王若曈,王建民,黄向东,等. MICAPS4 服务端系统架构设计. 应用气象学报,2018,29(1):1-12.
DOI: 10.11898/1001-7313.20180101

MICAPS4 服务端系统架构设计

王若曈[1)*] 王建民[2)] 黄向东[2)] 董一峰[2)] 龙明盛[2)]
[1)](国家气象中心, 北京 100081) [2)](清华大学软件学院, 北京 100084)

摘 要

MICAPS4 体系采用客户端/服务器的系统架构,其中服务端系统是 MICAPS4 的重要部分,利用分布式存储与分布式计算技术,构建可容纳 10^2 TB 量级的气象实时数据,千万数据总量,面向数百并发用户的服务器集群系统。MICAPS4 服务端系统在国内率先实现全部气象实时数据由文件到数据库、从集中式系统到分布式系统的迁移,该系统自 2015 年起在全国推广使用。在海量气象数据和大量用户并发访问的环境下,表现出很高的稳定性和优越的读写性能,同时具有便捷的扩展性和可维护性。MICAPS4 服务端系统分为分布式存储系统、分布式前处理系统、站点实况轮询系统、查询服务器系统和监控系统 5 个子系统,分布式存储子系统为 MICAPS4 客户端提供了近实时数据的高速随机与顺序读取服务,分布式前处理系统利用对等分布式架构实现了海量气象实时数据的流式计算,站点实况轮询系统实现了跨系统的实况数据异构副本的同步功能,查询服务器系统利用多线程服务器技术实现了 MICAPS4 客户端的实时计算请求,监控系统利用部署于每个节点的探针实现监控信息的主动上报。

关键词: MICAPS4; 大数据; 分布式存储; 分布式计算; 实时数据

引 言

中国气象局自 1994 年起组织人机交互气象信息处理和天气预报制作系统(MICAPS)[1]开发,截止到 2007 年,发布了第 1、第 2、第 3 版,并在中国气象局各级业务部门中广泛应用。随着气象数据规模持续高速增长,MICAPS3 自 2010 年起开始面临严峻的性能和存储压力,主要表现在海量气象数据解析、访问缓慢,集合预报产品及新型观测数据难以应用,MICAPS3 系统框架难以满足业务发展需求。

在海量气象数据场景下,存储系统面临访问与查询缓慢的问题。以 MICAPS3.0 为代表,目前国内外气象数据存储技术仍多以基于目录树形结构的文件系统进行组织。而传统文件系统往往难以承受每日百万级的文件数量增长,且文件目录的树形结构不能很好满足预报员对数据进行按序访问的需求。据统计,在现有的基于文件系统的天气预报系统中,当系统存储数据文件数量达到 2000 万时,仅服务器端文件定位需耗时 500 ms,按序访问时间将更长,无法很好地满足应用需求。传统关系型数据库虽然具有排序和索引能力,但其固定的关系表模式使系统设计十分复杂,其磁盘存储结构也限制了其有序访问的优化。因此,无论文件系统还是传统数据库技术的存储和查询方式都无法很好地满足气象数据的高性能查询。

为解决上述问题,中国气象局于 2013 年启动了 MICAPS4[2]的开发工作,采用跨层优化的软件工程理念,研发了 MICAPS4 服务端系统。针对海量气象实时数据多类型、高维度、弱模式的特性,利用多源异构数据的弹性表模型,在国内率先实现全部海量气象数据从文件系统至非结构化分布式数据库的彻底迁移。

MICAPS4 服务端系统由分布式存储系统、分布式前处理系统、站点实况轮询系统、查询服务器系统、监控系统5个子系统组成。本文首先简要介绍

2017-07-25 收到, 2017-12-01 收到再改稿。
资助项目: 中国气象局“2015 年山洪地质灾害防治气象保障工程建设”
* 邮箱: wangruo02@mails.thu.edu.cn

系统总体架构,然后从每个子系统的设计动机、应用场景入手,较详细剖析每个子系统的关键技术和系统设计。

1 MICAPS4服务端系统总体设计

1.1 应用场景

1.1.1 存储系统需求

基于MICAPS4客户端对于气象实时数据的应用特点,MICAPS4存储系统需要具备以下特征:作为海量实时数据的高速缓存使用,至少可以容纳上千万个海量实时数据,但不需要存储历史数据;系统可能存在大量的并发读取用户,且短时间内数据会出现爆发性写入。存储的数据基本都是5维度以上的多维数据,且数据维度不固定。由于MICAPS4客户端对于气象实时数据的查询方式绝大多数是针对非结构化数据的高并发、只读性查询,经过理论和试验的分析,最终选用了键-值(key-value)分布式数据库作为存储系统。

1.1.2 海量文件式数据到达

数据到达行为特征差别大,实时气象数据种类繁多,并具有非常大的数量和容量。以文件形式到达的数据主要包括确定性模式数据、集合预报数据、卫星数据、雷达数据、气象算法产品、预报产品等多种类型数据,每类数据的单个数据大小、每日数据总量、每日数据总量、单个数据处理时间、数据到达频率等有很大的差异性。

气象实时数据到达时间、到达顺序具有不确定性。尽管从长时间统计结果看,同一类数据到达时间有规律可循,但在多种因素共同作用下(网络可用带宽变化、通信链路异常、数据生产者策略调整、设备故障),每一类数据的到达都具有时间抖动、乱序、流式等特性。

数据到达即可见特性。当某一个数据满足了解析、计算、输出的前置条件时,处理系统必须立刻启动相应的计算流程,尽可能快速地将该数据处理为MICAPS4客户端可视化所需的最终产品。因此,设计了文件式海量气象实时数据的分布式前处理系统以应对上述应用需求。

1.1.3 海量报文式数据到达

对于地面、高空、海洋、重要天气报、闪电、环境监测等报文式到达的数据,无法延用类似模式数据的前处理系统的流程对这些数据进行处理。主要原因在于:①原始数据存储模型与为查询而优化的存储模型间的不一致。为满足预报员最常见的地面填图、高空填图、闪电填图、海洋填图等查询需求,较好的策略是同一观测时间的站点的部分物理量集合组织为一个BLOb(binary large object,二进制大对象)数据块整体,便于高效在地图上观看全站点的显示效果。但这样的存储模型很难适用于原始报文的写入需求,如果原始存储模型与查询存储模型全部使用BLOb数据块组织,每到达一条新的报文涉及一次读操作、一次合并操作和一次写操作、按照全国55000个自动气象站计算,每个数据块要被覆写55000次,严重影响数据库性能,因此,必须设计面向写和读操作的不同的数据存储模型。②数据产生时间与使用时间的时间错位特性。由于站点类型数据报文原始存储模型与查询存储模型的异构冗余性,造成了两个模型数据跨系统副本时间上的不一致。这样需要使用轮询的策略,定时将站点的物理量元组信息加工成全站点信息,该加工过程存在一定的时间延迟。轮询频率高,两个存储模型间的数据不一致时间就短,反之轮询频率低,两个存储模型间的数据不一致时间长。

因此,为满足报文式原始数据的处理和存储需求,应当设计同处理文件式数据不同的、基于轮询的数据加工系统。

1.1.4 数据库直接访问与查询服务器间接访问

MICAPS4客户端对于数据的访问方式,可以直连数据库,也可以通过查询中间代理服务器访问。若采用后者,会带来大量额外的开发部署工作量,而且为确保查询性能,查询服务器也必须使用分布式的架构。如果采用直连数据库的数据访问模式,则需要将查询服务器的一些功能下沉到客户端的接口层。为了部署和运维的简便,最大程度确保客户端的查询性能,在对MICAPS4客户端提供直接访问数据库模式的同时保留了查询服务器,同时查询服务器提供了用户数据写入和一些分布式存储无法提供的实时计算功能。

1.2 系统总体架构

图1给出了MICAPS4服务端系统的总体架构图。对于文件式数据,模式、卫星、雷达数据由CIMISS(China Integrated Meteorological Information Sharing System,全国综合气象信息共享平台)的CTS(China Telecommunication System,气象通信系统)向MICAPS4分布式前处理系统分发,经前

处理系统实时解析后,将最终的可以被 MICAPS4 直接可视化显示的产品数据写入到分布式存储集群中。而地面、高空数据等报文式数据解报后,数据存储在关系数据库中,利用 MUSIC 接口(Meteorological Unified Service Interface Community,气象数据统一服务接口),站点数据轮询系统将这些数据加工地面填图、高空填图等格式后,写入分布式存储集群中供预报员查看。预报员科研型算法通常部署于独立的算法服务器上,通过查询服务器读写分布式存储集群中的数据。查询服务器还提供了读写数据的统一接口。所有服务器上配置监控探针,采用定时策略向 CIMISS 的 MCP(Monitoring and Control Platform,业务监控系统)监控服务器上报自身的健康状况。

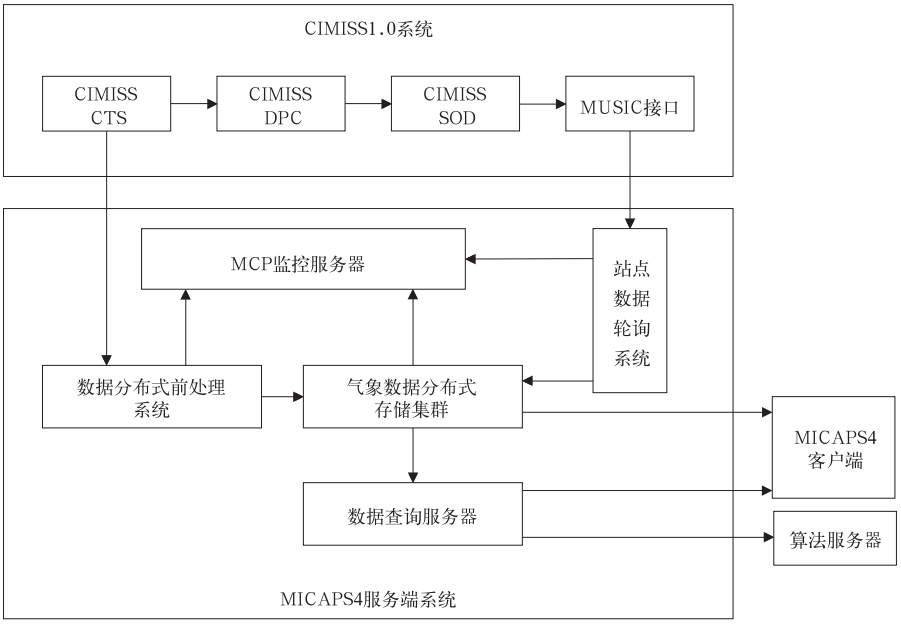

图 1　CIMISS-MICAPS4 服务端系统架构

Fig. 1　CIMISS-MICAPS4 server system architecture

2　海量气象实时数据分布式存储系统

2.1　关键技术选型

2.1.1　现有技术分析

使用传统关系数据库进行存储时,可以通过多字段的表结构描述多维度数据。传统关系数据库采用多种方式进行文件组织[3],然而无序的存储方式使其针对有序读取时,必然要付出排序的代价或更新索引的代价。在实际应用中,数据维度的多样性也使得关系数据库的模式难以设计,无法使用关系数据库作为解决方案。有学者提出了空间数据库[4]、时间序列数据库等新型数据库以应对具有空间关系、时间关系的数据,但这些技术仅仅从某一维度或几个维度对数据进行排序等特殊处理,不具备更强的通用性。

以气象预报行业为例,部分系统提出采用传统文件系统进行多维数据的存储,并在上层提供远程存取服务[5]。实际应用中使用文件系统时,往往以不同维度作为目录,构建出一个树形结构存储。每种维度作为目录树的一层内部节点,数据文件作为树的叶子节点。为了形成树形结构,需要人为规定数据文件维度的层次关系。这种方式简化了系统设计,数据文件的存储交由服务器的文件系统完成。然而由于文件系统本身并未考虑不同文件的逻辑顺序,对于检索维度上的有序访问,只能通过获取某一目录下所有文件,进行手动排序,即对叶子节点进行排序。当对其他维度进行排序时,则需要过滤不同中间节点的叶子节点,并进行合并和排序。当文件数量达到百万、千万级时,文件定位的速度将出现显著下降。虽然基于 SAN (storage area network:存储区域网络) 和 GPFS(general parallel file system,通用并行文件系统)的高性能气象数据存储集群架构,能够很好解决海量存储、容灾备份等特点[6],然

而并未对数据有序获取进行优化。

欧洲中期天气预报中心采用MARS(Meteorological Archival and Retrieval System,气象数据归档和检索系统)进行气象数据的管理[7],国家气象信息中心、国家卫星气象中心也分别建立了自己的存储系统[8-9],但是这些系统更多面向全数据集的存档查阅需求,也未对图形化预报平台随机/顺序访问做更多优化。我国早期的MDSS系统[10]尽管使用了数据库技术进行数据存储,但对于非结构化数据而言,实际存储于数据库中的内容仅为元数据,因此,仍然无法真正避免由于小文件数目增长带来的文件系统性能压力。

有学者提出使用HDFS代替传统文件系统,但由于其中心节点的元数据管理方式,HDFS并不适用于海量小数据文件的存储,且未考虑文件之间的顺序关联。Dong等[11]提出了在HDFS上高效排序和存取小文件的方法,试图解决上述问题。其将需要存储的原始文件和相关的描述文件合并为一个大文件进行存储,并使用固定大小的索引文件进行数据定位。通过该方法既减少了HDFS中心节点管理的文件数量,又将原始文件和与其相关的描述文件有序保存,加速了用户检索与下载原始文件的速度。然而,该方法仅仅考虑了数据的一种维度,无法很好地扩展到多种维度的快速存取,并且这种基于文件合并的HDFS存储方式,除非添加额外的分布式缓存中间件,否则无法满足数据持续写入的应用需求。

文献[12]提出使用REDIS(REmote DIctionary Server,远程字典服务器)分布式内存数据库与HDFS(Hadoop Distributed File System,Hadoop文件系统)相结合的方式解决海量小文件存储问题。这种方式弥补了HDFS的压缩文件无法实现追加写入的缺点,但REDIS基于文件大小的自动合并方案忽略了文件之间的联系,未对文件的按序遍历进行更多优化。

2.1.2 技术选择

同文件系统相比,数据库系统针对小数据一般会设计实现相当完善的读写缓存机制,从而大幅度提升性能。由于气象实时数据单个数据大小通常都小于5 MB,因此,首选数据库系统作为MICAPS4存储系统。

由于气象数据的使用以读取为主,没有强一致性要求,因此,分布式物理集群架构的系统将单台机器的存储、计算、网络读写压力分配到多个节点上,从而可以显著提升数据读取的效率。

传统关系型数据库等方案并不适合保存气象非结构化实时数据,因此,系统选型确定在非关系型分布式数据库上,通过理论以及试验的分析,最终确定使用Cassandra(开源分布式NoSQL数据库系统)作为MICAPS4气象实时数据存储系统。

Cassandra是一个基于键-值的点对点分布式系统,适合作为多维数据空间结构的实现[13],这同气象数据多维索引键值结构相呼应。此外,MongoDB(基于分布式文件存储的数据库)、HDFS(Hadoop分布式文件系统)、Hbase(面向列的分布式开源数据库)也是较为成熟的分布式存储系统,但MongoDB不适合存储非常大容量的数据,在气象数据处理领域,也有利用HDFS和HBase进行气象数据处理和存储的方法[14-15],且性能明显好于传统的处理方式。

HDFS仅支持单向索引,同时不支持追加写入,因此,无法用在MICAPS4实时数据写入和双向按序检索的场景。HBase在系统设计方面有很多与Cassandra的相似之处,利用HBase进行气象实时数据存储近些年也有一些较好的试验结果[16],但HBase主从架构同Cassandra的点对点架构相比更加复杂,HBase不采用哈希方式随机分布数据,需要手动指定初始化的数据分布,需要主节点记录每个机器上的数据位置,有潜在的节点过热和主节点阻塞的隐患。由于Cassandra在存储具有多维空间特点的海量小数据方面具有显著的优势,因此,采用Cassandra作为实时气象数据存储的实现方案。

2.2 数据模型与数据分布

在Cassandra分布式数据库中,每类数据各自对应一个列族,类似于关系型数据库的表,即T639,GRAPES_GFS及单站雷达分属不同的列族。每张表的行被称作列族的行键值,类似于该类数据的某一种物理量的磁盘路径。对于模式数据而言,有多少种物理量,一张表就有多少行。列名类似文件名,并且使用字典排序的方式,最先到达的数据,也就是最旧的数据位于表的最左列,最新的数据追加写入到表的最右列。因此,旧数据不停地从最左端向右滚动删除,新数据不停地从最右端追加写入。一张表的行数相对固定,而列数则取决于该类数据的存储时效。通过行列相交,可以得到这个数据的“值”,也就是这个数据的真实数据块。

在数据库的设计中采用了 3 副本的方案,这是性能和可靠性相对较优的一个配置。Cassandra 通过对每个行键值进行哈希计算,得到该行的所有数据将被放置在某 3 个物理节点上,因此,同一行数据将被存储在同一节点上,在另外两个物理节点上也有备份。如 T639 的 850 hPa 温度场,17062008.003 这个数据最终将被写入到节点 1,3,5 这 3 台服务器上。当 MICAPS4 客户端向任何节点查询这个数据时,通过相同的哈希计算,就可以得到该数据位置,从而返回该数据。这样的方式可以分散网络和存储的压力,且数据的查询性能及可靠性得到提升。

3　海量气象数据分布式前处理系统

3.1　关键技术选型

3.1.1　现有通用技术分析

前处理系统的特点符合流式处理特性,业界多采用消息队列与流式处理相结合的解决方案[17-19]。其中 Kafka(一种高吞吐量的分布式发布订阅消息系统)以其支持分布式、良好性能、与大数据生态系统紧密结合等特点被广泛使用。文献[20-23]对其他常用流式处理框架进行了总结。这些系统往往提供自动数据分区的分布式协议、动态资源调度、流式任务的拓扑图定制等功能。然而为了支持这些高级功能,这些系统需要在容错、任务分配方面牺牲一些性能,并且大幅度增加了系统的运维成本。当面对如气象数据这种数据量大、计算复杂的应用时,往往还需要分布式计算框架,如 MapReduce,Apache Spark 等[24]。

此外,在实际气象业务中需要考虑更多阶段,如被动式数据接收、数据处理任务的自动实时触发、并发调度策略、任务分布化、系统模块的跨平台与跨语言特性等。

3.1.2　被动式数据接收

由于原始数据的到达时间不确定、到达顺序混乱、数据偶尔丢失等特点,采用被动的数据接收设计,将极大的简化数据接收模块以及后续数据解析模块的设计。反之如果采用数据主动采集方式,则会带来极为复杂的前处理系统设计,因为需要对数据可用时间进行猜测,对原始数据目录进行轮询,对迟到/缺失数据设计重试算法,对数据产生速度超过网络带宽等各种异常情况进行人为处理等。因此,主动式数据采集同被动采集相比,系统复杂度高,维护难度大,难以保证数据的实时性,同时还会带来额外的性能开销。

3.1.3　基于嵌入式 FTP 和数据触发器的设计

通信系统大多采用 FTP 客户端进行数据推送,采用被动的数据接收,需要前处理系统自身集成相应的数据接收模块。常规的 FTP 软件只能进行数据接收,并不能在数据接收完成的瞬间触发其他任务,不能保证实时数据到达即处理的要求[25]。为保证数据到达后立刻启动解析任务,最大程度确保系统的简单性,前处理系统采用了嵌入式 FTP 服务器的方案,在前处理系统主进程启动后,即可启动 FTP 服务器线程,该 FTP 服务器具备“钩子”功能,通过捕获 FTP 协议中的文件写入结束或者文件重命名的指令,立刻在毫秒时间内触发针对该文件的异步解析线程,而针对海量数据同时爆发性到达场景,由于解析线程异步启动,因此,既可以保证数据的实时解析,又不影响新数据的接收,同时也可以实现数据的并行处理,从而提升性能。

3.1.4　解析线程池

解析系统对于数据解析的过程和异步,这样同一时间内可能会有大量的解析任务并发运行,如果不加控制,有可能对系统运行的稳定性造成影响。如果几十个、上百个文件同时到达,有可能导致服务器的 CPU 及内存等资源消耗殆尽,造成严重的系统灾难。在实时解析系统设计中,利用解析线程池保证该过程的稳定、高效。根据处理数据类型的不同以及系统的 CPU 核数、内存总量等参数,解析线程池可被设置不同的线程数阈值,达到该阈值后,新解析任务将等待,避免过度消耗系统资源。

3.1.5　对等分布式前处理计算系统设计

根据原始数据到达的特点分析,模式数据在某些特定的时间段具有爆发性到达的特征,卫星雷达数据则呈现出大批量匀速流式到达的特征,在每日的很多时段中,数据产生的速度已经超过了单台服务器对于数据处理的最大速度,需要采用水平扩展的结构,利用服务器集群、分布式计算的方案提升数据的流式解析性能。为确保系统的简单性、可伸缩性,前处理系统被设计为 1 台或多台服务器组成的集群,每台服务器地位对等,即所有服务器都具有完全一致的软硬件配置,任何数据被分发到任何一台服务器上都可以被解析。同自动化的任务分配算法不同,前处理集群的分布式计算采用手动的任务预配置划分方案,即根据专家先验知识,预先指定数据

类别同服务器的对应关系。这样做可以保证前处理系统对于流式数据的本地化计算，如果利用负载均衡器进行文件随机推送，则一些后续的计算将不得不跨节点完成，带来系统复杂度和性能的开销；另外可以保证系统维护的简单，巡检某一类特定数据的处理日志时，只需要访问一台服务器即可。由于前处理系统的重要性，所有硬盘、电源、网卡、交换机都作了冗余，尽可能降低故障造成的影响。

3.1.6 流式计算特性

对于常规计算场景，采用嵌入式带有数据触发功能的FTP服务器及解码线程池，即可完成数据的实时计算。

而在一些场景下，当前数据的处理同先前到达的数据分片序列具有依赖关系，如从累积降水到分段降水的计算等，这些场景对解析系统的流式计算功能提出了较为复杂的设计要求。经向风和纬向风分量矢量合成计算，模式全球区域网络拼接，风云四号等气象卫星数据实时乱序分片重组，集合预报成员到齐实时检测等场景均对前处理系统提出了相似的技术要求。对于分段降水计算、风场矢量合成计算、模式网格拼接、卫星数据分片组装等场景，可以形式化描述为顺序到达的 N 个文件，前$(N-1)$个文件不输出，第 N 个文件到达时才会输出，且输出结果依赖于前$(N-1)$个文件集中式计算的结果，这种情况采用中间结果序列化存储及嵌入式数据统计组件的技术路线实现，确保满足数据最终计算的前置条件时，立即对先前的中间结果反序列化并执行相应的计算流程。而对于集合预报数据的前处理，由于运算复杂度极高，单一的服务器集中式CPU计算需要相当长的处理时间，因此，可以描述为当集合预报全部成员到齐后，立刻启动相应的分布式计算任务，这种场景采用中间结果序列化存储、嵌入式数据统计组件、消息服务器及分布式计算的技术路线实现。

3.1.7 集合预报分布式计算模块

为实现集合预报数据的快速流式计算，需要利用Cassandra集群和数据分布的特性，在解析集合预报数据的同时，启动部署在Cassandra集群上的分布式计算系统，将集合预报的算法分布在多个节点上同时进行，并将最终结果输出到Cassandra数据库中。MICAPS4前处理系统采用Spark进行集合预报产品数据的计算和加工。

3.1.8 集合预报成员个数实时记忆统计模块

MICAPS4客户端对于集合预报数据使用方式决定了只有当所有成员到齐之后才会进行集合预报产品的生成。因此，在系统设计中，所有物理量成员的原始数据文件经解码后，首先持久化在本地文件系统中，通过实时的成员个数记忆统计模块，在所有成员到齐的瞬间，对先前的数据序列进行反序列化及合并，写入分布式存储的原始数据表中，为后面的分布式计算做准备。

3.1.9 消息服务器

集合预报分布式计算需要的原始成员数据通过前处理系统可以写入到分布式存储的原始数据表中。但分布式计算系统需要得到启动计算信号才能开始执行，而Hadoop，Spark等开源分布式计算框架均没有原生的远程启动接口，因此，引入消息服务器实现数据的实时分布式计算。当前处理系统将成员数据合并，并输出至分布式数据库后，立刻向消息服务器发送计算就绪的消息，分布式计算系统一旦收到新的计算就绪消息，立刻将该消息取回并解析，然后启动相应的分布式计算程序。

3.2 前处理系统设计

图2为前处理服务器集群同通信系统数据推送服务，以及同分布式存储服务器的集成示意图，并详细描述了1台前处理服务器的内部设计。

如图2所示，通过预配置的任务分配，通信系统将不同的数据分发至4台前处理服务器，这4台前处理服务器构成对等的分布式计算集群，各自具有完全相同的数据处理能力。集群通过主从备份的内部交换机将最终生成的MICAPS4格式数据写入分布式存储中，避免写入时的网络流量对外部网络造成影响，此外，每台前处理服务器的网卡也是主从备份的。

图2的解析服务器模块表示一台前处理服务器DPC(data processing center，数据处理中心)的数据处理流程，通过嵌入式数据接收模块接收通信系统的实时数据，利用数据触发器，实现解码器的实时动态加载。解码线程池用于实现多任务的并发解码，并对当前最大解码线程数进行控制，一个数据解析结束后，利用嵌入式的负载均衡软件模块在存储服务器连接池中随机选择一个连接进行最终的数据写入。文件清除器、管理员服务端接口都作为DPC的内部线程，在主进程启动后一直运行。

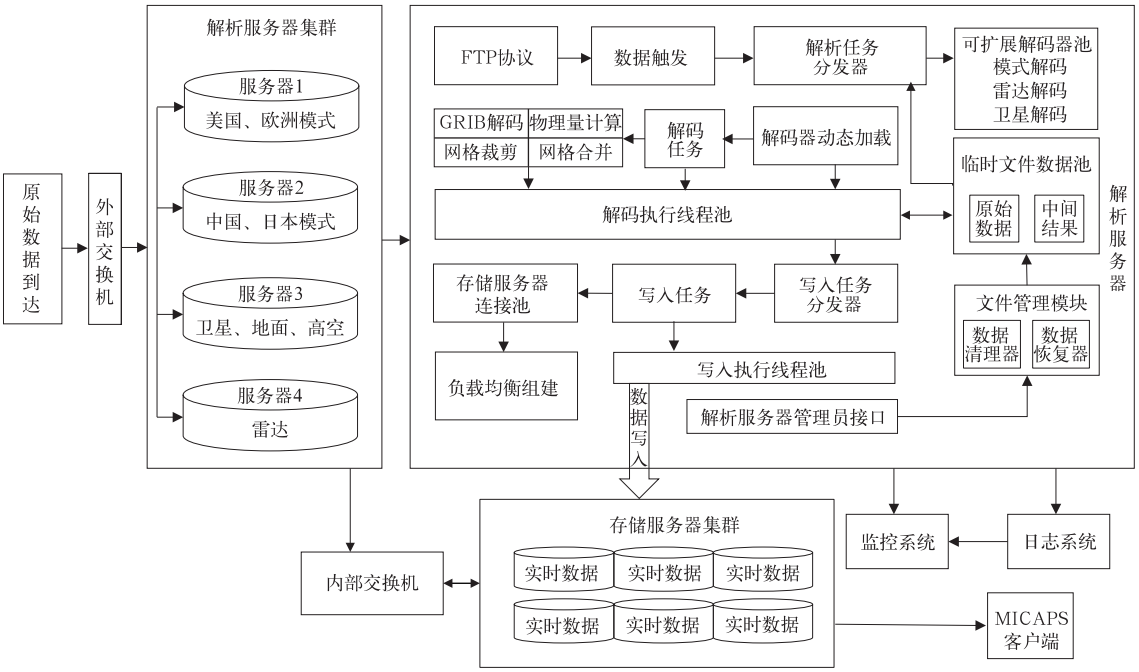

图 2　数据前处理系统

Fig. 2　Data pre-processing system

4　站点实况轮询系统

4.1　应用场景描述

从站点报文到达特点以及 MICAPS4 客户端的查询需求看,站点数据的处理场景可以归纳为以下几个方面:①某个观测时间对应的绝大部分报文会在该时间几分钟之后的某段时间大批量集中到达,只有少部分报文到达的时间较晚,即数据的到达呈现先密集、后稀疏的现象。②尽管所有站点数据的到达几乎都体现了先密集后稀疏的特点,但对于不同类型的数据,数据密集到达时段的开始出现时间,持续时长,存在较大的差异性。③对 MICAPS4 用户而言,临近时间数据为热数据,久远时间的数据为冷数据,即预报员更加关注临近数据。对于热数据,预报员希望新到达的报文可以尽快被查看。

4.2　关键技术选型

4.2.1　面向多种类型数据的冷热轮询策略

每次站点轮询任务的执行都对应一次定时作业的启动,LINUX 的 crontab 以及 Java 的 Jcrontab 和 Quartz 等框架都可以简单方便地设置任务启动的时间或者均匀的执行时间间隔。然而对于不同类型数据,针对冷热数据不一致的轮询算法策略,目前并没有成熟的框架可以直接实现。因此,站点实况轮询系统必须支持多种不同轮询策略的扩展,每种策略可以容纳多种相似特征数据的轮询,由于轮询策略算法设计复杂性较高,没有现成的定时作业框架可以使用,必须对轮询系统的策略模块进行定制化开发。

4.2.2　高并发异步轮询线程池设计

根据不同数据不同轮询策略,在不同时间可能要求分别启动大量的轮询任务。如果采用同步的轮询方式,前一次启动的轮询任务在下一个任务时间点如果没有结束,就可能影响后面时刻任务的正常执行。类似前处理系统解析线程池的设计,站点实况轮询系统也采用设置了最大线程并发数阈值的线程池模块,所有轮询任务异步启动,并发执行,超过线程阈值的轮询任务将在线程队列中等待,待其他轮询任务结束运行后再执行该任务。

4.2.3　接口检索与数据库直接检索互相备份

站点实况轮询系统目前通过 CIMISS-MUSIC 接口对站点数据进行轮询。由于站点实况数据的重要性,为防止 MUSIC 服务器故障或检索超时造成的轮询失败,轮询系统增加了数据库直接检索的备份机制:当通过 MUSIC 接口检索某一类数据失败时,会自动切换数据库直接检索的方式重试,从而确保数据的可靠性。

4.3 站点实况轮询系统的设计

图 3 为站点实况轮询系统同站点数据关系型数据库,以及同分布式存储服务器的集成示意图,并详细描述了轮询系统的设计。

同 DPC 前处理系统不同,轮询系统通常 1 台服务器即可,但为了站点实况数据的可靠性,通常配置 2 台轮询服务器同时启动,用任务调度器 1 和任务调度器 2 分别表示轮询服务器和备份轮询服务器,两台服务器的配置是完全一致的。在实际系统中,也可以将前处理系统和站点实况轮询系统部署在相同的物理服务器上。

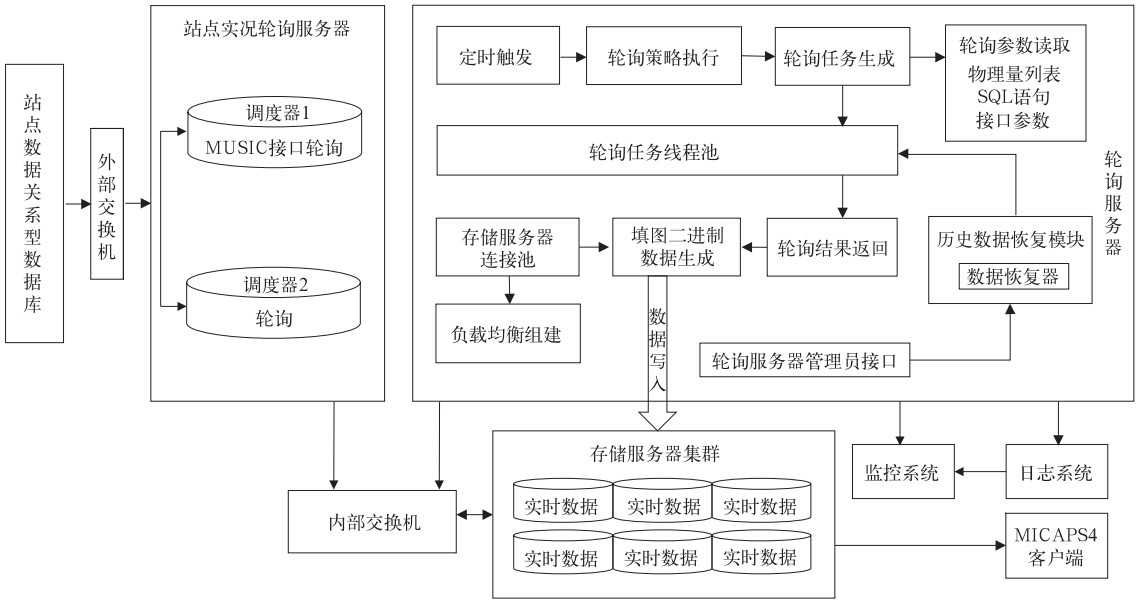

图 3 站点数据轮询系统

Fig. 3 Station data polling system

图 3 的轮询服务器模块表示了 1 台站点轮询服务器的数据处理流程,通过每分钟触发的轮询信号,遍历所有的轮询策略,并筛选出当前时间每一种轮询策略中要轮询的全部数据列表,通过配置信息的加载,生成 MUSIC 接口轮询信息或者 SQL 查询语句形式的轮询信息,进而构建轮询任务,并在轮询线程池中启动这些任务,主动向 MUSIC 接口或站点数据关系数据库中获取对应的数据。轮询线程池用于实现多任务的并发轮询,并对当前最大轮询线程数进行控制,一个数据轮询结束后,将生成可以被 MICAPS4 客户端直接可视化显示的二进制数据(如地面填图),轮询服务器利用嵌入式的负载均衡软件模块在存储服务器连接池中随机选择一个连接进行最终的数据写入,类似前处理服务器的写入过程。

5 查询服务器

为了最大程度的简化系统设计,同时发挥分布式存储的性能优势,MICAPS4 客户端采用可直接访问数据库的数据获取方式。

但在某些场景下确实需要查询服务器完成某些功能,包括数据下载需求、MICAPS4 客户端的实时计算需求以及少量数据写入请求等。如果数据通过直连数据库的方式下载,可能会出现少数用户消耗大量的服务器 CPU 和网络资源,严重影响 MICAPS4 客户端的使用,需要构建查询服务器满足非 MICAPS4 客户端的数据访问需求。此外,对于模式剖面、模式 $T\ln p$、模式单点时序图、实况时序图等类型的查询,查询结果同 MICAPS4 用户的桌面交互操作密切相关,这样的查询请求无法预先生成相应的结果,而 Cassandra 数据库本身只提供基本的数据存储与访问,不提供计算功能,要满足这样的需求需要在分布式存储与客户端之间增加查询服务器,保证 MICAPS4 客户端的数据实时计算需求。此外,对于少量数据的写入的情况,需要查询服务器实现灵活的写入权限控制,保护数据库中的关键数据不被破坏,文献[26-27]也采用了类似的设计。

5.1　关键技术选型

5.1.1　嵌入式 HTTP 服务器

有多种现有技术可以响应用户读写请求,如 HTTP, ICE, Google gRPC 服务等。其中 Google gRPC 属于较新服务,尚无成熟的应用;而 ICE 采用长连接的方式,适合用于大数据量的下载;HTTP 作为无状态协议的代表,有丰富、成熟的高并发应用案例,适合于单个数据量较小的访问场景,同时 HTTP POST 和 HTTP GET 请求也适合多种类型的查询参数的传递。由于气象实时数据绝大多数的单个数据大小都在 5 MB 左右的范围,因此,通过 HTTP 构建查询服务器是一个比较好的方案。Tomcat, Jetty, Web Logic 等都是可靠的 HTTP 应用服务器软件,Tomcat, Web Logic 属于偏重量级的应用服务器,而 Jetty 既可用于独立应用服务器,也可用于嵌入式 HTTP 服务器,嵌入式服务器可以将服务器作为程序组件启动和停止,共享相同的日志文件,便于部署,管理员巡检和故障诊断。因此,选用 Jetty 服务器作为查询服务器的对外访问接口。

5.1.2　查询数据的序列化

查询服务器的重要功能之一是为用户读取数据提供途径,而气象实时数据绝大多数都是非结构化二进制数据,因此,需要设计用户请求/应答的数据序列化方案。常见的数据序列化技术包括 XML(Extensible Markup Language,统一标记语言)、JSON(JavaScript Object Notation)、Google Protocol Buffer 以及各种编程语言自带的对象序列化软件包等。由于查询服务器将面对不同类型的编程语言,因此,必须选择与语言无关的序列化方式。XML,JSON 虽然通用,但只针对文本类型数据,将二进制数据转换为 JSON 字符表示需要额外的性能和网络开销。Protocol Buffer 是由 Google 研发,与编程语言无关的一种高效的二进制数据序列化方案,支持多种数据类型的封装,适合用于应答结果的表示。而用户下载数据的请求通常较为简单,可基于 HTTP(HyperText Transfer Protocol,超文本传输协议)协议 GET 方式实现。对于实时计算的场景,例如模式剖面计算,用户请求和应答都可以使用文本表示,因此,选用 JSON 作为数据序列化方案,用户请求的传递采用 HTTP POST。

5.2　查询服务器的设计

图 4 和图 5 描述了用户利用查询服务器从分布式数据库中下载数据、实时计算以及数据写入流程。

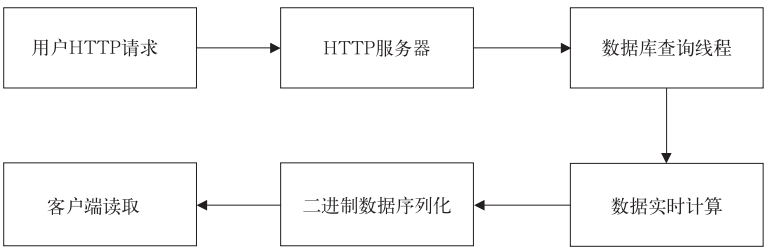

图 4　查询服务器的数据检索和实时计算

Fig. 4　Data retrieval and real-time computing of data query server

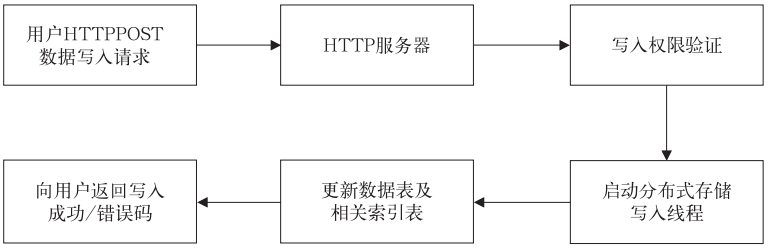

图 5　查询服务器的数据写入

Fig. 5　Data writing of data query server

首先客户端(MICAPS4 或者其他客户端)将查询参数封装到 HTTP GET/POST 请求中,发送给查询服务器,该服务器启动了嵌入式 HTTP 服务器模块。查询参数可能包含用户希望得到的数据的名称或剖面轨迹、物理量名称等信息。HTTP 服务器解析这个请求后,立刻启动数据库查询线程,将用户希望操作的数据检索出来,如果是实时计算类型的请求,比如模式剖面、模式 $T\ln p$ 图、模式时序图等,则 GDS(Global Data Sharing,全局数据共享)查询服务器会启动多线程计算,得到最终的查询结果。如果是简单的数据检索,则查询结果会用 Protocol Buffer 来封装,对于实时计算请求,会封装为 JSON 字符串,通过 HTTP 应答返回客户端,客户端对返回结果进行 Protocol Buffer 或 JSON 反序列化得到查询结果后进行后续处理或可视化显示。

对于来自 MICAPS4 客户端或者其他算法的小批量数据写入请求,待写入数据的二进制文件首先被封装在 HTTP POST 请求中,向 GDS 查询服务器发送。查询服务器收到写入请求后,首先对用户及写入位置进行权限验证,授权通过后,GDS 查询服务器启动数据库写入线程,用户的写入请求被转化成相应数据表和索引表的数据插入语句,数据在分布式存储持久化结束后,将写入的成功/错误码返回用户。

6 监控系统

在系统的实际部署时,Cassandra 分布式数据库,分布式前处理系统,站点实况轮询系统、查询服务器通常分别部署在不同物理服务器上,有时前处理系统、轮询系统和查询服务器也可以共享相同的物理服务器。所有服务器上都启动了监控探针服务,整点后每隔 15 min 向 CIMISS 的监控服务器主动上报该服务器以及相关进程的健康状况。监控系统同 CIMISS 的集成设计如图 6 所示。

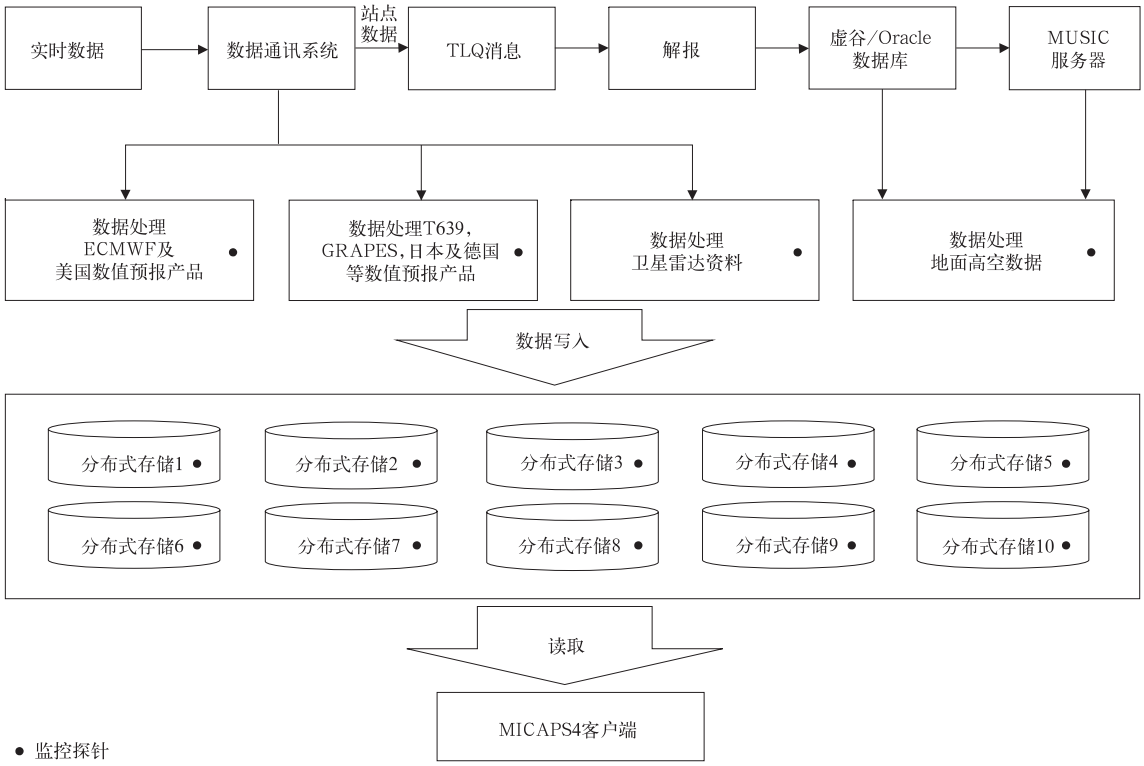

图 6 监控探针与系统集成

Fig. 6 Monitor agent and system integration

7 结 论

1) 针对海量实时气象数据的存储需要,MICAPS4 服务端系统采用对等架构的分布式键值库的方案,将高速到达的数据通过分布式系统的自动负载均衡分配给多台机器进行分散数据存储,而每台机器接受的数据暂时驻留在内存中并定期进行持久

化，既可以减少磁盘操作次数，又可以保证系统在高读写压力情况下减轻系统的随机写入压力。为了保证系统的高可靠性，分布式存储采用多副本机制，提高了数据读取的可靠性和吞吐量，确保全部数据毫秒级写入与查询。

2）针对海量实时气象数据处理需要，MICAPS4 服务端系统采用分布式处理与流式处理相结合的大数据方案，实现数据产生即可见的高速加工流水线，大幅提升数据导入速度和时效性。根据实际业务系统中的运行情况，即便是数据量最大、到齐时序差异最大的集合预报数据也可以在数据到达 2 min 内完成全部处理任务并持久化到存储系统中，比传统气象数据导入系统的性能提升两个数量级以上。

3）MICAPS4 服务器端的容灾备份能力、高度可靠性保障、完备的系统监控、自动化及智能化的数据处理流程、极少的人工参与、便捷的管理控制台为数据管理员和运维工程师建立了可靠的系统平台。

4）大数据系统设计的重要原则之一是从客户端查询的功能和性能需求出发，对服务器端的存储和数据处理进行特定的优化设计，因此为保证所有查询需求的性能最优，冗余的方案是不可避免的，通用型的服务器端解决方案是不存在的。MICAPS4 服务端系统从 MICAPS4 客户端查询需求出发，做了大量的优化工作。尽管该系统也不是一个气象实时数据通用型服务器端解决方案，但 MICAPS4 服务端系统仍然可以很好地满足 SWAN2.0，智能网格预报平台（MICAPS-GFE）、气象内网和大量科研/业务算法的实时数据的读写需求，并基本涵盖了传统文件服务器的全部功能，是一个开放性、集约化的系统。

参 考 文 献

[1] 李月安，曹莉，高嵩，等. MICAPS 预报业务平台现状与发展. 气象，2010，36(7)：50-55.

[2] 高嵩，毕宝贵，李月安，等. MICAPS4 预报业务系统建设进展与未来发展. 应用气象学报，2017，28(5)：513-530.

[3] Batory D S. Concepts for a Database System Compiler // Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, New York, USA：ACM，1988：184-192.

[4] 龚健雅. 空间数据库管理系统的概念与发展趋势. 测绘科学，2001，26(3)：4-9.

[5] 齐贵滨，周尔滨，鞠洋. 利用 samba 服务实现信息共享. 黑龙江气象，2012，28(4)：40-41.

[6] 赵春燕，孙英锐，董峰，等. 高性能气象数据存储集群及在线扩展技术应用. 计算技术与自动化，2013，32(3)：117-121.

[7] 肖华东，孙婧，张玺，等. MARS 软件在数值预报模式产品数据管理中的应用. 应用气象学报，2015，26(2)：247-256.

[8] 沈文海，赵芳，高华云，等. 国家级气象资料存储检索系统的建立. 应用气象学报，2004，15(6)：727-736.

[9] 钱建梅，孙安来，徐喆，等. 风云气象卫星数据存档与服务系统. 应用气象学报，2012，23(3)：369-376.

[10] 李集明，沈文海，王国复. 气象信息共享平台及其关键技术研究. 应用气象学报，2006，17(5)：621-628.

[11] Dong B，Qiu J，Zheng Q，et al. A Novel Approach to Improving the Efficiency of Storing and Accessing Small Files on Hadoop：A Case Study by PowerPoint Files // 2010 IEEE International Conference on Services Computing (SCC). 2010：65-72.

[12] 刘高军，王帝澳. 基于 Redis 的海量小文件分布式存储方法研究. 计算机工程与科学，2013，35(10)：58-64.

[13] 王若曈，黄向东. 海量气象数据实时解析与存储系统的设计与实现. 计算机工程与科学，2015，37(11)：58-64.

[14] 肖卫青，杨润芝. Hadoop 在气象数据密集型处理领域中的应用. 气象科技，2015，43(5)：823-828.

[15] 陈东辉，曾乐. 基于 HBase 的气象地面分钟数据分布式存储系统. 计算机应用，2014，34(9)：2617-2621.

[16] 李永生，曾沁，徐美红，等. 基于 Hadoop 的数值预报产品服务平台设计与实现. 应用气象学报，2015，26(1)：122-128.

[17] Videla A，Williams J J W. RabbitMQ in action：Distributed messaging for everyone. *Manning*，2012.

[18] Hintjens P. ZeroMQ：Messaging for Many Applications. O'Reilly Media，Inc，2013.

[19] Kreps J，Narkhede N，Rao J. Kafka：A Distributed Messaging System for Log Processing // Proceedings of the NetDB. 2011：1-7.

[20] Toshniwal A，Taneja S，Shukla A，et al. Storm@twitter // Proceedings of the 2014 ACM SIGMOD International Conference on Management of data. ACM，2014：147-156.

[21] Zaharia M，Chowdhury M，Das T，et al. Fast and interactive analytics over Hadoop data with Spark. *USENIX Login*，2012，37(4)：45-51.

[22] Carbone P，Katsifodimos A，Ewen S，et al. Apache Flink：Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*，2015，38(4)：28-38.

[23] Ranjan R. Streaming big data processing in datacenter clouds. *IEEE Cloud Computing*，2014，1(1)：78-83.

[24] Zaharia M，Chowdhury M，Das T，et al. Resilient distributed datasets：A Fault-tolerant Abstraction for In-memory Cluster Computing // Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation. USENIX Association，2012：2.

[25] 杨润芝，马强，李德泉，等. 内存转发模型在 CIMISS 数据收发系统中的应用. 应用气象学报，2012，23(3)：377-384.

[26] 邓莉，王国复，孙超，等. 基本气象资料共享系统建设. 应用气象学报，2004，15(增刊 I)：33-38.

[27] 王国复，李集明，邓莉，等. 中国气象科学数据共享服务网总体设计与建设. 应用气象学报，2004，15(增刊 I)：10-16.

The Architecture Design of MICAPS4 Server System

Wang Ruotong[1)] Wang Jianmin[2)] Huang Xiangdong[2)] Dong Yifeng[2)] Long Mingsheng[2)]

[1)] (*National Meteorological Center*, *Beijing* 100081)

[2)] (*School of Software*, *Tsinghua University*, *Beijing* 100084)

Abstract

Meteorological data are typical non-structure data, which reach dozens of TBs per day. Data pre-processing, data storage and data access based on RDBMS and file system become the bottleneck of MICAPS3. To fulfill MICAPS4 users' need of fast, in-time query of meteorological real-time data, according to the multi-dimension model and the user query behavior of meteorological data, using non-relational key-value DDBMS, a high performance massive meteorological data storage system and a stable 7×24 distributed data pre-processing system is designed and established. MICAPS4 uses a client/server system architecture, and high-performance server cluster system is the critical component of MICAPS4. Using distributed key-value data model and P2P infrastructure, MICAPS4 server system distributes all real-time data which arrive at a very high speed to multiple servers through an automatic load balance algorithm, and all data are stored in memory initially and persistent to hard disk periodically, which can not only reduce the disk I/O operating times, but also guarantee the reduction of writing pressure accompanying the high load of reading pressure. To enhance the data and system reliability, distributed system architecture and multiple data replica are used, which also improves the throughput capacity of the system. According to statistic results gained from product environment, the performance of MICAPS4 server system improves 100 times more than MICAPS3. MICAPS4 server system transits all meteorological real-time data storage from file system to database, from centralized system to distributed system. The system becomes the core production system of China Meteorological Administration in 2015 and is popularized nationwide. Under the condition of massive meteorological data and concurrent access of many users, it shows high stability and excellent read-write performance, and it is also highly scalable and maintenance friendly. MICAPS4 high performance server system includes 5 sub-systems including distributed storage system, distributed pre-processing system, station data polling system, data query server and monitoring probe. The distributed storage system provides high performance data accessing services of meteorological real-time data in both random and sequence mode, the distributed pre-processing system implements the stream computing function of massive meteorological real-time data by adopting the peer to peer distributed system infrastructure, the station data polling system implements the heterogeneous station observation replica data synchronization function over different systems, the data query server implements MICAPS4 client real-time computing function by means of the multi-threading server technology, and the monitoring probe is deployed in each server node and reports host health messages periodically. The overall design of MICAPS4 server system is depicted, and the motivation, core technologies and the design of each sub-system are also introduced.

Key words: MICAPS4; big data; distributed storage; distributed computation; real-time data