计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2017.20170005
54(2):248-257,2017

天文大数据挑战与实时处理技术

杨 晨[1] 翁祖建[1] 孟小峰[1] 任 玮[1] 忻日辉[1] 王春凯[1] 都志辉[2] 万 萌[3] 魏建彦[3]
[1](中国人民大学信息学院 北京 100872)
[2](清华大学计算机科学与技术系 北京 100084)
[3](中国科学院国家天文台 北京 100012)
(yang_chen@ruc.edu.cn)

Data Management Challenges and Real-Time Processing Technologies in Astronomy

Yang Chen[1], Weng Zujian[1], Meng Xiaofeng[1], Ren Wei[1], Xin Rihui[1], Wang Chunkai[1], Du Zhihui[2], Wan Meng[3] and Wei Jianyan[3]
[1](*School of Information*, *Renmin University of China*, *Beijing* 100872)
[2](*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)
[3](*National Astronomical of Observatories*, *Chinese Academy of Sciences*, *Beijing* 100012)

Abstract In recent years, many large telescopes, which can produce petabytes or exabytes of data, have come out. These telescopes are not only beneficial to the find of new astronomical phenomena, but also the confirmation of existing astronomical physical models. However, the produced star tables are so large that the single database cannot manage them efficiently. Taking GWAC that has 40 cameras and is designed by China as an example, it can take high-resolution photos by 15 s and the database on it has to make star tables be queried out in 15 s. Moreover, the database has to process multi-camera data, find abnormal stars in real time, query their recent historical data very fast, persist and offline query star tables as fast as possible. Based on these problems, firstly, we design a distributed data generator to simulate the GWAC working process. Secondly, we address a two-level cache architecture which cannot only process multi-camera data and find abnormal stars in local memory, but also query star table in a distributed memory system. Thirdly, we address a storage format named star cluster, which can storage some stars into a physical file to trade off the efficiency of persistence and query. Last, our query engine based on an index table can query from the second cache and star cluster format. The experimental results show our distributed system prototype can satisfy the demand of GWAC in our server cluster.

Key words astronomy big data management; the ground-based wide-angle camera array (GWAC); two-level cache; star cluster; index table

摘 要 超大型天文观测技术的出现不仅能够让研究人员观测到新的天文现象,更能用于验证已有物理模型的正确性.这些最新天文成果的发现是建立在海量天文数据的近乎实时产生、管理与分析的基础上,因此给目前的数据管理系统带来了新的挑战.以我国自主研发的地基广角相机阵(the ground-based wide-angle camera array, GWAC)天文望远镜为例,15 s 的采样和处理周期都处于短时标观测领域的

收稿日期:2017-01-06;**修回日期**:2017-01-09
基金项目:国家重点研发计划项目(2016YFB1000602)
This work was supported by the National Key Research Program of China (2016YFB1000602).
通信作者:孟小峰(xfmeng@ruc.edu.cn)

世界前列,但却对数据管理系统提出了很多问题,包括多镜头并行输出数据管理、实时瞬变源发现、当前观测夜数据的秒级查询、数据持久化和快速离线查询等.基于上述问题,设计了分布式 GWAC 数据模拟生成器用于模拟真实 GWAC 数据产生场景,并基于产生的数据特性,提出一种两级缓存架构,使用本地内存解决多镜头并行输出、实时瞬变源发现,使用分布式共享内存实现秒级查询.为了平衡持久化和查询效率,设计一种星表簇结构将整个星表数据划分后聚集存储.根据天文需求特点,设计基于索引表的查询引擎能从缓存和星表簇以较小的代价对星表数据查询.通过实验验证,当前方案能够满足 GWAC 的需求.

关键词　天文大数据管理;地基广角相机阵;两级缓存;星表簇;索引表

中图法分类号　TP311.133.1

随着各种最新观测技术的出现,天文领域迎来了信息爆炸的时代,而该时代的第一波浪潮就是天文大数据的管理.进入 21 世纪,天文学已经进入了一个信息丰富的大数据时代,天文数据正在以 TB 量级甚至 PB 量级的速度快速增长.2000 年斯隆数字巡天(Sloan digital sky survey, SDSS)项目启动时,位于新墨西哥州的望远镜在短短几周内收集到的数据,已经比天文学历史上总共收集的数据还要多.到了 2010 年,信息档案已经高达 1.4×2^{42} B.不过,预计 2019 年在智利投入使用的大型视场全景巡天望远镜(large synoptic survey telescope, LSST)能在 5 d 之内就获得同样多的信息.在我国,郭守敬望远镜(the large sky area multi-object fiber spectroscopic telescope, LAMOST)和即将上线的地基广角相机阵(the ground-based wide-angle camera array, GWAC)等巡天项目每天都要产生海量的天文数据.

天文数据管理系统经历了从开始的基于文件系统的管理到目前基于关系数据库管理的发展阶段.早期限于观测设备,天文领域数据量不大,可以以文件的方式管理天文数据,仅支持一些简单的归档、排序和整理服务.随着观测设备升级,文件系统已不能满足当前数据规模的管理工作,因此天文领域开始使用关系型数据库管理数据.此时,天文研究者开始要求数据库能够按天文需求条件查找查询,并定制一些查询.但在可期的将来,超大型天文设备短时间产生(TB 级)和累计下的数据量(PB 或 EB 级)将超过关系数据库的管理能力,给当前天文数据管理带来巨大的挑战.

目前在国际上,美国的 SDSS 望远镜的采样周期是 71.72 s,每次采样的数据处理周期是 1 个月.美国设计的 LSST 采样周期是 39 s,每次采样的数据处理周期是 60 s.中国设计的 GWAC 不同于其他天文观测技术,由 40 台广角望远镜组成,单个观测夜中,所有望远镜要求同步地每 15 s 采样一次,并于下一次采样前将原始图像数据转化为星表数据,与此同时数据还要处于可查询状态,以支持短时标天文现象的发现.因此,无延迟的处理数据,对当前数据管理系统提出了新的挑战.

传统的关系型内存数据库,如 MonetDB[1],入库时间变动过大,极容易引起下一次数据来到后上一次数据还未写入完成,造成拥塞[2].而非关系型数据库,如 Kafka[3]+Hbase[4],实验表明单个镜头一次产生的数据量写入 Kafka 需要 2.7 s,不会造成数据拥塞,但写入 Hbase 需要 5 min 左右,并不能满足实时查询要求.

针对上述问题,本文提出一种两级缓存架构方案以支持无拥塞承接每个采样周期数据,且保证数据可查.结合 GWAC 背景,本文认为每个望远镜产生的数据之间相互独立,因此设计每个望远镜由一台服务器承接刚产生的数据,进行必要的处理后,如交叉认证(见 1.2 节),缓存入本地内存(一级缓存),并直接进行异常天文现象的识别.继而,每台服务器上的缓存管理器异步将一级缓存的数据写入二级缓存.二级缓存使用分布式共享内存实现,如 Redis cluster[5].可以看出,一级缓存的作用如下:1)快速承接每个望远镜产生的数据;2)快速进行异常检测;3)二级缓存故障后,保证数据的可靠性.两级缓存架构可以实现多镜头并行输出、异常天文现象发现、低延迟存储和秒级查询.

由于不能事先确定异常天文现象的发生,但异常发生时仍需要实现相关瞬变源数据的快速查询,以帮助天文研究人员快速定位重大科学发现.因此,本文将当前观测夜的所有数据全部缓存入二级缓存.在白天时,对二级缓存数据进行持久化操作.为了支持高效的离线天文数据分析并兼顾持久化效率,本文设计一种星表簇结构作为持久化数据的存储

模式.简单地说,通过一定的天文业务背景和策略,将二级缓存中的星表数据聚合后存入不同的物理文件中.这样做的好处是:1)天文查询的基本需求是对星亮度变化的查询,不同星之间很少做关联操作,因此将部分星聚集存储,能保证只访问整个数据集的部分,从而提高查询效率;2)比起单颗星存储而言,能有效降低管理整个数据集的开销(如文件元数据和文件索引个数).

对于不同的数据源,如二级缓存和持久化数据,查询引擎都必须支持且能联合查询.本文提出一种基于索引表的查询策略以支持快速查询.索引表除了记录星名,还记录星属性不变的部分(如位置信息)和统计信息,并动态更新.如果查询请求不需要检索随时间变化数据,如位置信息,那么查询索引表后直接返回;如需要随时间变化数据,则查询索引表获取满足条件星名集合,进而使用星名集合从二级缓存或持久化数据查询满足条件的星表数据.此时,星表簇结构可以保证查询引擎仅扫描持久化数据的某几个小子集.

为了验证上述方案的有效性,本文改写单镜头版 GWAC 模拟数据生成器[6]为分布式 GWAC 模拟数据生成器,可用于模拟多个望远镜同步产生星表数据.作者在一个小型的原型系统上实现了整个数据产生、存储和查询过程并进行了实验验证.实验结果表明,当前的设计方案能够支撑 GWAC 系统日常的业务需求.

本文主要贡献如下:

1) 设计分布式 GWAC 模拟数据生成器,能够更真实地模拟多个望远镜同步产生星表数据的应用场景;

2) 提出一种两级缓存架构的数据管理方案,针对性地解决多镜头并行输出、实时瞬变源发现和秒级查询等问题;

3) 针对当前观测夜星表数据持久化时间限制及离线查询效率的相应要求,设计一种星表簇的存储结构兼顾持久化与查询性能;

4) 设计一种针对索引表和星表簇结构的查询策略,能够尽可能少地访问持久化数据,或访问持久化数据的部分子集以提高查询效率.

本文首先介绍 GWAC 背景和面向星表数据管理的相关工作,然后重点介绍面向 GWAC 星表数据管理系统设计,最后给出系统验证与分析结果.

1 相关背景

1.1 GWAC 相机阵介绍

我国兴建中的地基广角相机阵 GWAC 由 40 台口径为 18 cm 的广角望远镜组成,每台望远镜配备 4 k×4 k 的电荷耦合器件(charge coupled device, CCD)探测器①.整个相机阵的天区覆盖 5 000 平方度,时间采样周期为 15 s.每个观测夜对固定天区目标的持继观测长达 10 h.从观测视场的大小和观测时间的采样频度上,地面广角相机阵在时域天文观测中具有特殊的优势.巨大的数据量和高速采样率,对数据的管理和处理问题提出极大的挑战.地面广角相机阵的星表数据指标是:1)星表数据每幅图像大约有 1.7×10^5 条记录,整个相机阵在 15 s 内共产生 6.8×10^6(数据量约为 1.3 GB)条记录,每晚约有 $2400\times40=96000$ 幅图,大约需要 2 TB 的存储开销;2)以 10 年为设计周期,GWAC 总计将产生 3~6 PB 量级的超大规模星表.

1.2 GWAC 天文数据处理流程

如图 1 所示,GWAC 相机阵将整个观测天区划分为 40 块,每块子天区由一个 CCD 负责采集数据,且所有 CCD 每 15 s 同步地产生一次数据.采集到的原始数据为图像,并经由预处理、点源提取(把光学影像转化为数字信号,形成星表数据)和星表天测定标(将一个星表中的星亮度校准到天文领域通用的标准下)等天文处理过程,转换为每颗星一行记录的星表数据.该星表数据对天文科研数据而言,最重要的 2 个属性是星的亮度和相对应的时间戳.根据

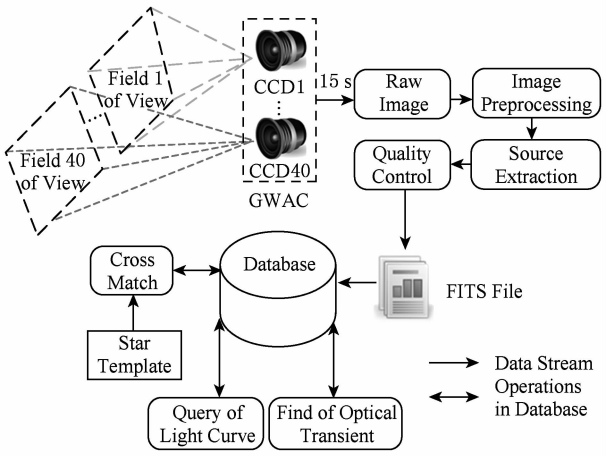


Fig. 1 Data processing workflow in GWAC

图 1 GWAC 天文数据处理过程示意图

① 如无特别所指,本文将使用专业名词 CCD 表示 GWAC 中的单个望远镜.

瞬时星亮度或变化规律的异常可以分析该星的异变,而该异变现象可以用于探知宇宙的变化和对已有物理模型的验证.根据长期星亮度的变化规律可绘制该星的光变曲线,以用于分析星的长时标的变化周期,如发现流浪行星.

从实时角度来看,持续产生的星表数据主要有以下3个特征:

1)多镜头并行输出.虽然每个CCD最终产生的星表数据量不大,但是40个CCD每隔15 s就会产生规模庞大的数据量.这些数据需要及时存储便于查询.

2)实时瞬变源发现.异常天文现象稍纵即逝,为了给天文科研人员留出足够的时间观测异常星,要求整个数据处理系统能够实时捕获异常星变化,并给予报警.

3)秒级查询.天文科研人员往往需要对瞬变源或疑似瞬变源的近期历史数据快速查询,以便综合分析该天文现象.

上述需求对后台的天文数据处理系统提出了巨大的挑战,要求系统能够快速响应,尤其对于当晚的星表数据而言要求能够做到快存快取.

从持久化角度来看,GWAC所有的历史数据都要进行持久化操作,以便离线状态下对星表数据进行光变曲线规律的分析和一定的数据挖掘工作.虽然为离线过程,但也要求查询过程要在合理的时间范围给予响应.

对GWAC数据管理系统的要求可总结为:1)高数据吞吐能力,所有相机阵15 s内产生的观测星表可用于查询的延迟时间控制在15 s以内;2)在数据高速采集下能够完成实时分析,面对持续不断的高密度海量星表的快速关联计算能力,即每个CCD每15 s产生的星表数据与模板星表相关联(交叉认证:将观测的目标星映射到模板星表的已知星的过程)形成光变曲线;3)每个观测夜的2 TB星表最晚完成持久化时间保证在下一个观测夜开始前;4)从长期存储的角度而言,管理系统需要有极强的海量数据管理能力,至少要能满足6PB数据的存储和离线查询能力.

1.3　天文数据管理系统的相关工作

目前国内外天文数据库的主要功能仍集中在电子化归档、搜索和下载等方面,且主要历经3个阶段[7].

1)兴起阶段,此时的天文数据库主要基于文件系统的数据存储.较为著名的有法国特斯拉斯堡的恒星数据中心CDS(centre de Données stellaires,即center for stellar data)的天文天体数据交互服务SIMBAD(set of identifications, measurements, and bibliography for astronomical data),利用计算机管理天文数据,能够将数据加以归档、排序和整理,并为全球星表提供交叉识别和文献目录检索功能.

2)关系数据库实现天文数据管理阶段,以提供星表服务的VizieR和SDSS为代表.到20世纪90年代末,SIMBAD服务已经无法满足更为复杂的查询需求,CDS又开发了更为强大的VizieR系统.VizieR底层依赖关系数据模型,支持基于ID和位置的搜索,且没有最大搜索半径的要求,具有较快的响应速度,但搜索的定制程度较低.此外,另一个专业的天文数据管理服务为斯隆数字巡天SDSS自主开发的数据库.SDSS的天文数据库Skyserver[8]是基于微软的SQL Server定制开发的,具有快速查询、批量下载、SQL检索和可视化图形界面等特点.这一阶段的天文数据管理开始在数据库的基础上定制了各种天文数据的科学应用,以满足天文数据特殊的检索需求.

3)即将到来的超大天文数据库阶段,以美国大口径全景巡天LSST和SKA(square kilometre array)为代表[2].一些新兴的天文领域如伽玛暴、超新星爆发对时域天文观测的要求更加迫切,直接导致天文数据量的爆发式增长.美国LSST设计每15 s记录3幅10亿像素级的图像,每晚收集的数据量大约15～30 TB,每3 d可巡天1次,预计2022年接受观测任务.澳大利亚SKA计划每秒产生的数据量大于12 TB,一天产生的原始图像为1 EB,预计从2020年开始第一阶段的建设.上述大型天文观测项目已对当前的数据管理框架产生了巨大的挑战,高吞吐量、大规模存储与快速的查找已成为了主要的问题.

值得一提的是,万萌等人[9]已对当前的GWAC数据管理场景进行了一定的研究工作,并提出了基于MonetDB数据库的管理方案.已开发出的GWAC数据生成器gwac_dbgen[6]能够模拟一个CCD连续产生的真实数据格式和量级.此外,基于该生成器的模拟数据使用SQL实现了MonetDB数据库内的交叉认证算法以避免数据的移动.但当累计数据规模较大时,MonetDB的扩展性较差且入库时间不够稳定.

2　面向GWAC的星表数据管理系统设计

结合GWAC天文大数据的特性和研究现状,本文采用两级缓存架构和星表簇模型,建立一个

高性能、可扩展的面向 GWAC 的星表数据管理系统. 该系统能够实现在 15 s 内存储多镜头并行输出的数据、瞬变源发现和提供秒级查询服务,此外星表簇模型有利于平衡持久化时间与离线查询效率.

如图 2 所示,该系统中和数据管理相关的部件主要包括 4 个部分:一级缓存管理、二级缓存管理、数据持久化和查询引擎.

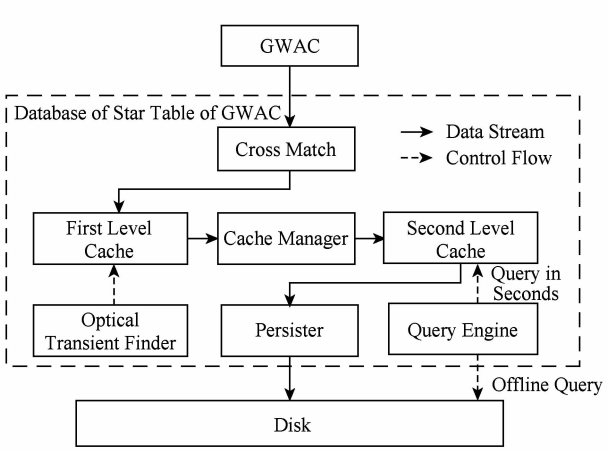


Fig. 2 Architecture of GWAC data management system

图 2 面向 GWAC 的星表数据管理系统架构

在文献[9]中,所有 CCD 产生星表汇入同一个 MonetDB 数据库后,再使用 SQL 对其进行交叉认证,从而产生一定的性能瓶颈. 本文设计的 GWAC 星表数据管理系统为分布式结构,一级缓存为分布式节点的本地内存,二级缓存为分布式共享内存. 当某 CCD 客户端发送星表数据进入系统后,系统会在某节点上创建对应客户端的接收端接收星表数据,直接进行交叉认证,然后将星表数据交由瞬变源发现模块进行异常检测,最后每个 CCD 对应的接收端将星表数据写入分布式共享内存中,供用户实现高速查询. 设计一级缓存的目的是:1)不同 CCD 产生的星表数据是无共享的(shared-nothing),因此处理就具备了并行性;2)瞬变源的发现与预警需要实时检测,因此需要获取数据后尽快在本地处理;3)为了保证分布式共享内存故障后数据高可靠,需要使用本地内存做缓存实现延时写. 设计二级缓存的目的是:1)天文研究者会在某颗星异常后,快速查询其最近的光变曲线以快速定位科学发现,但事先并不知道哪颗星会异常,因此需要将一个观测夜的数据缓存入分布式共享内存中供研究者快速查询;2)一级缓存容量是有限的,不足以承载一个 CCD 的整个观测夜数据.

在观测夜结束后,将当前观测夜的数据持久化到硬盘. 因为实际需求决定了星之间没有太多物理关联,因此几乎不做连接操作. 介于此,既可以将所有的星存储到一个物理表文件中,又可以将每个星单独建表. 单独建表的开销很高,但查询单颗星的光变曲线的效率很高,反之亦然. 本文设计一种星表簇存储结构将某些星聚合为大表,兼顾存储与查询. 因为星表簇结构的存在,对星光变曲线的查询,需要解析为对星表簇文件的搜索和查询.

2.1 分布式 GWAC 数据模拟生成器

由于 GWAC 系统目前正在调试阶段,并未上线. 为了更为真实地模拟 GWAC 数据对系统的压力测试,本文重写了 gwac_dbgen 模拟器为 gwac_dbgen_cluster,可以用于模拟多个 CCD 同步产生星表数据流,每个模拟的 CCD 以每 15 s 产生约 17 万行星表数据的速度产生数据(为了尽量还原真实场景,每次产生的星表数据行数不一定相等),每行数据 23 列.

如图 3 所示,为了保证所有 CCD 产生数据的同

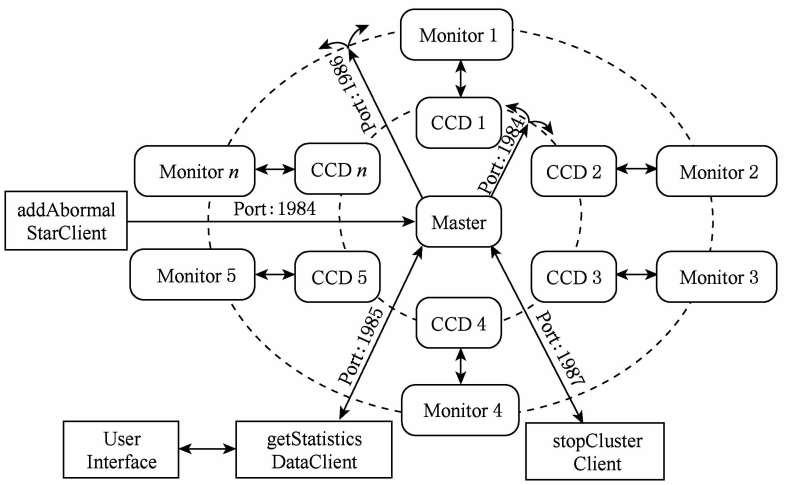


Fig. 3 Distributed GWAC data generator

图 3 分布式 GWAC 数据模拟生成器架构

步性,本文设计分布式 GWAC 数据模拟生成器为主从结构,时钟同步器由主节点控制.详细来说,生成器分为主节点、模拟 CCD(从节点)、监控端和客户端等 4 部分组成:1)主节点负责整体时钟同步、与客户端交互、整体集群控制和监控集群数据产生情况;2)模拟 CCD 负责产生星表数据,将数据发送一级缓存,并向主节点汇报当次数据产生情况;3)监控端负责开启、关闭模拟 CCD 并监控模拟 CCD 的性能情况向主节点汇报;4)客户端用于控制整体集群状态,目前设计的有 addAbormalStarClient 向某个模拟 CCD 加入异常星、getStatisticsDataClient 获取整个集群数据产生情况,并向 UI 节点输出展示和 stopClusterClient 用于正常或强制关闭集群所有服务.

2.2　两级缓存架构

如图 4 所示,每个 CCD 各自产生星表数据,通过各自的交叉认证模块匹配模板星表.随后将认证后的星表发送给一级缓存管理器,一般交叉认证模板和缓存管理器在同一物理服务器上,通过命名管道完成数据交互.由于要实时进行异常检测,缓存管理器将整个星表拆分为若干块,并行进行异常检测,如图 4 中 CCD 2 的一级缓存.检测完成后将星表写入二级缓存,供天文科研人员快速查询.综上,本文使用各子 CCD 数据在本地处理的策略,将整个大星表数据分而治之,在一级缓存中进一步拆分星表,进行实时异常检测和快速写二级缓存.以上即可完成多镜头输出数据不堆积,且实时进行异常检测的目的.

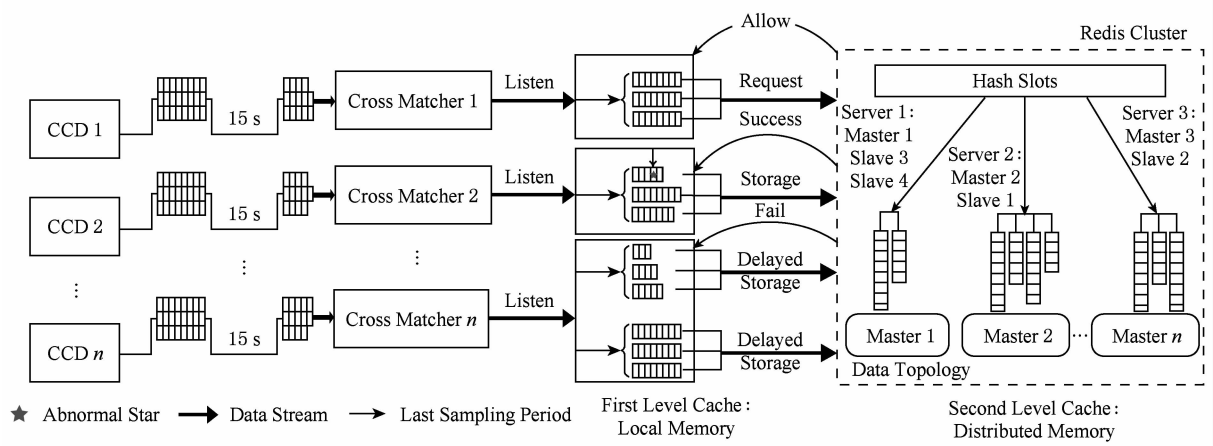


Fig. 4　Architecture of two-level cache

图 4　两级缓存架构示意图

本文使用 Redis cluster 作为二级缓存,Redis cluster 是典型的 KV(key-value)式缓存.从不同星的角度看,星表结构本身就是一个典型的 KV 结构,星名为 key,随后的属性为 value.从同一颗星的时间序列来看,可以用 list 结构存储.因此,如图 4 所示,随时间累积的整个星表是一个典型的 KV-list 结构,而 Redis cluster 支持该结构,并通过优化可以达到很好的性能.

Redis cluster 是一个无中心的集群结构,每个逻辑的 Master 节点,一般都会加入一个对应的 Slave 节点备份其数据以提供读服务,因此会形成很多的〈Master, Slave〉节点对.然而,若某节点对中 Master 或 Slave 故障后,虽然整个集群还可以工作,但会存在单点风险.介于此,本文向整个 Redis cluster 再加入部分 Slave 节点作为整个集群的后备,以进一步提高集群可用性.如图 4 中,Slave 1 和 Slave 4 都属于 Master 1 节点的从节点,若 Slave 2 节点(Master 2 节点的从节点)故障,Slave 4 节点会自动迁移为 Master 2 的从节点,以预防潜在的单点风险.需要注意的是,加入过多的后备 Slave 节点会提高网络和内存开销,使整体性能下降.

本文并不认为 Redis cluster 是可靠的,在 Redis cluster 故障后,恢复阶段的写操作会引起数据丢失.如图 4 所示,在 Redis cluster 恢复阶段,一级缓存管理器将未写完的上一个 15 s 数据挂起实现延时存储,以承接下一次 15 s 的星表数据.该策略可以实现 CCD 产生数据的及时处理和数据的高可靠性.

2.3　持久化操作

二级缓存用于快速处理当前观测夜的数据,并不能用于持久化所有观测夜的星表数据.随着当前观测夜结束,二级缓存上数据的“热度”自然下降,需要持久化到速度更慢的硬盘上,为离线分析做准备.

天文科研人员很大一部分的查询需求是关注某颗星的光变曲线,因此最为理想的存储方法是一颗

星一张物理表. 当需要查询光变曲线时,只需按星名找到该表,并按照给定的时间段扫描该表即可. 但这种存储方式的问题是小表太多,按照 GWAC 设计标准来说,40 个 CCD 能发现 680 万颗左右的星,也就对应着 680 万个小表. 过多的小表的随机写和管理代价过高,有时候甚至导致整个白天的时间不足以完成持久化操作,导致数据堆积在二级缓存中.

如图 5 所示,本文依托天文数据查询的特征发现多数查询会批量搜索某一范围内星的各类属性,因此按照星之间的距离特征作为 Hash 函数,将邻近星映射到同一个 Hash 桶内,构成星表簇. 再考虑到不同星表簇的查询热度不同,因此不同的星表簇内所聚合的星的个数不同. 例如,越容易查询到的星(异常星),所在的星表簇越小,查询就会越快,极端情况下会将单颗星作为一个星表簇. 反之,查询热度不高的区域存储为一个大的星表簇便于存储,降低数据管理的开销.

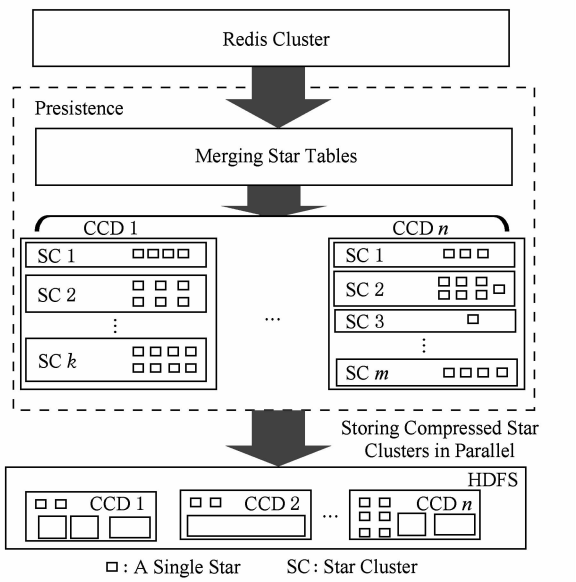


Fig. 5 Persistence of star tables

图 5 星表数据持久化示意图

实现上,本文使用 Spark[10] 和 HDFS[11] 实现持久化操作. 在将星名映射到不同的 Hash 桶之后,使用 Spark 对每个 Hash 桶中的星批量从 Redis cluster 读出,经过 Spark sql[12] 的建表操作,压缩后写入 HDFS. 为提高写性能,不同 Hash 桶的写操作是可以并行执行的. 需要注意的是,同一个 CCD 的星表簇数据需要写入同一个 CCD 的目录下,以便于快速检索. 从本质上看,为了查找某颗星的光变曲线,需要经过两级检索:1)确定该星所在的 CCD;2)确定该星所在的星表簇.

2.4 查询操作

基于星表簇的存储方式,查询操作并不能直接使用 SQL 完成,对于某些查询而言需要做一定程度的解析.

查询共分为 2 种类型:1)返回的结果不需要随时间变化的属性,如查询某位置范围内所有出现的星名;2)返回的结果需要随时间变化的属性,如返回某颗星的光变曲线. 通过收集天文查询需求,本文发现天文学家虽然关注光变曲线,如图 6,但是锁定查询哪条光变曲线的操作往往集中在不变的属性列上. 如查询某个范围内所有星的光变曲线,首先是查询该范围内的所有星名,其次是查询该星的光变曲线. 因此,无论结果是否需要随时间变化的属性数据,都需要先查询与时间无关的属性数据.

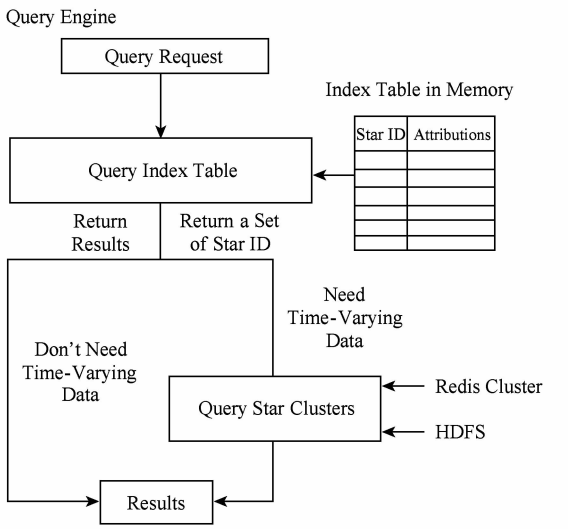


Fig. 6 Procedure of query operations

图 6 查询操作流程

如图 6 所示,本文合并所有 CCD 交叉认证时使用的模板表,并加入一些随时间变化数据的统计列,如亮度均值、方差、最大值和最小值等,构成一个逻辑上的索引表,因为该表很小,可以常驻内存. 一个查询请求到来后,先判断结果的属性列是否需要随时间变化的属性,如果不需要,查找索引表,直接返回;如果需要,查找索引表,找出满足要求的星名,然后查找原始数据. 这样做的好处是:1)对于结果不需要随时间变化的数据,查找索引表能避免扫描原始星表数据,提高查找性能;2)对于结果需要随时间变化的数据,查找索引表能最小化扫描原始星表数据的次数和范围,以提高查找性能.

对于采用缓存架构的本系统而言,原始星表数据存储于二级缓存(Redis cluster)和硬盘(HDFS)

上,查找条件有可能涉及到这 2 个数据源. 因此,就查询数据源而言,可分为以下 3 种策略对原始星表查询.

1) 仅从二级缓存查询. 查找索引表找出满足条件的星名集合;将该集合传入 Redis cluster 查询对应星的全部属性列;筛选符合条件的属性列.

2) 仅从硬盘查询. 查找索引表找出满足条件的星名集合;使用持久化的 Hash 函数将星名的结果集合映射到星表簇;对同一星表簇的星进行批量查询并筛选符合条件的属性列;对于不同的星表簇可以并行查询;最后合并不同星表簇的结果.

3) 从二级缓存和硬盘查询. 并行从 2 个数据源按照上述的策略查询,最终将结果合并.

3　系统验证与分析

3.1　实验准备

本文在 Ubuntu 14.04, Hadoop 2.6.0, Spark 1.6.2 和 Redis 3.2.5 平台上建立了验证系统,采用 10 台物理机构建 Hadoop, Spark, Redis cluster 和分布式 GWAC 数据模拟生成器集群. 对于 Hadoop, Spark 和分布式 GWAC 数据模拟生成器集群都使用相同的物理机作为主节点,剩下的 9 个物理机作为从节点(模拟 9 个 CCD),此外 Spark 使用 standalone 模式进行资源管理. 对于无中心结构的 Redis cluster,每个物理机上开启 4 个 Redis 节点进程,共 40 个节点,并且另加入 8 个 Redis slave 节点作为整个 Redis 集群的冗余备份.

整个物理集群为同构环境,每台物理机使用英特尔 i7-4790 处理器、32 GB DDR3-1600 内存和 1 块 500 GB 7200 SATA 硬盘,节点之间采用万兆以太网互联.

3.2　实验结果分析

为验证 GWAC 星表数据管理系统的有效性,实验开展了如下工作:1)分布式 GWAC 数据模拟生成实验;2)写缓存效率;3)持久化效率;4)查询效率.

1) 分布式 GWAC 数据模拟生成

由于产生的数据需要缓存进 Redis cluster 中,因此本集群环境的内存容量不可能容纳当前观测夜的所有数据量(约 3 TB). 经过测试,现有集群环境能模拟 9 个 CCD 同步拍摄 300 次(相当于 1.25 h)的整个过程. 实验结果表明,模拟数据产生过程很稳定,单个星表的产生时间约为 3 s,低于真实 CCD 产生一副图像需要的 15 s,因此可以用来模拟 GWAC 数据产生过程. 单个模拟星表的星个数在(175567, 175675)之间不等,每个星表大约 33 MB 符合真实场景,300 次共产生 87 GB 星表数据.

2) 写缓存效率

如图 7 所示,9 个 CCD 单次交叉验证且并行写 Redis cluster 的时间维持在 1.35 s 以下,平均耗时 1.08 s,在整个写缓存过程中时间没有出现显著的抖动. 因此,两级缓存结构的设计方案能够满足 GWAC 实际需求,各 CCD 产生的数据不会出现堆积现象. 此外,9 个 CCD 的星表数据写入 Redis cluster 300 次共消耗内存 269 GB. 消耗的内存量大于总共产生的数据量,原因如下:①所有 Master 节点存储原始数据,Slave 节点备份产生的星表数据,总共需要消耗 178 GB 内存空间(较产生的原始数据需要增加 2 列交叉认证数据,因此实际存储数据量大于 2×87 GB=174 GB);②另加入的 8 个 Slave 节点在集群正常工作时,作为某个 Master 节点的备份需要消耗 36 GB 内存空间;③Redis cluster 维护整个数据结构,如指针、元数据等,需要消耗 55 GB 左右的内存空间,约占所消耗总空间量的 20%.

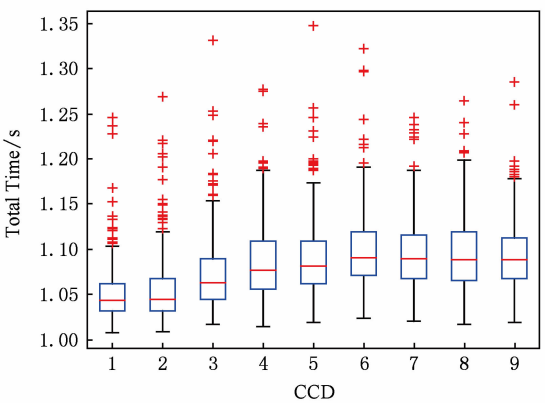


Fig. 7　Time consumed by cross match and writing to Redis

图 7　交叉认证和写 Redis cluster 总时间的箱线图

3) 持久化效率

本节实验对已写入 Redis cluster 的 9 个 CCD 的星表数据进行持久化操作,并测试效率. 为了发现持久化时间和星表簇之间的关系,本文分别测试了如图 8 横坐标所示的 8 种不同的星表簇个数,如 90 代表将 9 个 CCD 的星表数据划分入 90 个星表簇中,其中一个 CDD 平均有 10 个星表簇.

如图 8 所示,随着星表簇的增多,持久化所需要的时间越来越长,主要原因是:①簇内星表数据减少

导致随机写的代价升高;②簇个数的增多导致创建簇的代价升高.上述现象导致星表簇个数和持久化时间的正相关关系.当星表簇个数达到 9 000 时,持久化时间为 7.9 h,继续增加星表簇个数会导致持久化时间的继续增长,过长的持久化时间已经不能满足白天进行持久化的基本要求.介于此,本实验不再继续增加星表簇个数,并认为每个 CCD 的星表数据划入 1 000 个星表簇是当前实验集群能接受的最大值.当星表簇个数在 90～900 之间时,持久化时间为 11～52 min,该时间范围是可容忍和接受的,因此下文将对上述已持久化好的星表簇进行查询,以评估查询效率.

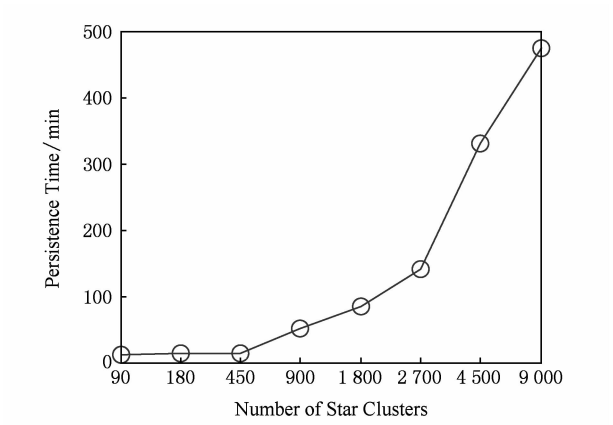


Fig. 8 Time consumed by the persistence of star tables sampled by 9 CCDs

图 8 9 个 CCD 星表数据持久化为不同数目的星表簇所需的时间

4) 查询效率

本节实验实现 2.4 节的查询策略,并测试对 Redis cluster 和星表簇的查询效率.

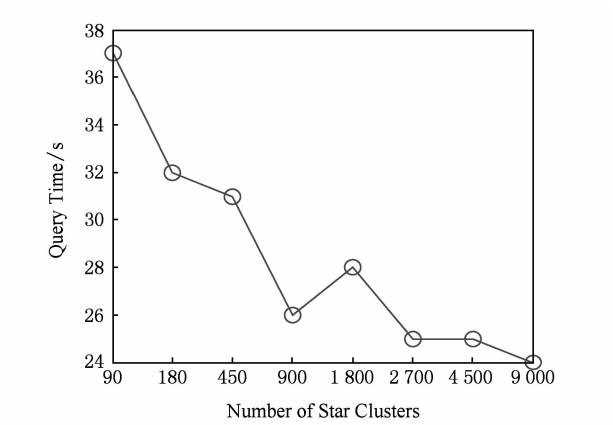


Fig. 9 Time consumed by the query of 20 light curves under the different number of star clusters

图 9 在不同的星表簇数目下批量查询 20 颗星的光变曲线所需的时间

实验随机选取 20 个星名(对不同数量星表簇的测试都使用这 20 颗星),并查询这 20 颗星在 1.25 h 之间的光变曲线.当查询 Redis cluster 时,查询时间维持在 4～5 s 之间.如图 9 所示,当查询星表簇时,查询时间也以秒为单位,且整体趋势是随着星表簇个数的增加,查询所需时间越短.这种现象的主要原因是星表簇内星的减少,导致每次读取的数据量更少,扫描效率更高.

随着星表簇个数的增加,持久化效率和查询效率呈现不同的性质,因此就实验结果而言本文认为 900 个星表簇是较好的一个平衡点.

4 结 论

针对 GWAC 天文大数据的特性和面向该类数据的管理挑战,本文提出了一套有针对性的数据管理架构和系统原型.其中包括:1)分布式 GWAC 模拟生成器;2)两级缓存架构;3)基于星表簇的存储策略;4)基于索引表的查询策略.通过实验验证,目前设计的原型系统能够实现实时存储 GWAC 每个采样周期的数据、实时瞬变源发现,秒级查询响应,持久化当前观测夜数据和快速的离线查询.未来的挑战主要在于提高查询效率和减小管理持久化数据的代价,实现对星表簇拆分和合并操作.

参 考 文 献

[1] Boncz P, Grust T, Van Keulen M, et al. MonetDB/XQuery: A fast XQuery processor powered by a relational engine [C]//Proc of the 2006 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2006: 479-490

[2] Wan Meng. An application research of column store MonetDB databaseon GWAC large-scale astronomical data management [D]. Beijing: National Astronomical of Observatories, Chinese Academy of Sciences, 2016 (in Chinese)
(万萌. 列存储 MonetDB 数据库在 GWAC 海量天文数据管理的应用研究[D]. 北京:中国科学院国家天文台, 2016)

[3] Apache. Kafka [OL]. [2016-12-30]. http://kafka.apache.org/

[4] Apache. Hbase [OL]. [2016-12-30]. http://hbase.apache.org/

[5] Redislabs. Redis [OL]. [2016-12-30]. https://redis.io/

[6] Wan Meng. Gwac-dbgen [OL]. [2016-12-30]. https://github.com/wan-meng/gwac_dbgen

[7] Jian L I, Cui C Z, Bo-Liang H E, et al. Review and prospect of the astronomical database [J]. Progress in Astronomy, 2013, 31(1): 1-16

[8] SDSS. Skyserver [OL]. [2016-12-30]. http://skyserver.org/

[9] Wan Meng, Wu Chao, Wang Jing, et al. Column store for GWAC: A high-cadence, high-density, large-scale astronomical light curve pipeline and distributed shared-nothing database [J]. Publications of the Astronomical Society of the Pacific, 2016, 128(969): 114501-114516

[10] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets [C] //Proc of USENIX Conf on Hot Topics in Cloud Computing. Berkeley, CA: USENIX Association, 2010: 1765-1773

[11] Apache. Hadoop [OL]. [2016-12-30]. https://hadoop.apache.org/

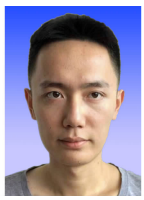
[12] Armbrust M, Xin R S, Lian C, et al. Spark sql: Relational data processing in spark [C] //Proc of the 2015 ACM SIGMOD Int Conf on Management of Data. New York: ACM, 2015: 1383-1394

Yang Chen, born in 1987. PhD candidate. His main research interests include science data management and performance bottleneck quantification.

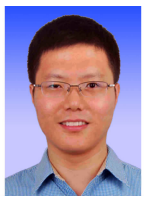
Weng Zujian, born in 1992. Postgraduate student. His main research interests include data stream computing and energy efficient computing.

Meng Xiaofeng, born in 1964. PhD, professor at Renmin University of China. CCF fellow. His main research interests include data fusion and knowledge fusion, big data management for new hardware, big data real time and interactive analysis, and big data privacy management.

Ren Wei, born in 1990. PhD candidate. His main research interests include database theory, data storage, and software engineering.



Xin Rihui, born in 1993. Postgraduate student. His main research interests include science data management and performance bottleneck quantification.



Wang Chunkai, born in 1981. PhD candidate. His current research interests include data stream computing and distributed systems.

Du Zhihui, born in 1970. PhD, associate professor. Senior member of IEEE and CCF. His main research interests include high performance computing, cloud computing, energy efficient computing and big data analysis.

Wan Meng, born in 1983. PhD, engineer. Her main research interests include astronomical big-data architecture, column-store databases, and computing system analysis.

Wei Jianyan, born in 1965. PhD, researcher, PhD supervisor, the leading scientist of SVOM astronomical satellite. His main research interests include space astronomy, active galactic nuclei, gamma burst observation, etc.