据读取效率取得了质的突破。在算法设计上，使绝大多数的数据通信量完成在仅需执行一次的预处理阶段，尽量地避免了交叉证认计算过程中的节点间通信，最大限度地利用了 MapReduce 模型的特性，保证了证认过程的效率。实验证明，在上亿

条的数据量级上此方法的性能远远优于多核环境下基于数据库的并行交叉证认算法，并且随着集群节点个数的增多，其性能表现出了接近于线性的加速比，为实现在线实时交叉证认服务打下了基础，并为今后数据规模的继续扩张预留了空间。

值得一提的是，在研究高效交叉证认方法时所提出的基于位运算的快速邻域编码计算算法是本文的另一贡献。它不仅是高效交叉证认得以实现的一个基础性效率保证，也在诸如锥形检索等多种天文数据处理应用中有着重要的作用。

此外，本文研究的几种高效交叉证认方法以及邻域编码计算算法均打破了对单一索引方式的依赖，通过理论结合实验的方式已证明其在 HEALPix、HTM 两种当今最为常用的天文数据索引上均具有较好的适用性，保证了本文方法的广泛应用价值和现实可行性。

综上所述，本文在多核环境、分布式集群环境上研究的高效交叉证认算法为解决交叉证认在海量数据上的效率问题提供了可行的途径。但多波段交叉证认的设计与开发本身就是一个复杂的问题，在未来的研究工作中，还需要深入展开以下方面的探索：

研究解决基于数据挖掘、概率统计等更复杂交叉证认方法在海量数据上的效率问题，争取更高的证认精确度。本文基于位置信息的交叉证认存在着一定的精度局限，仪器的探测能力有限，而望远镜的分辨能力同样有限，从而导致所有的观察数据在位置信息上都存在不同程度的误差。所以要想精确地对不同星表间的数据进行匹配，不可避免地要利用概率统计、数据挖掘、机器学习等更加复杂的方法，针对不同数据进行更有针对性地具体分析。但当前这些方法受限于特定的目标星表，且效率较低，因此进一步研究具有广泛适应性和高效率的精确交叉证认方法是未来工作的重点和难点。

研究并实现可在线访问的交叉证认服务系统。本文当前设计的几种交叉证认算法的可行性虽已被实验证明，但要构建出具有实际应用价值的交叉证认系统还有许多工作要做，包括多种数据源间的格式转换、多层系统架构的实现、对多种交叉证认扩展方法的支持、与数据查询系统的整合等，这些是下一步交叉证认系统设计中的关键。

此外，基于交叉证认计算中具有的数据间独立性，可为更加复杂、更加专用的交叉证认方法提供基于数据划分的自动并行化方法，由此可进一步设计开发出支持多种交叉证认方法扩展的自动并行化系统，从而可以使天文学家仍然专注于串行的交叉证认方法的研究，进一步满足天文科学研究的需要。

参考文献

[1] 崔辰州，虚拟天文台和网格技术 [R]，网格战略研讨会，北京，2002.4

[2] 崔辰州，虚拟天文台——网格技术最好的试验场 [R/OL]，（2004-10-28）[2008-05-16].http://www.lamost.org/website/news/200410281339.html.

[3] Ken Ebisawa, Frangoise Genova, Robert Hanisch, The Astronomy Data Centre of the National Astronomical Observatory of Japan [R/OL], Report of the Evaluation Committee, (2008-2-28) [2008-11-24], http://www.nao.ac.jp/Report/ADC.pdf.

[4] 刘超，基于虚拟天文台的数据挖掘技术及其在银河系晕结构研究中的应用 [博士学位论文]，北京：中国科学院国家天文台，2008.1.

[5] Strauss, Michael A., Tyson, J. A., Sweeney, D., etc., LSST Observatory System and Science Opportunities, American Astronomical Society, 2010 (42): 216.

[6] J. Becla, K. T. Lim, Report From the First Workshop on Extremely Large Databases [J], Data Science Journal, 2008, (7): 1~13.

[7] The Two Micron All Sky Survey at IPAC [EB/OL], http://www.ipac.caltech.edu/2mass/.

[8] 李令坤，罗泽，阎保平，虚拟天文台中星表数据访问网格服务的设计与实现 [J]，计算机应用研究，2006（23）：125~128

[9] Brunner R., Djorgovski S., Szalay A. Towards a National Virtual Observatory [J], Virtual Observatory of the Future. Michigan, 2001: 343~372.

[10] 崔辰州，中国虚拟天文台系统设计 [博士学位论文]，北京：中国科学院国家天文台，2003.

[11] Astronomy and Astrophysics Survey Committee, Board on Physics and Astronomy, Space Studies Board, National Research Council. Astronomy and Astrophysics in the New Millennium. Washington, D.C.: National Academy Press, 2001

[12] International Virtual Observatory Alliance, http://www.ivoa.net/

[13] 张彦霞，多波段天体物理中的自动分类方法研究 [博士学位论文]，北京：中国科学院国家天文台，2003

[14] Patel J. M., Dewitt. D. J., Partition Based Spatial-Merge Join [C], Proceedings of ACM SIGMOD, 1996: 259~270.

[15] Huang, Y. W., Jing, N., Rundensteiner, E., Spatial Joins using R-trees:

Breadth First Traversal with Global Optimizations [C]. Jarke M, et al. eds. Proc. of the VLDB'97, San Francisco: Morgan Kaufmann Publishers, 1997: 396~405.

[16] 李立言，秦小麟，空间数据库中连接运算的处理与优化 [J]，中国图像图形学报 A 辑，2003.8（7）：732~737.

[17] T. Brinkhoff, H.-P. Kriegel, B. Seeger, Parallel Processing of Spatial Joins Using R-trees [C], Proceedings - International Conference on Data Engineering., New Orleans: IEEE, 1996: 258~265.

[18] G. Luo, J. Naughton, C. Ellman, A Non-Blocking Parallel Spatial Join Algorithm [C], Proc. International Conference on Data Engineering, San Jose: Institute of Electrical and Electronics Engineers Computer Society, 2002:697~705.

[19] Sutherland, W., Saunders, W., On the Likelihood Ratio for Source Identification [J], Monthly Notices of the Royal Astronomical Society, 1992(259): 413~420.

[20] D. J. Rohde, M. J. Drinkwater, M. R. Gallagher, etc., Applying Machine Learning to Catalogue Matching in Astrophysics [J], Monthly Notices of the Royal Astronomical Society, 2005(360): 69~75.

[21] Chen Cao, Applying the Support Vector Machine Method to Matching IRAS and SDSS Catalogues[J], Data Science Journal, 2007(6): 756~759

[22] Tamas Budavare, Alexander S. Szalay, Probabilistic Cross-Identification of Astronomical Sources[J], The Astrophysical Journal, 2008.5(679): 301~309.

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

[24] Nieto-Santisteban, M.A., Thakar, A.R., Szalay, A.S.: Cross-Matching Very Large Datasets [R/OL], (2006)[2008-04-16]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.3240&rep=rep1&type=pdf.

[25] Gray, J., Szalay, A., Fekete, G.: Using Table Valued Functions in SQL Server 2005 to Implement a Spatial Data Library [R/OL]. Microsoft Technical Report, MSR-TR-2005-122. Redmond, WA (2005) [2008-5-12]. http://research.microsoft.com/pubs/64532/tr-2005-122.pdf.

[26] Gray, J., Nieto-Santisteban, M.A., Szalay, A.S., The Zones Algorithm for Finding Points-Near-a-Point or Cross-Matching Spatial Datasets [R/OL]. Microsoft Technical Report, Redmond, MSR-TR-2006-52. WA (2006) [2008-5-15]. http://arxiv.org/ftp/cs/papers/0701/0701171.pdf.

[27] Gray, J., Szalay, A. S, Nieto-Santisteban, M.A., Heber, G., Rots, A.H.: There Goes the Neighborhood: Relational Algebra for Spatial Data Search [R/OL]. Microsoft Technical Report, MSR-TR-2004-32, Redmond, WA (2004)

[2008-06-06]. ftp://ftp.research.microsoft.com/pub/tr/tr-2004-32.pdf.

[28] Gray, J., Szalay, A., Budavri, T., Thakar, A.R., Nieto-Santisteban, M. A., Thakar, A.: Cross-Matching Multiple Spatial Observations and Dealing with Missing Data [R/OL]. Microsoft Technical Report, MSR-TR-2006-175, Redmond, WA (2006) [2008-06-13]. http://research.microsoft.com/pubs/64519/tr-2006-175.pdf.

[29] AstroGrid [EB/OL], http://www.astrogrid.org/

[30] Spatial Joins and Spatial Indexing Revisted [R/OL]. Technical report of AstroGrid, (2003-12-17)[2007-12-11] http://wiki.astrogrid.org/bin/view/Astrogrid/SpatialIndexing, 2003

[31] Clive Page, Comments on the XMATCH function in ADQL [R/OL]. Technical report of AstroGrid, 2004

[32] Rajendra Bose, Robert G. Mann, Diego Prina-Ricotti, AstroDAS: Sharing Assertions across Astronomy Catalogues through Distributed Annotation, Proceedings of the International Provenance and Annotation Workshop, Chicago, 2006 (4145): 193~202

[33] VizieR [EB/OL], http://vizier.u-strasbg.fr/.

[34] Simbad [EB/OL], http://simbad.u-strasbg.fr/.

[35] Aladin [EB/OL], http://aladin.u-strasbg.fr/.

[36] NED Batch Jobs [EB/OL], http://nedwww.ipac.caltech.edu/help/batch.html/

[37] Gao Dan, Zhang, Y.X., Zhao, Y.H.: Implementation of Cross-Matching on Very Large Multi-Wavelength catalogs [J], Astronomical Research and Technology, publications of National Astronomical Observatories of China, 2005

[38] 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—工具开发与算法研究 [博士学位论文]，北京：中国科学院国家天文台，2008

[39] Gao Dan, A System Integrated with Query, Cross-matching and Visualization [J]. SPIE The International Society for Optical Engineering, 2006 (6274): 14

[40] 崔辰州，指尖上的宇宙——虚拟天文台 [J]，中国科学院国际天文台，科学画报，2004(9): 34

[41] 程景全. 天文望远镜原理和设计——射电、红外、光学、X射线和射线望远镜（第1版）[M]. 北京: 中国科学技术出版社, 2003.

[42] David C. Brock, Understanding Moore’s Law: Four Decades of Innovation, Chemical Heritage Foundation, 2006.7

[43] Nichol, R. C., Connolly, A. J., Moore, A. W., etc., Computational AstroStatistics: fast algorithms and efficient statistics for density estimation in large astronomical datasets [C]. Proceedings of Virtual Observatories of

the Future. ASP Conference Series, 2001(225): 265

[44] Soto, A. Cansado, F. Zavala, Detection of Rare Objects in Massive Astronomical Datasets Using Innovative Knowledge Discovery Technology [C], Astronomical Data Analysis Software and Systems XIV, 2004.10.

[45] 崔辰州，虚拟天文台和网格技术，网格战略研讨会，北京，2002.4

[46] 大百科全书天文学编委会，中国大百科全书（天文卷）[M]，中国大百科全书出版社，1980.12

[47] Beautifully Detailed Supercomputer Simulations, Discover, USA, 2010

[48] Baker, John G., Centrella,Joan; Choi, Dae-Il; Koppitz, Michael; van Meter, James, Gravitational-Wave Extraction from an Inspiraling Configuration of Merging Black Holes, Physical Review Letters, 2006 (96), Issue 11, id. 111102.

[49] 科学家建巨型计算机识别黑洞 [J/OL]，发现和创新，2008(3)： 31，http://www.cqvip.com/qk/91878A/200803/26662193.html.

[50] Volker springel, Simon D.M.White, Adrian Jenkins, Carlos S.Frenk, Naoki Yoshida, Simulating the Joint Evolution of Quasars, Galaxies and their Large-Scale Distribution, 2005(435): 629~636.

[51] Iliev I T, Mellema G, Pen UL, Merz H, Shapiro PR, Alvarez MA, Simulating Cosmic Reionization at Large Scales I: the Geometry of Reionization, Mon.Not.Roy.Astron.Soc.2006 (369):1625~1638.

[52] Trac H, Cen R, Loeb A. Imprint of Inhomogeneous Hydrogen Reionization on the Temperature Distribution of the Intergalactic Medium[J]. Astrophysical Journal, 2008 (689): 81~84.

[53] 白春礼，“我国天文学取得三大进步，国际影响空前”，2009年国际天文年纪念大会，北京，2009.4.26.

[54] TOP500, SUPERCOMPUTER SITES, http://www.top500.org/

[55] John M. Vlissides，James O.Coplien，Norman L.Kerth，Pattern Languages of Program Design 2，Addison-Wesley Longman Publishing Co.，Inc., Boston, MA, 1996. 311.

[56] 陈国良，并行计算——结构、算法、编程（修订版） [M]，高等教育出版社，2003.8

[57] Barry Wilkinson, Michael Allen, Parallel Programming Techniques and Applications Using Networked Workstations and Parallel Computers, (2nd Edition) [M], Prentice Hall, 2004.3

[58] Barry F. Smith, Petter Bjorstad, William Gropp, Domain Decomposition: Parallel Multileve Methods for Elliptic Partial Differential Equations [M], Cambridge University Press, 2004

[59] Andrade, H. Kurc, T. Sussman, A. Saltz, J., Exploiting functional decomposition for efficient parallel processing of multiple data analysis queries [C], Parallel and Distributed Processing Symposium, Proceedings. International. 2003

[60] Ferreira, A., On Space-Efficient Alogorithms for Certain NP-Complete Problems [J], Theoretical Computer Science, 1993(120): 311~315.

[61] 于策，孙济洲等，一种应用于网格计算环境的任务调度模式[J]，计算机应用研究，2008.5（25）：1500~1503

[62] 朱福喜，何炎祥，并行分布计算中的调度算法理论与设计[M]，武汉：武汉大学出版社，2003

[63] Jeffrey Dean, Sanjay Ghemawat, MapReduce: Simplified Data Processing on Large Clusters [C], Google, Inc. the Proceedings of the 6th Symposium on Operating Systems Design and Implementation, 2004.12

[64] Amazon Elastic MapReduce [EB/OL], http://aws.amazon.com/elasticmapreduce/

[65] Cheng-Tao Chu, Sang Kyun Kim, Yi-An Lin, etc., Map-Reduce for Machine Learning on Multicore [C], NIPS conference 2006, 2006.12

[66] Grossman, R. L., Gu, Y., Data mining using high performance clouds: experimental studies using Sector and Sphere [C], Proc. 14th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2008), Las Vegas, NV, 2008

[67] Cardona, K., Secretan, J., Georgiopoulos, M., etc., A Grid Based System for Data Mining Using MapReduce [R], Technical Report TR-2007-02, The AMALTHEA REU Program, Summer 2007

[68] D. Gillick, A. Faria, and J. Denero, Mapreduce: Distributed computing for machine learning [EB/OL], 2008: http://citeseerx.ist.psu.edu/viewdoc/summary10.1.1.111.9204

[69] Hadoop [EB/OL], http://hadoop.apache.org/

[70] Lucene [EB/OL], http://lucene.apache.org/

[71] Nutch [EB/OL], http://nutch.apache.org/

[72] Dhruba Borthakur, the Hadoop Distributed File System: Architecture and Design [EB/OL], Hadoop Project Website, 2007, http://hadoop.apache.org/common/docs/r0.18.0/hdfs_design.pdf

[73] HDFS [EB/OL], http://hadoop.apache.org/hdfs/

[74] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, The Google file system [C], Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003.10.13, Bolton Landing, NY, USA

[75] Gary R. Bradski, Christos Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems [C]. Proceedings of the 2007 IEEE 13th International Symposium on High Performance Computer Architecture, 2007: 13~24

[76] Patricio F. Ortiz, Why Indexing the Sky is Desirable, Astronomical Data Analysis Software and Systems XII [C], ASP Conference Series, 2003 (285).

[77] 刘学富，基础天文学 [M]，高等教育出版社，2008.1

[78] Theodore Johnson, Dennis Sasha, The performance of current B-tree algorithms [C], ACM Transactions on Database System (TODS), 1993 (18): 51~101

[79] Clive G. Page, A New Way of Joining Source Catalogs using a Relational Database Management System [C], Astronomical Data Analysis Software and Systerns XII, ASP Conference Series, 2003 (295)

[80] Timos Sellis, Nick Roussopoulos, Christos Faloutsos, The R+-Tree: A Dynamic Indes for Multi-Dimensional Objects [C], Proceedings of the 13th International Conference on Very Large Data Bases, 1987: 507~518.

[81] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, etc., The R*-tree: an efficient and robust access method for points and rectangles [C], ACM SIGMOD Record, 1990.6 (19):322~331

[82] KuMar A, G. tree: a new data structure for organizing multidimensional data [C], IEEE transactions on knowledge and data engineering, 1994 (6): 341~347.

[83] B. Ooi, K. McDonell, R. Sacks, Spatial KD-Tree: an Indexing Mechanism for Spatial Database, IEEE Computer Society Press, 1987.10

[84] Kalpakis, K., Riggs, M., Pasad, M., etc., A System for Low-Cost Access to Very Large Catalogs [C], Astronomical Data Analysis Software and Systems X, ed. F. R. Harnden, Jr., F. A. Primini, & H. E. Payne, ASP Conf. Ser., 2001 (238), San Francisco: ASP

[85] Clive Page, Indexing the Sky [EB/OL], (2002-5-9) [2007-11-10] http://wiki.astrogrid.org/bin/view/Astrogrid/SkyIndexing

[86] HTM, Hierarchical Triangular Mesh, http://www.sdss.jhu.edu/

[87] Kunszt P. et al. Mapping the Sky using the Hierarchical Triangular Mesh [C], MPA/ESO/MPE Joint Astronomy Conference, 2000

[88] HEALPix, Hierarchical Equal Area isoLatitude Pixelisation [EB/OL], http://healpix.jpl.nasa.gov/

[89] Connolly, A. J., Genovese, C., Moore, A. W., etc., HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere [J], The Astrophysical Journal, 2005.4 (622): 759~771

[90] W. O'Mullane, A.J.Banday, K.Gorski, Splitting the sky – HTM and HEALPix [C], MPA/ESO/MPE Joint Astronomy Conference, Mining the Sky 2000

[91] Szalay, A.S., Large Databases in Astronomy [C], MPA/ESO/MPE Joint Astronomy Conference, Mining the Sky 2000

[92] GAIA, http://astro.estec.esa.nl/GAIA

[93] Giardino G. et al. 2000, Analysis of CMB foregrounds using a database for Planck [C], MPA/ESO/MPE Joint Astronomy Conference, Minning the Sky 2000

[94] David A. Bader, Varun Kanade, and Kamesh Madduri, SWARM: A Parallel Programming Framework for Multicore Processors [C], Proceeding of International Conference on Parallel and Distributed Processing Symposium 2007: 1~8

[95] Ren, Shangping: Nogiec, Jerzy, Developing concurrent applications on emerging multicore platforms, Proceedings [C] – CISIS 2008: 2nd International Conference on Complex, Intelligent and Software Intersive Systems, 2008: 632~637

[96] MPICH [EB/OL], http://www-unix.mcs.anl.gov/mpi/mpich

[97] 都志辉，高性能计算并行编程技术——MPI并行程序设计 [M]，北京：清华大学出版社，2000

[98] Toby Velte, Anthony Velte, Robert Elsenpeter, Cloud Computing, A Practical Approach [M], McGraw-Hill Osborne Media, 2009.9

[99] Michael Miller, Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online [M], Que Press, 2008.8

发表论文和科研情况说明

发表的论文：

本人在读博士期间以第一作者发表学术论文共计 4 篇，其中被 EI 检索 1 篇。

[1] Qing Zhao, Jizhou Sun, Ce Yu, Chenzhou Cui, Xiao Zhang, and Liqiang Lv, A Parallel Large-Scale Astronomical Cross-Matching Function, International Conference on Algorithms and Architectures for Parallel Processing (ica3pp) 2009, LNCS5574: p604~614, Springer (EI: 20093912332041)

[2] Qing Zhao, Jizhou Sun, Ce Yu, Chenzhou Cui, Jian Xiao, Xiao Zhang, Improved Parallel Processing Method for High-Performance Large-Scale Astronomical Cross-Match, Journal of Tianjin University (Engish Edition), 2010

[3] 赵青，孙济洲，于策，崔辰州，肖健，面向海量数据的并行天文交叉证认，计算机应用，2010

[4] 赵青，孙济洲，肖健，于策，崔辰州，刘旭，袁鳌，基于 MapReduce 模型的分布式天文交叉证认，计算机应用研究，2010

参与的科研项目：

[1] 国家自然科学基金项目——虚拟天文台环境下的海量数据存储和共享系统的研究（10778623）

[2] 国家自然科学基金项目——AST3 实时数据处理关键技术与系统，国家自然科学基金项目（10978016）

[3] 天津市科技支撑重点项目——面向广域网环境中数据密集型应用的高性能计算平台（09ZCKFGX00400）

致　　谢

首先诚挚的感谢我的导师孙济洲教授，他严谨的治学态度和科学的工作方法给了我极大的帮助和影响，使我受益匪浅，他的悉心指导使我得以完成实验室的科研工作和论文的撰写，在此向孙济洲教授表示衷心的谢意。

另外还要感谢实验室的于策、肖健两位老师，他们对于我的科研工作和论文都提出了许多的宝贵意见和建议，并对我的学习和生活给予了很多关心和照顾，使我的三年博士生活过得顺利而难忘。

本论文的完成还要感谢中国科学院国家天文台的崔辰州、张彦霞、刘超、何勃亮、杨阳等各位研究人员，他们不仅对我的科研工作提供了直接的指导与帮助，更使我在科研思维方式、团队协作等方面受益良多。

在实验室工作及撰写论文期间，吕立强、张啸、刘旭、袁鳌、汤善江、徐祯、曹玮等同学对论文中实验设计与测试等工作给予了热情帮助，在此向他们表达我的感激之情。同时感谢我的宿舍室友尹琳琳同学，她对我的科研工作和日常生活给予了很多关心和鼓励。

另外也感谢我的父母及朋友王鹏，他们的理解和支持使我能够专心完成我的学业。