一种利用并发提高数据处理吞吐率的模型①

周钦强　(广东省大气探测技术中心 广东 广州 510080)

摘　要: 针对分布式组网终端集中式数据采集与处理系统实时性高、数据吞吐量大、周期性强等特点，引入了一种利用并发机制的大吞吐量实时数据处理模型，在讨论模型基本结构和原理的基础上，深入研究了决定模型数据吞吐效率和时效性能的并发缓存机制和算法设计。利用该模型实现的数据采集处理系统在基于 GPRS 组网的 1500 个自动气象站数据采集与处理业务应用中，性能稳定可靠，能够满足对周期性大批量实时“浪涌”数据进行实时快速处理的要求，大大提高了数据处理的吞吐效率。

关键词: 并发; 数据处理; 实时; 吞吐量; 模型

Real-Time and Large Throughput Data Processing System Using Concurrency

ZHOU Qin-Qiang

(Air Observing Center of Guangdong Meteorology, Guangzhou GuangDong 510080, China)

Abstract: Data acquiring and processing from distributed terminal unit by network are usually real-time, large throughput and periodic, so data acquisition, processing and result should be done within data output period in high-efficiency. A real-time and large throughput data processing model using concurrency is built for this given application. The working principle and basic structure for the model are discussed in detail, also concurrency for the model and corresponding key buffer which concurrent threads put data into and gets data out are analyzed in particular. As is used in observation application networked by 1500 automatic weather stations, which has indicated that the model is capable in processing periodic surge observation data and has higher data throughput, as well as implements high speed and accurate results, high stability and reliability.

Keywords: concurrency; data processing (DP); real-time; data throughput; model analysis

分布式数据采集终端具有分布式、实时、同步、高密度、周期性输出等特点，数据采集往往通过无线或有线方式集中采集到数据处理中心。基于 GPRS 通信技术的新一代自动气象站组网[1,2]模式实现了整网探测数据高密度实时高效传输，在一个有限的时间段内数据采集处理系统要完成大批量周期“浪涌”数据的采集、处理及输出工作，而输入和输出一般属于 IO 操作相对而言会占用比较多的时间资源[3]，能否及时准确处理实时探测数据成为气象业务应用的关键环节，也成为了制约气象探测网布点规模发展的瓶颈所在。

引入并发机制的实时数据处理模型，着力分析利用并发机制提高数据采集处理吞吐效率的原理，重点讨论了决定并发效能的关键点--并发缓存与算法设计，基于所讨论模型建立的大吞吐量实时数据处理系统，对全省 1500 个数据采集终端的周期性“浪涌”数据能够及时准确处理，输出统一的业务应用数据产品，以满足整网数据处理实时、快速、准确的性能要求。实际应用表明，该模型的对组网终端数据的采集、处理及时准确、运行稳定可靠，为短时临近气象预报服务提供了及时可靠的观测数据保障，并为集中式数据采集处理的自动气象站探测网规模发展瓶颈提出了

① 基金项目:广东省科技计划项目:基于无线通信技术的地面气象探测站网的研究与开发(项目编号:2005B60401024)
收稿时间:2010-03-04;收到修改稿时间:2010-03-31

一种切实有效的解决方案。

1　模型结构与原理

1.1 模型结构

并发数据处理模型的输入为分布在全省各地的数据采集终端按照一定采集周期同步发送的探测数据，输出为固定的业务应用数据产品。基于 GPRS 通信技术的数据采集网实现了整网数据采集高密度实时高效传输，正常网络状态下数据从采集终端到模型输入端可认为是实时传输，因此整网数据采集、处理的实时性要求基本上取决于模型输入端到输出端的处理效能，模型基本结构如图 1 所示。

1.2 工作原理

首先，按照数据采集终端归属地原则对各地区的终端数据采集进行并发管理，也就是说不同地区的数据采集接收相互独立互不干扰;

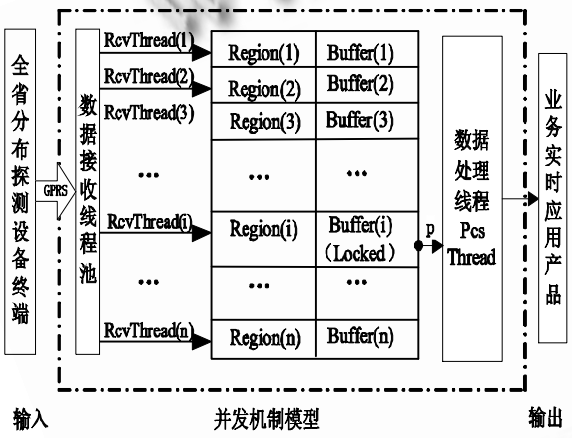

图 1　模型基本结构

其次所有地区数据的处理与接收引入并发管理机制，使每个数据接收线程与数据处理线程对数据缓存堆栈进行并发读写操作时效性能最大化。模型设计为每个地区 Region(i) 分配有独立的数据堆栈 Buffer(i)，应用线程池化技术[4]动态为每个堆栈分配一条独立的数据接收线程 RcvThread(i)，该线程将接收数据存储到对应地区的数据堆栈，避免了线程的频繁创建和销毁，节约系统资源，保证数据终端大批量“浪涌”数据及时消化而不致输入端堵塞。同时，数据处理线程 PcsThread 数据堆栈遍历指针 p 循环遍历每个地区数据堆栈，读取处理完成后，清空该地区数据堆栈。

1.3 模型并发效能分析

由上述分析可见，所有数据接收线程之间不存在竞争并发操作，而任一条数据接收线程 RcvThread(i) 与处理线程可能会同时访问同一个数据堆栈，存在访问竞争，因此引入多线程并发管理机制。

在数据处理及时的前提下，模型启动一条或多条数据处理线程。处理线程数量越少，与数据接收线程并发读写缓存堆栈的可能性越小。以一条处理线程读取缓存区数据为例，仅有一个数据缓存堆栈加锁，而其他每个数据缓存堆栈仍可由一一对应的数据接收线程自由写入；如果每个地区数据缓存堆栈分别启动一条独立数据处理线程，则可能导致多个或全部数据缓存堆栈被锁，从而导致模型输入端大批量“浪涌”数据堆积或模型输出端大部分时间空闲。在现代计算机硬件成熟的背景下，开辟适当大的数据缓存堆栈，及时把全部终端采集数据接收到缓存堆栈区，避免大批量数据堆积在模型输入端，从而增加模型的数据吞吐量。当然，如果数据处理任务繁重导致数据处理无法及时完成，无法保障业务应用产品实时性时，可适当增加数据处理线程的数量，数据处理线程数量多少与模型性能效能指数是一个最优问题，是后续研究的重要内容。

2　并发管理机制

提高模型数据处理的时效性以及数据吞吐量，每个地区采集终端的数据接收线程、数据处理线程对数据缓存区的并发读写操作均以最小等待时间工作，也就是说，数据处理线程与不同地区采集终端数据接收线程协作最优化，以完成所有周期性大吞吐量“浪涌”数据的接收处理任务。并发管理设计避免了数据单一读或单一写，数据处理线程读取某个堆栈数据的同时，其他堆栈对应的数据接收线程仍然可以进行写入操作，以此提高模型数据处理的吞吐效率。

(1) 接收线程池化

线程池的基本原理是先启动若干数量的数据接收线程，并让这些线程都处于睡眠状态，当某个地区采集终端请求数据上传连接时，就会唤醒线程池中的某一个睡眠线程，让它来处理该连接请求，当处理完这个请求任务后，该条线程又处于睡眠状态。如果为每

个连接请求分别创建一条新线程的话，那耗费的 CPU 时间和内存将是惊人的，而线程池将会节约大量的系统资源，使得更多的 CPU 时间和内存用来处理接收数据的解析任务，而不是频繁的创建与销毁线程[2]。

(2) 数据接收线程独立写入管理

由于每个地区采集终端分配有专用数据缓存堆栈用以存储该地区终端的接收数据，每条数据接收线程可同时对其所对应的数据缓存堆栈进行数据写入操作，即 RcvThread(1),RcvThread(2),…,RcvThread(i),…,RcvThread(n)可同时将各自接收到的数据分别写入对应的缓存堆栈 Buffer(1),Buffer(2),…,Buffer(i),…Buffer(n)。因此，当地区 i 的数据接收线程 RcvThread(i)对数据缓存堆栈 Buffer(i)进行写入操作时，其它地区数据接收线程 Buffer(n)(ni)不必排队等候[5]。基于此堆栈模型的并发管理机制大大提高了大批量“浪涌”数据的并发接收能力，避免了数据接收排队阻塞，增强了模型输入数据的消化吸收能力。

(3) 数据处理与接收线程对数据堆栈的并发读写管理[5]共享资源的同步读写操作是典型的并发管理。当数据处理线程 PcsThread 读取地区 Region(i)对应的数据堆栈 Buffer(i)时，Buffer(i)被数据处理线程 PcsThread 加锁，对应的数据接收线程 RcvThread(i)必须等待数据处理线程 PcsThread 完成堆栈 Buffer(i)读取操作并释放锁后才能进行数据写入操作，而其他数据接收线程 RcvThread(n)(ni)仍可对分别对应的数据堆栈 Buffer(n)(ni)进行写入操作；反之，当数据接收线程 RcvThread(i)对数据堆栈 Buffer(i)进行写入操作时，如果数据处理线程 PcsThread 恰好读取该数据堆栈 Buffer(i)，则数据处理线程 PcsThread 必须等待数据接收线程 RcvThread(i)完成数据写入操作并释放 Buffer(i)锁后才能读取缓存堆栈 Buffer(i)的数据。

(4) 并发缓存堆栈性能

模型基于 JavaSE 设计实现，具有良好的可扩展性和跨平台性，并发缓存堆栈基于 JavaSE1.5 的 ConcurrentHashMap[6,7] 设计实现。ConcurrentHashMap 提供比 Hashtable(或 synchronizedMap)更高程度的并发性。它为不同的 hash bucket(桶)使用多个写锁和使用 Java 内存管理(JMM)的不确定性来最小化锁被保持的时间--或者根本避免获取锁，使得并发应用程序有着非常好的吞吐量。ConcurrentHashMap 摒弃了单一的 map 范围的锁，取而代之的是由 32 个锁组成的集合，其中每个锁负责保护 hash bucket 的一个子集，具有 32 个独立的锁意味着最多可以有 32 个线程可以同时修改 map[6]。而 Hashtable 或 synchronizedMap 可伸缩性的主要障碍是它使用了一个 map 范围(map-wide)的锁，为了保证插入、删除或者检索操作的完整性必须保持这样一个锁，而且有时候甚至还要为了保证迭代遍历操作的完整性保持这样一个锁。这样一来，只要锁被保持，就从根本上阻止了其他线程访问 Map，即使处理器有空闲也不能访问，这样大大地限制了并发性。

可见，该模型对于数据处理系统效率的主要帮助在于引入了并发，并且系统大吞吐率的提高也得益于并发缓存堆栈深度的改善。

3 算法设计

模型设计实现是以采集终端数据的接收、处理为根本，因此除了数据接收线程和数据处理线程之间并发优化协作管理机制设计外，数据接收算法和数据处理算法自身的良好设计也直接影响模型性能的关键点[8]。数据接收算法负责高效快速接收所有地区完整的终端采集数据，数据处理算法负责所有缓存堆栈的数据解析处理并形成可靠的业务应用产品。

3.1 数据接收算法

按照数据采集终端归属地原则，每个地区数据采集终端通信设置参数中包含数据上传服务器 IP 地址、端口和通信协议，其中端口与该地区终端数据上传接收缓存堆栈一一对应，线程池为每个缓存堆栈分配一条数据接收线程以执行该堆栈的数据接收存储任务，程序流程图如图 2 所示。

数据接收子线程 RcvThread(i)经线程池唤醒后，首先获取对应数据缓存堆栈 Buffer(i)的访问权限，即“锁”权限，若获取“锁”权限成功，则首先对 Buffer(i)进行加“锁”操作，然后将接收数据全部写入缓存堆栈 Buffer(i)。写入成功后，释放数据缓存堆栈 Buffer(i)的“锁”权限。数据接收子线程 RcvThread(i)继续侦听数据上传接收请求，若连续超过 6 小时无数据上传请求，则断开该连接，数据接收子线程 RcvThread(i)回归线程池，并进入睡眠

状态，以节省系统资源。

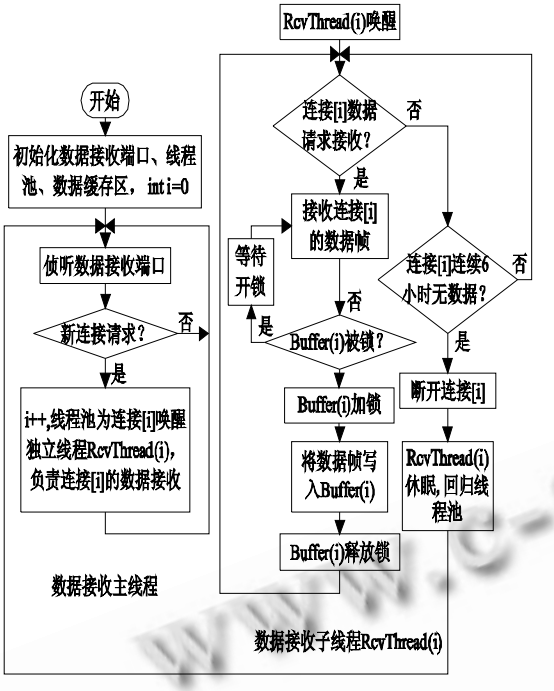

图 2　数据接收算法流程图

值得注意的是，数据采集终端的每一份完整数据报通过 TCP 通信协议上传时，可能会被分割成为两个或两个以上的数据帧上传。因此，数据接收子线程接收到的每个数据帧可能由一份或多份完整数据报和不完整数据报组成，数据接收子线程 RcvThread(i)获得数据缓存堆栈 Buffer(i)“锁”权限后，按照“先进先出”原则进行数据写入操作，并按照每个数据帧接收的先后顺序依次将其写入缓存堆栈 Buffer(i)，并释放“锁”权限。因此在任意时刻，数据接收堆栈 Buffer(i)中的缓存数据由若干份连续完整的数据报和最多两份(堆栈出、入口)不完整探测数据报组成。

3.2 数据处理算法

数据处理线程 PcsThread 负责遍历每个数据接收缓存堆栈 Buffer(1),Buffer(2),…,Buffer(i),…Buffer(n)，根据通信协议和数据格式对所有数据缓存堆栈中的数据报进行解析，并按照业务需求生成固定格式数据应用产品输出，程序流程图如图 3 所示。

数据处理线程 PcsThread 在遍历数据缓存堆栈时一旦获得堆栈 Buffer(i)“锁”权限，按照“先进先出”原则读取堆栈 Buffer(i)中的全部数据，数据解析完成后将该堆栈清空。按照数据报文格式，数据处理线程 PcsThread 首先判断不完整数据报，提取数据缓存堆栈中的所有完整探测数据报作解析处理，并将不完整数据报返存至数据缓存堆栈 Buffer(i)，以便与后续接收数据拼接成为完整数据报；然后释放堆栈 Buffer(i)的“锁”权限。

由数据接收算法决定了数据处理算法对任一个数据接收缓存堆栈 Buffer(i)，数据处理线程 PcsThread 在释放 Buffer(i)“锁”权限之前必须按照如下算法，否则数据处理线程 PcsThread 将会因为无法识别不完整数据报而丢失大量的采集数据。

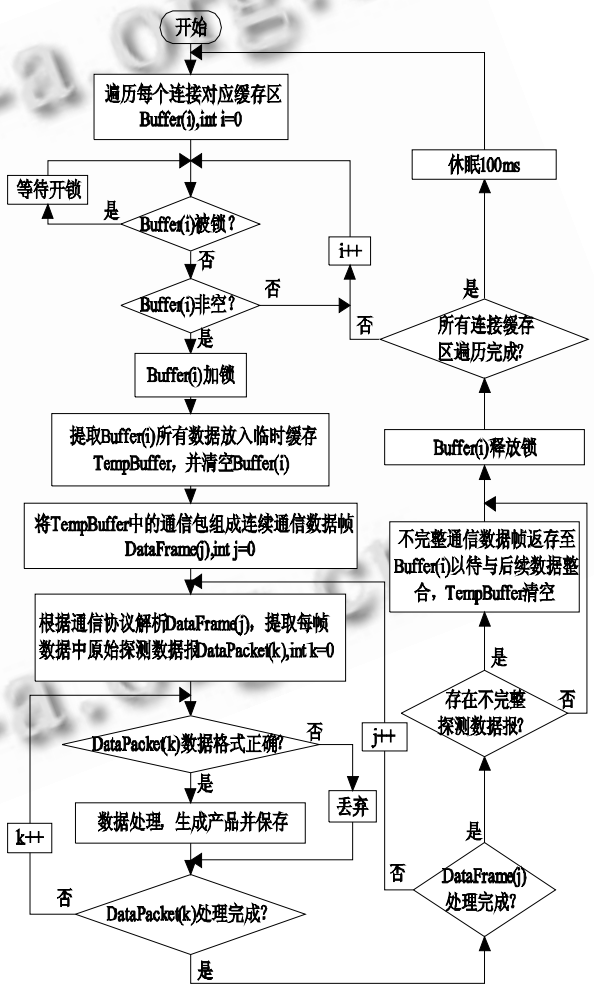

图 3　数据处理算法流程图

(1) 读取缓存堆栈数据后清空缓存堆栈 Buffer(i)；

(2) 判断提取所读取数据的完整数据报作解析处理，并将提取后的不完整数据报返存至数据缓存堆栈 Buffer(i)，以便与后续接收的数据帧组成完成数据报；

(3)数据处理线程 PcsThread 释放缓存堆栈 Buffer(i)“锁”权限。

4 应用结果及分析

基于所讨论模型实现的数据采集处理系统成功部署于广东省近 1500 个 WP3103 型自动气象站数据观测网和深圳 100 个 Vaisala Maws301 型自动气象站观测网的数据采集处理[5]，系统运行稳定可靠，对于周期为 6 分钟一份探测数据报采集处理高效、准确、及时。在将采集周期改为 1 分钟一份和模拟 4000 个组网自动站的大吞吐量数据采集处理压力测试中，系统仍能准确及时采集处理全部站网气象探测数据。随着气象现代化发展，全省自动气象站观测网的容量与日俱增，基于该模型实现的数据采集处理系统性能稳定，输入输出数据准确可靠，满足了自动气象站观测网容量不断扩大所带来的大吞吐量数据处理的时效性要求，同时也为其他行业类似应用背景提供了一种有效参考解决方案。

文中所阐述模型结构设计仅采用了一条数据处理线程循环遍历所有数据接收缓存堆栈，实现接收数据的处理和应用产品输出。当模型无法满足更大输入数据量的处理要求时，可适当增加数据处理线程的数量，以优化模型满足数据吞吐量的性能要求[9]。但多条数据处理线程之间以及其与所有数据接收线程之间的并发竞争管理需要充分考虑优化，也是今后继续深入研究的重要内容之一。

参考文献

1 敖振浪,伍光胜,周钦强等.基于 GPRS 通信组网的自动气象站数据采集系统.广东气象, 2007,4:152－153,200.

2 刘聪,顾建,吴国平等.基于 GPRS 的远程气象观测数据实时采集传输系统及其应用.应用气象学报, 2004,15(6):712－718.

3 Vincent_lon,一种利用并发提高系统数据吞吐率的模型及其分析.2009.10.03,http://blog. csdn.net/vincent_lon/archive/2008/09/09/2904521.aspx.

4 Scott Oaks, Henry Wong. Java Threads, Third Edition. O'Reilly, 2004.

5 周钦强,谭鉴荣,伍光胜等.基于 TCP 多连接通信实时并发数据处理技术研究.计算机工程与应用, 2007,18:246－248.

6 BrianGoetz. ConcurrentHashMap & CopyOnWrite ArrayList offer thread safety and improved scalability. 2009.06.18,http://www-128.ibm.com/developerworks/.

7 Doug Lea. JAVA 并发编程—设计原则与模式(第二版).北京:中国电力出版社, 2004.

8 罗续业.海滨自动观测系统数据处理技术研究[硕士学位论文].国家海洋局海洋技术研究所. 1997.

9 唐勇,胡振宁,李泽滔.利用 C++Builder 进行多线程实时数据处理的研究. 贵州工业大学学报(自然科学版), 2000,107(6):89－90.