计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(12):262−267 [doi: 10.15888/j.cnki.csa.006710]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

高性能时间序列数据库网络中间件研究①

徐　丹, 艾文凯

(南京南瑞继保电气有限公司 研究院数据平台部, 南京 211101)
通讯作者: 徐　丹, E-mail: xud@nrec.com

摘　要: 本文研究了时间序列数据库网络中间件, 总结分析了常见的网络中间件特点. 根据时间序列数据库的高性能要求, 对网络中间件的关键技术进行了研究, 采用多个 Reactor 和 Thread Pool 的并发模型来提高线程并发度, 并提出了多帧消息格式, 实现消息协议的动态扩展. 最后在 PCS-9000 时间序列数据库中实现了相关的网络中间件, 达到了预期的性能指标要求.

关键词: 时间序列数据库; 网络中间件; 多线程; 多帧消息; 事件多路分离

引用格式: 徐丹,艾文凯.高性能时间序列数据库网络中间件研究.计算机系统应用,2018,27(12):262–267. http://www.c-s-a.org.cn/1003-3254/6710.html

Research on Network Middleware of High Performance Time Series Database

XU Dan, AI Wen-Kai

(Data Platform of Research Institute, Nanjing Nari-Relays Electric Co. Ltd., Nanjing 211101, China)

Abstract: In this paper, we present the study of network middleware of time series database, summarize and analyze the characteristics of common network middleware. According to the high performance requirement of time series database, the key technology of network middleware is studied. The concurrent model of multiple reactor and thread pool is adopted to improve the concurrency of thread, and the multi-frame message format is proposed to realize the dynamic expansion of the message protocol. Finally, the related network middleware is realized in the PCS-9000 time series database. The middleware achieves the expected performance requirements.

Key words: time series database; network middleware; multi thread; multi frame message; event dispatch

随着电力系统高实时要求应用的发展, 时间序列数据库在电力系统中的应用越来越广泛, 广域测量系统、变电站监控、调度、直流、稳控等多个电力自动化系统中都能见到身影, 用于满足存储海量、高频数据的需求. 时间序列数据库主要用于处理秒级、毫秒级高频数据, 每个存储周期 (毫秒或是秒) 会产生一批数据, 因此数据量极大, 对网络流量和网络中间件的读写性能要求极高, 通用架构的网络中间件很难满足时间序列数据库的特别要求, 需要定制开发适用于时间序列数据库的网络中间件, 针对报文、网络和存储特点进行全面优化.

1　传统网络中间件分析

1.1　基于远程调用的网络中间件 (Remote Procedure Call, RPC)

RPC 的基本通信模型是基于 Client/Server 进程间相互通信模型的一种同步通信形式; 它对 Client 提供了远程服务的过程抽象, 其底层消息传递操作对 Client 是透明的. 在 RPC 中, Client 即是请求服务的调用者 (Caller), 而 Server 则是执行 Client 的请求而被调

① 收稿时间: 2018-05-21; 修改时间: 2018-06-15; 采用时间: 2018-07-19; csa 在线出版时间: 2018-12-03

用的程序 (Callee)[1,2].

1.2 基于分布式对象的网络中间件 (Object Request Broker, ORB)

这种形式的网络中间件以 DCOM 与 COBRA 为代表, 但无论哪种标准与实现, 均由以下部份组成.

(1) 实际完成服务和功能的远程对象, 负责实际完成系统服务, 接受远程请求.

(2) 访问客户端代理, 负责从客户端接受请求, 并将请求转换为远程调用发送到服务端.

(3) 对象请求代理 ORB, 提供一个通信框架, 透明的在异构分布式计算环境中传递对象请求, 负责定位对象实现, 并将请求传输给对象实现后返回结果, 在分布式网络中间件中处于核心地位[3,4].

1.3 基于消息队列的网络中间件 (Message-oriented Middleware)

基于消息的网络中间件主要用于在不同应用间投递消息, 专注于异构环境的消息交换, 目前最流行的面向消息的网络中间件是 Apache 的 ActiveMQ.

消息可不包括语意和状态, 因此面向消息的网络中间件对应用而言是最透明的选择. 应用无需关心消息的来源、连接方式, 而中间件无需关心消息的语意, 这样可以很好的实现与应用解耦[5–11].

1.4 面向连接的网络中间件

此类中间件一般并不直接提供高级服务, 而仅仅是负责对网络业务进行封装, 降低编程复杂性, 以 ACE、boost.asio、libevent 等为代表.

这些库基本都实现了 Reactor 与 Proactor 设计模式, 实现了高效的多路事件分离. 下面以 ACE 库为代表介绍其基本概念.

ACE 自适配通信环境 (ADAPTIVE Communication Environment) 是可自由使用、开放源码的面向对象构架 (framework), 它实现了许多用于并发通信软件的核心模式. ACE 提供了一组丰富的可重用 C++包装外观 (wrapper facade) 和构架组件, 可跨多种平台完成通用的通信软件任务, 其中包括: 事件多路分离和事件处理器分派、信号处理、服务初始化、进程间通信、共享内存管理、消息路由、分布式服务动态 (重) 配置、并发执行和同步等[12–16].

以上网络中间件虽然都是较为成熟的框架, 但并不适用于大数据量、高性能要求的时序数据处理. 首先、这些框架都是通用框架, 针对时序数据的键值对特性无法优化, 使得通讯协议过于庞大而造成资源浪费; 其次、通用框架需要考虑过多的场景, 因此中间件过于复杂, 例如 CORBA 就是一个重量级框架, 将会拖累整个时序数据库部署; 再次、传统中间件很好的屏蔽了底层操作系统相关性, 做到通用性和跨平台, 但这是在损失一定的系统性能基础上, 因此无法满足时序数据的大数据量和低响应时延的要求.

2 高性能中间件研究

2.1 高性能中间件架构研究

本文在设计高性能中间件时, 提出了以下核心原则:

(1) 尽量减少数据共享, 以减少共享的互斥开销. 为此在中间件内部, 所有的数据都设计为线程私有, 两个线程之间不存在共享数据, 数据通过消息进行传递, 这样减少了大量的互斥量、信号量、竞争条件等同步手段.

(2) 每个线程异步且独立运行, 一个线程对其他线程没有时间依赖性, 不存在等待其他线程的时序一致性要求. 线程间的协作请求、中间件内部命令也通过消息发送. 每个线程维护自身的消息队列, 消息队列除本线程外, 仅有分发器可以访问, 不与其它线程共享.

(3) 在本地尽量采用进程间管道进行通讯, 在节点间通讯, 可选点对点模式, 减少中间件消息代理的负荷, 提高效率并避免形成瓶颈.

根据上述原则, 首先需要确定高性能中间件的并发模型, 并发度的好坏决定了网络中间件的性能优劣. 目前主流的并发模型有以下几种:

表 1 主流并发模型

编号	并发方式	阻塞	开销	线程数
1	Accept/thread	是	中	不定
2	Poll(reactor)	是	低	固定
3	Reactor+thread-per-task	否	高	不定
4	Reactor+worker thread	否	低	固定
5	Reactor+thread pool	否	低	固定
6	Reactors in thread	否	中	不定
7	Reactor(多个)+thread pool	否	中	不定

时序数据客户端进程的多个连接存在相关联的可能性很小, 且多个 IO 线程可明显提高系统资源利用率, 因此必须要有良好的线程扩展性; 其次通知的异步调用有助于提高效率, 线程池的设置则明显减少了创建开销. 结合上述特点和七个方案的特性, 我们最终选择

了方案 7.

本文将整个系统内的线程分为两种类型:应用线程 (Application thread), 在中间件以外由应用程序创建的业务线程, 会访问中间件的接口函数完成功能; 核心线程 (Core thread/IO Thread), 在中间件内部创建的线程, 用于实际的功能实现.

线程通过线程池进行管理, 根据配置预先创建读线程池和写线程池, 默认创建 8 个读线程和 8 个写线程, 通过统一的线程管理器进行管理. 读写线程池依据选择的负载均衡算法, 选定具体的处理线程, 并将消息放入对应的线程消息队列, 网络中间件默认选择的策略为均衡轮循策略. 线程对象关系图如图 1.

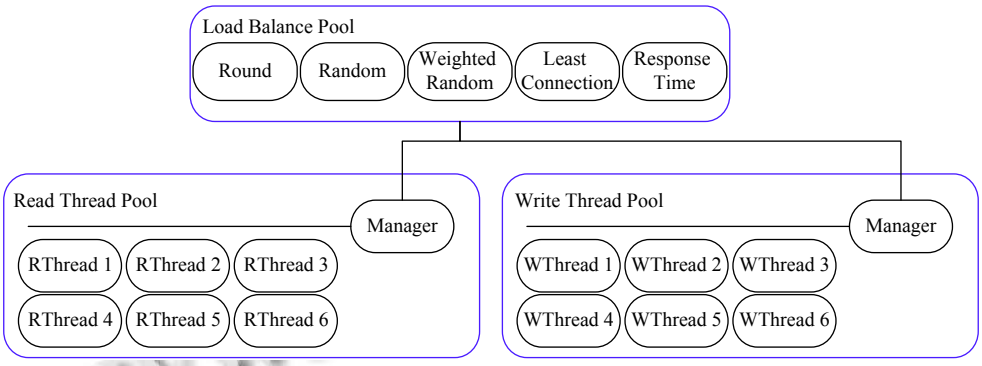

图 1 线程对象关系

负载均衡策略池中包含多种负载均衡策略, 有均衡轮循策略、随机均衡策略、权重随机均衡策略、响应速度均衡策略、最少连接数均衡策略等. 读写线程池依据选择的负载均衡算法, 选定具体的处理线程. 每一个读写线程池根据配置预先创建一定数量线程的读线程池和写线程池, 通过统一的线程管理器进行管理, 使用多线程进行 IO 读写可明显提高系统资源利用率, 而采用预先创建线程可以明显减少后期频繁创建销毁线程的开销. 每个线程都有自己的消息队列, 从消息队列中获取处理报文.

2.2 消息与协议机制研究

2.2.1 消息格式设计

对于消息的要求一般来说比较复杂, 因电力行业软件在具有大量小尺寸应用消息的同时, 也会有很多比较巨大的数据库同步消息, 同时由于消息的实际传输方式存在多种可能性, 如两个网络节点间、同一节点的两个应用之间、同一进程的两个线程之间. 对于小尺寸的消息而言, 在栈上分配空间的开销要远小于在堆中分配-析构的开销, 因此对于小尺寸的消息应尽量在栈上分配, 同时由于大尺寸的消息容易造成栈溢出, 不可在栈上分配, 因此对于不同大小的消息应提供不同的内存分配方案. 且同一帧消息, 有可能被发送至不同的接收端, 如果接收方处于同一进程内, 同一进程共享地址空间, 消息拷贝没有意义; 如果消息体很大, 拷贝代价很高, 因此对于消息应提供引用计数的功能.

结合以上需求, 对消息进行以下设计:

小尺寸消息在栈上分配, 因大多数消息长度有限, 如果使用内存操作符进行堆分配、析构, 会造成性能瓶颈, 也容易造成内存空间碎片, 虽然部分中间件使用预先分配、统一使用的方式, 但在栈上分配性能更佳.

大尺寸消息在堆上分配, 防止栈溢出, 同时支持引用计数, 以避免内存拷贝开销, 此外由于不同的应用, 不同的系统可能有不同的内存管理机制, 消息可支持外部特定的析构函数.

小尺寸消息在栈上分配时, 消息内容直接存储在 msg_buffer 中, msg_buffer 的长度由常量在编译时指定. 设计第一个 Byte 为消息类型 (msg_type), 第二个 Byte 为消息大小 (msg_size,最大支持 256), 最后为 256 Byte 的消息内容 (msg_buffer), 由 msg_size 指定. 消息结构如图 2 所示.

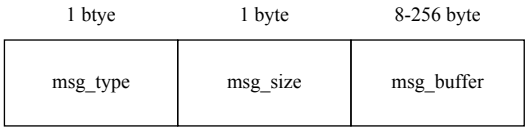

图 2 消息结构

大尺寸消息, 需要防止栈溢出, 同时支持引用计数以避免大内存拷贝开销, 采用堆上分配方式, 设计第一

个 Byte 为消息类型 (msg_type), 第二个 Byte 为空, 其后 8–256 Byte 用于记录实际数据的地址 (dest), 指向的地址空间包含数据的大小 (size)、函数指针 (Extern Func)、引用计数 (Ref counter) 和数据内容信息 (User data). 消息结构如图 3 所示.

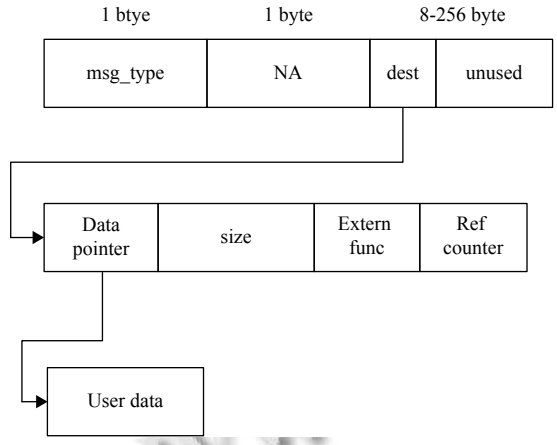

图 3 消息体结构

2.2.2 可扩展协议设计

传统的网络消息格式会被设计为消息头 (Message Head)+消息体 (Message) 的模式. 如图 4 所示.

图 4 传统网络消息格式

为了能对消息协议进行方便的扩充, 本文引入了多帧消息的概念, 通俗的说, 每个消息可以有多个消息体组成. 每一帧报文长度固定, 由常量在编译时指定, 消息头也被视为一个独立的消息帧, 这样可以通过扩展消息帧数的方式扩展通讯协议. 如图 5 所示.

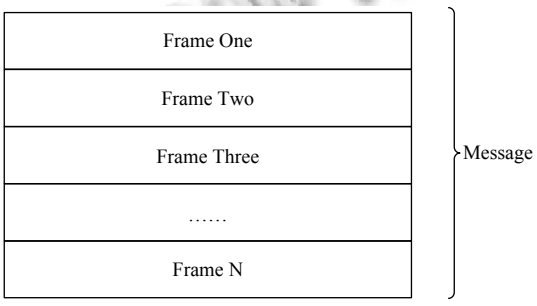

图 5 多帧消息结构

一个消息不再硬性的被区分为消息头+消息体的机制, 一个消息可以由多个消息帧组成. 每个消息帧均为对等的独立消息缓存, 每一帧的大小固定. 应用与中间件可以采用扩展消息帧的方式扩展通讯协议, 而无须破坏现有结构与应用消息结构.

3 高性能中间件实现

基于上述可扩展消息协议和负载均衡链接池并发模型, 设计开发了时间序列数据库高性能中间件组件, 中间件主要包含两个核心模块:

OS_Wrapper 为操作系统基础接口封装层, 用于屏蔽不同操作系统底层调用的差异性, 从而实现中间件的跨平台.

WP_Network 为高性能中间件的核心模块, 包含了协议解析、链接池管理分配、负载均衡实现等功能, 该模块位于 OS_Wrapper 之上, 调用了 OS_Wrapper 封装的底层接口.

3.1 OS_Wrapper 的实现

电力系统时间序列数据库会运行在 Windows、Linux、Solaris、AIX 等多种操作系统上, 由于各个系统的数据类型、系统调用、线程管理、同步原语等均存在差异, 为了保证跨平台可移植性, 采用 OS_Wrapper 来进行包装, 屏蔽差异性, 主要有以下类组成:

表 2 OS_Wrapper 主要构成类

类名	用途
c_system	系统调用与基本功能封装, 包括获取时间、获取线程 id、进程 id、获取 ip 地址等
thread_t	线程类, 封装了不同操作系统的线程操作
memory_allocator	实现了内存分配器, 用于提高内存分配效率

3.2 WP_Network 的实现

WP_Network 是高性能中间件的核心功能模块, 其具有高性能、可扩展协议和负载均衡等特点, 能有效节省网络流量和提高响应速率. 主要由以下多个类组成:

WP_Network 中包含的类较多, 逻辑也比较复杂, 其中各类的静态继承和依赖关系如图 6 所示.

4 高性能中间件的应用与总结

本文研究了时间序列数据库所需的高性能网络中间件特点, 并对其进行了开发实现, 本文所涉及的网络中间件已经运用于 PCS-9000 时间序列数据中. 采用上述方案后, 高性能中间件具有以下特点:

(1) 不需要为每个客户端创建处理线程, 降低了系统资源开销, 支持客户端最大并发数显著提高;

表 3 WP_Network 主要构成类

类名	用途
mailbox_t	实现邮箱功能
wp_object	抽象父类, 声明了基本的 io 接口
wp_socket_base	通讯基类, 封装了基本通讯功能, 屏蔽了不同通讯的差异, 如管道、网络等
tcp_listener_t	Tcp 服务端的侦听功能封装, 实现异步的 tcp 侦听功能
tcp_connecter_t	Tcp 客户端的连接功能封装, 实现异步的 tcp 连接
wp_pub	订阅\发布模式的发布类, 用于实现发布功能
wp_sub	订阅\发布模式的发布类, 用于实现发布功能中的订阅功能
wp_request	请求应答模式中的请求端
wp_reply	请求应答模式中的应答端
wp_msg	用于系统传输的消息单帧, 小尺寸消息采用在栈上分配的方式, 大尺寸消息
wp_poller	不同操作系统、不同通讯方式的 poller 封装.
io_thread_t	系统后台的 io 线程封装, 基于 thread_t 实现, 实现异步的数据传输.

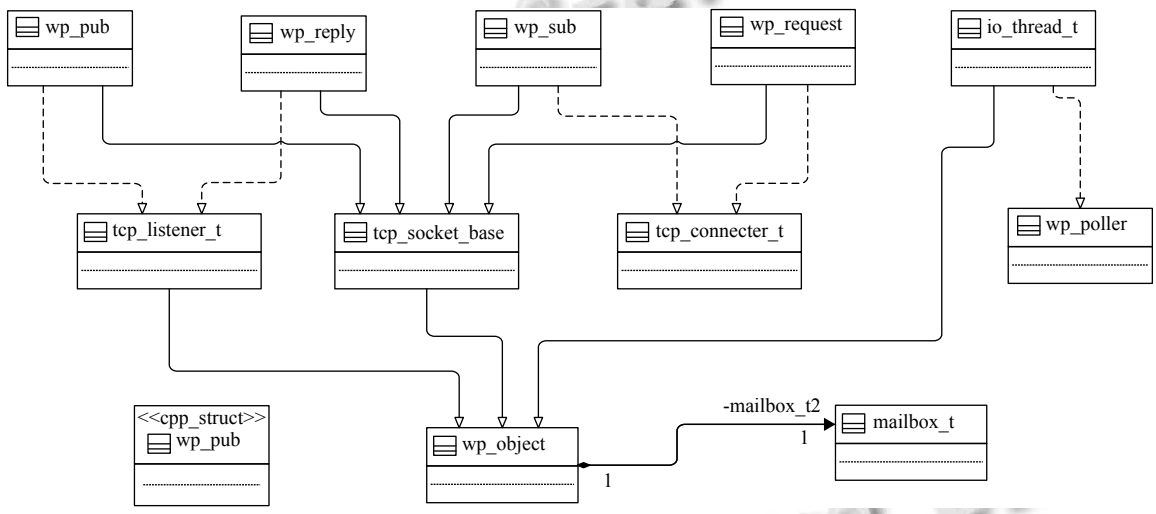

图 6 WP_Network 主要结构

(2) 显著提高报文处理能力, 每秒可处理 500 万事件 (记录);

(3) 降低了网络流量, 服务端占用的 CPU 和内存明显减少.

表 4 与国内外主流时间序列数据库比较

比较项	PCS-9000	国外同类产品	国内同类产品
数据写入速率	单机>500 (万事件/秒)	200<单机<300 (万事件/秒)	400<单机<500 (万事件/秒)
数据查询速率	单机>600 万事件/秒	300<单机<400 (万事件/秒)	400<单机<500 (万事件/秒)
跨区同步速率	300 万事件/秒	不支持	<100 万事件/秒
压缩比	30 倍	20–30 倍	20–30 倍

时间序列数据库在国内外都有广泛应用, 主流产品如海迅时间序列数据库、PI 时间序列数据库、eDNA 时间序列数据库等. 相比较, PCS-9000 时间序列数据库具有以下的优势:

综上所述, 高性能中间件技术的引入大大提高了时间序列数据库的处理能力, 降低了系统和网络资源的开销, 使得同样硬件资源可以实现更高数据量的存储, 具有推广意义.

参考文献

1 吴泉源. 网络计算中间件. 软件学报, 2013, 24(1): 67–76. [doi: 10.3724/SP.J.1001.2013.04296]

2 Dionysiou I, Frincke D, Bakken D E, *et al*. Actor-oriented trust. Technical Report EECS-GS-006, Pullman, WA, USA: Washington State University, 2005.

3 Sivaharan T, Blair G, Coulson G. GREEN: A configurable and re-configurable publish-subscribe middleware for

pervasive computing. Proceedings of OTM Confederated International Conferences on the Move to Meaningful Internet Systems. Agia Napa, Cyprus. 2005. 732–749.

4 Goel S, Sharda H, Taniar D. Message-oriented-middleware in a distributed environment. Proceedings of the 3rd International Workshop on Innovative Internet Community Systems. Leipzig, Germany. 2003. 93–103.

5 Ezenwoye O, Sadjadi SM. RobustBPEL2: Transparent autonomization in business processes through dynamic proxies. Proceedings of the 8th International Symposium on Autonomous Decentralized Systems. Sedona, AZ, USA. 2007. 17–24.

6 Schmidt DC, Cleeland C. Applying patterns to develop extensible ORB middleware. IEEE Communications Magazine, 1999, 37(4): 54–63. [doi: 10.1109/35.755450]

7 罗皓. 一种基于 UNIX 的互联网站搭建方案关键技术研究[硕士学位论文]. 北京: 中国水利水电科学研究院, 2007.

8 Baldoni R, Beraldi R, Piergiovanni ST, *et al*. Measuring notification loss in publish/subscribe communication systems. Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing. Papeete, Tahiti, French Polynesia. 2004. 84–93.

9 栾向晨. 网络安全管理系统中的中间件设计与实现[硕士学位论文]. 沈阳: 东北大学, 2006.

10 Bittner S, Hinze A. On the benefits of non-canonical filtering in publish/subscribe systems. Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops. Columbus, OH, USA. 2005. 451–457.

11 Chand R, Felber P. XNET: A reliable content-based publish/subscribe system. Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems. Florianpolis, Brazil. 2004. 264–273.

12 Chand R, Felber P. Semantic peer-to-peer overlays for publish/subscribe networks. Proceedings of the 11th International Euro-Par Conference on Euro-Par 2005 Parallel Processing. Lisbon, Portugal. 2005. 1194–1204.

13 Erradi A, Maheshwari P, Tosic V. Recovery policies for enhancing web services reliability. Proceedings of 2006 IEEE International Conference on Web Services. Chicago, IL, USA. 2006.

14 Ezenwoye O, Sadjadi SM. Composing aggregate web services in BPEL. Proceedings of the 44th Annual Southeast Regional Conference. Melbourne, FL, USA. 2006. 458–463.

15 Cybok D. A grid workflow infrastructure. Concurrency and Computation: Practice and Experience, 2006, 18(10): 1243–1254. [doi: 10.1002/(ISSN)1532-0634]

16 Erradi A, Maheshwari P. wsBus: QoS-aware middleware for reliable web services interactions. Proceedings of 2005 IEEE International Conference on e-Technology, e-Commerce and e-Service. Hong Kong, China. 2005. 634–639.