文章编号：2095-2163(2020)07-0176-04 中图分类号：TP316.81；TP311.1 文献标志码：A

基于 Linux 的高并发网络聊天系统设计

王 林
（合肥工业大学 仪器科学与光电工程学院，合肥 230009）

摘 要：网络聊天已经成为人们沟通的一种重要方式。在面对众多用户的情况下，一些聊天室由于性能低下已经无法满足人们的需要，这正是本文设计高并发高性能网络聊天系统的目的。本文详细分析各种提高服务器性能的方案，比较其优缺点，研究高并发高性能的软件架构，并将其应用到网络聊天中。本系统采用 C/S 架构，由客户端，服务端组成，客户端与服务端之间通过 TCP 来传递消息。

关键词：网络聊天；高并发；高性能；C/S 架构

Design of high concurrency network chat system based on Linux

WANG Lin
（School of Instrument Science and Opto-electronics Engineering, Hefei University of Technology, Hefei 230009, China）

【Abstract】 Internet chat has become an important way of communication between people. However, faced with many users, some chat rooms have been unable to meet people´s needs due to their poor performance, which is the purpose of designing high concurrent and high performance network chat system. This article will analyze in detail various solutions to improve server performance, compare their advantages and disadvantages, study high-concurrency and high-performance software architectures, and apply them to network chat.This system adopts C / S architecture, which consists of client and server, and the client and server pass messages through TCP.

【Key words】 Online chatting; High concurrency; High performance; C / S architecture

0 引 言

通过网络实时的沟通，其时效性超过了电子邮箱，而其费用也低于电话，所以网络聊天服务备受青睐。

为了使用户有良好的交流体验，实时性很重要。而随着使用聊天软件人数的增加，也对服务器的承载量提出了更高的需求。如果依然采取用户每过几秒询问服务器，从而获得服务端最新的数据。不仅客户端不能在第一时间内获取更新的数据，也是对服务端资源的浪费[1]。研究高并发高性能的服务端架构，降低服务器压力，提高用户体验，也显得很有必要。

本文使用套接字技术，在 Linux-ubuntu 环境下设计服务端程序，windows 环境下设计客户端程序。socket 是程序间进行网络通信的一种接口。服务端程序会有一个 IP 和端口号，客户端通过服务器的 IP 和端口来找到是哪台机器上的哪个应用程序[2]。

1 服务端高并发方案分析

在 Linux 环境下服务端实现高并发有三种模式：多线程、线程池、IO 复用模型。

1.1 多线程模式

单线程的服务端程序只能串行地处理客户端的请求，即一次只能为一个客户端服务。使用多线程的模式后，服务端程序可以并行处理客户端请求，同时为多个客户端服务，大大提高了服务端的处理效率。

1.1.1 Liunx 多线程接口

Linux 操作系统下多线程编程的一些接口，包括：线程创建 pthread_create，调用该函数后，程序除了在主线程中运行，同时在线程执行函数中运行；线程等待 pthread_join，执行该函数会等待某个线程结束；线程分离 pthread_detach，调用该函数让子线程相对主线程独立运行；线程退出 pthread_exit，调用该函数退出线程。

1.1.2 Linux 多线程流程

多线程通信示意，如图 1 所示。服务端先建立自己半相关，创建 socket，并将 socket 联编到某个端口上，并进入监听状态[3]。客户端会向服务端发起连接的要求，服务端的侦听套接字会接待所有到来的客户端，并给每个客户端找一个代理即连接套接字。此模式下就是一个客户端一线程，一线程一套

作者简介：王 林(1995-)，男，硕士研究生，主要研究方向：机器视觉与光电检测。

收稿日期：2020-03-23

接字，由这个套接字接收客户端的消息，服务端再把处理后的结果通过这个套接字反馈给客户端。

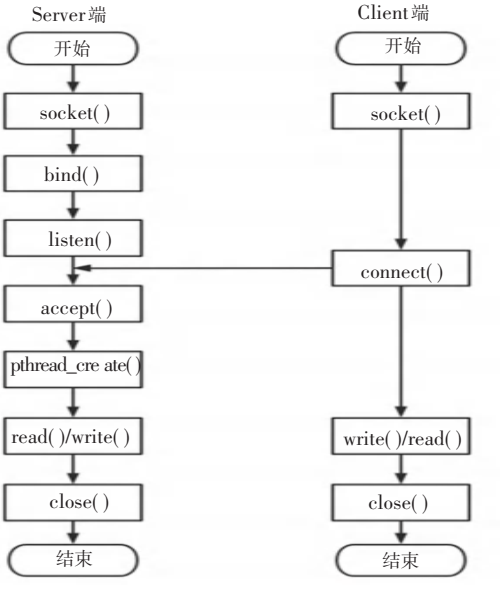


图 1　多线程通信示意图

Fig. 1　Multi thread communication diagram

此方法的优点是易于理解，操作简单；缺点是需要为每个客户端都创建一个线程来对其进行处理，当客户端数量过多时，就需要同时运行众多的线程，需要消耗大量的服务端资源。此外，建立线程和回收线程都需要时间，如果服务端不停创建和销毁线程，时间开销必然会导致服务端的响应速度变慢。

1.2　线程池模式

多线程的模式需要不停的建立与摧毁线程，虽然同生成进程相比，创建线程的时间已经很短了。但是这个操作被大量执行，依然会造成很多时间资源的浪费。而线程池提供的方案就是对线程资源的多次使用。

线程池的工作流程，如图 2 所示。主线程：新建一些线程于线程池中。开始，线程池中线程数量较少，在没有超过线程池中枢线程数量上限的情况下，只要有任务，就建立一个新线程。当线程数量达到线程池所允许的中枢线程数时，任务就被压入队列中等候。主线程向任务队列中添加任务，激活线程池中不忙碌的线程。当任务队列也被塞满时，如果此时线程数量没有超过线程池的最大线程允许数量，就创建线程去执行。

线程检测到任务队列中的任务数量不为零时，就从任务队列尾部取出任务节点，执行相应的任务，并将其从任务队列中删除。如果任务队列中任务数量为零，则让线程池中的线程休眠，等待主线程再次发送信号激活空闲的线程。

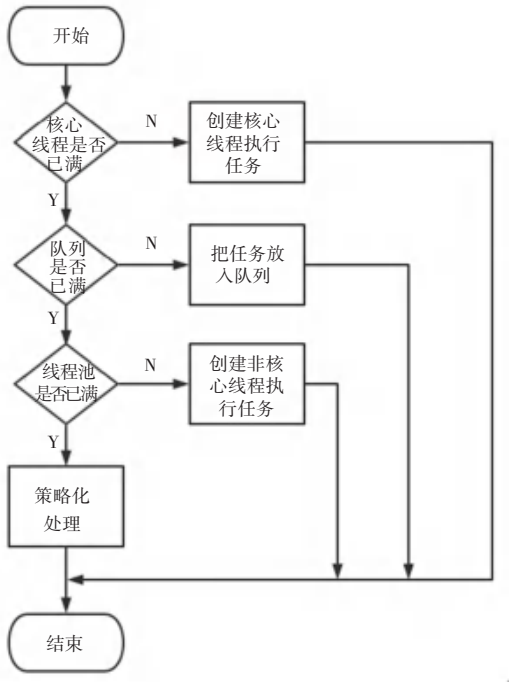


图 2　线程池工作流程图

Fig. 2　Workflow of thread pool

线程池的优点是节约了线程的创建和销毁所占用的时间，缺点是线程池适用于处理时间比较短的操作，当客户端和服务端完成通信后，客户端就会断开连接，从而回收线程。但如果客户端和服务端长期建立连接，就无法回收线程，线程池会很快被占满，使得程序无法运行。

1.3　IO 复用模式

线程的 IO 操作就是数据的读入或者写出，又或是对网络数据的请求。当服务端需要处理多个客户端的请求时，IO 复用可以用一个函数监听多个 I/O 操作，一旦某个操作需要处理，就通知程序进行相应的读或写。Linux 下常用的 I/O 复用模型有两种：select 模型和 epoll 模型。

1.3.1　select 模型

Select 的工作流程是：首先清空 fdset 集合，把侦听套接字加入集合中。使用 select 函数阻塞所有检测的套接字，一旦集合中有套接字就绪就去处理。使用 FD_ISSET 去遍历集合中的每个套接字，直到找出就绪的套接字。如果就绪的套接字是侦听套接字，则调用 accept 函数接受客户端的连接，将产生的新的套接字加入集合中。如果是与客户端连接的套接字则从套接字中读出数据或者向套接字里写入数据。

1.3.2　epoll 模型

epoll 模型是 linux 操作系统独有的 I/O 复用

函数,是 select 的进化版本,能让内核应对众多的文件描述符。

epoll 相比于 select/poll,有以下优点:

(1) select 支撑的文件描述符数量是有限的,而 epoll 能同时处理大量的 socket 描述符;

(2) selelct 每使用一次就要经历数据从用户空间到内核空间的一次拷贝,epoll 只在注册的时候把文件描述符拷贝到内核一次;

(3) select 只能判断有文件描述符就绪,但是并不知道具体是哪一个就绪了,需要逐个访问集合中的每个描述符,而 epoll 在描述符就绪时,会调用一个与此套接字关联的函数,把该套接字加入一个到位链表。根据这个返回的到位链表就可以确切知道究竟是哪些描述符需要程序去处理。

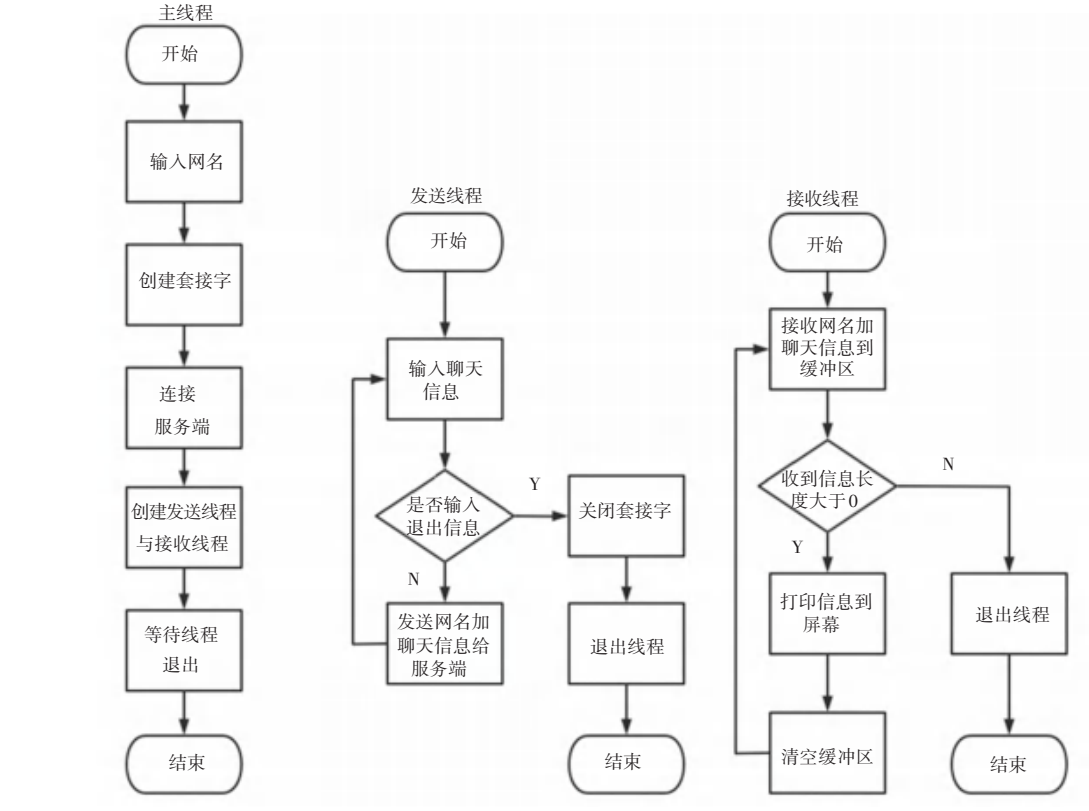
使用 epoll 模型一般分为三步:创建 epoll 描述符;注册 epoll 事件;最后等待 epoll 集合里的事件产生,并返回就绪的事件。

2 高并发网络聊天系统具体实现

由于客户端需要收发消息,要求创建接收消息的线程和发送消息的线程,来处理收发信息的异步问题[4]。而服务端需要在一个众所周知的地址监听客户对服务的请求[5]。所以本系统的客户端使用多线程的方式来执行任务,服务端使用 epoll 模型来管理多个客户端的请求,当有套接字就绪时则做出相应的处理,在主线程中等待连接,在子线程中处理与客户端之间的数据交互。

2.1 客户端实现流程

客户端的流程图如图 3 所示,其中(a)为主线程流程,(b)为发送线程流程,(c)为接收线程流程。在主线程中,第一步用户输入网名;第二步使用 WSAStartup 函数绑定 socket 库文件;第三步调用 socket()函数创建套接字,通讯模式设为 TCP 模式;第四步设置要连接的服务端 IP 和端口,端口号需要使用 htons()函数转换为网络字节序,这样可避免不同机器的兼容性问题;第五步客户端调用 connect()函数向服务端发起相连请求,如果连接成功则可以开始聊天;第六步,调用 beginthreadex()函数创建发送线程和接收线程,此时客户端的两个线程运行,既可以接收信息,也可以发送信息;最后,在 main 函数中调用 WaitForSingleObject()函数,等候发送和接收线程的结束。两个线程结束后,使用 closesocket()关闭客户端的套接字。



(a) 主线程流程 (b) 发送线流程 (c) 接收线程流程

(a) Main thread process (b) Sending thread process (c) Receiving thread process

图 3 客户端流程图

Fig. 3 Client flow chart

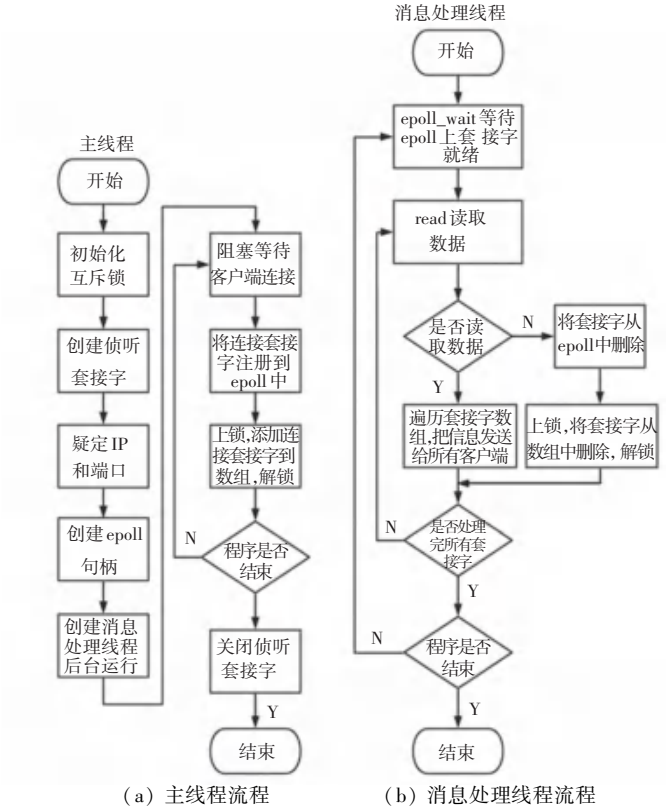
在客户端的发送线程中，用户可以循环输入聊天信息，客户端将用户的网名和聊天信息一起发送给服务端。如果用户输入退出信息，则关闭客户端套接字并退出发送线程。

在客户端接收线程中，用户循环接收服务端发来的信息（包括网名加聊天内容），将信息输出到屏幕，并清空缓冲区，为下次接收信息做准备。如果收到信息的字节数为零，则表示无数据可接收，退出客户端接收线程。

2.2　服务端实现流程

图 4 为服务端流程图。其中（a）为主线程流程，（b）为消息处理线程流程。在主线程中，（1）调用 pthread_mutex_init（）完成互斥锁的创建；（2）调用 socket（）创建侦听套接字，通讯模式设为 TCP；（3）设置服务端机器标识号以及应用程序号，捆绑套接字捆绑；（4）侦听套接字；（5）采用 epoll_create（）创建一个 epoll 实例；（6）使用 pthread_create（）创建消息处理线程，并使用 pthread_detach（）让该线程独立于主线程；（7）调用 accept（）阻塞等待客户端的连接，并将产生的连接套接字设置为非拥堵形式，放入到 epoll 监听集合中，注册事件为是否可读；（8）将连接套接字添加到客户端套接字数组中，由于数组是共享资源，所以对数组操作前需要上锁，操作完成再解锁。当服务端退出时，关闭服务端的侦听套接字。

消息处理线程中，调用 epoll_wait（）等待套接字就绪，并返回就绪的套接字数目，逐个处理每个就绪的套接字。由于监听的事件类型为是否可读，所以当套接字就绪时，调用 read（）去套接字中读出数据到缓冲区中，遍历客户端套接字数组，使用 write（）把接收到的消息发送给每一个客户端。如果没有读取到数据，则把该套接字从 epoll 中以及数组中删除。



（a）主线程流程　　（b）消息处理线程流程

（a）Main thread process　　（b）Message processing thread process

图 4　服务端流程图

Fig. 4　Server flow chart

3　结束语

本文分析了实现服务端高并发的三种模式，分别是多线程，线程池与 IO 复用。经过对比，最终选择并使用 IO 复用模式中的 epoll 模型来实现服务端的开发。采取 C/S 架构，使用套接字编程技术进行网络通信，设计实现了一个简易的高并发局域网聊天系统。

参考文献

[1] 丁乐. 高并发高性能网络技术研究及在网络聊天室的应用[D]. 广州：华南理工大学，2018.
[2] 董敏，桑建建. 基于 Linux 的网络聊天系统的设计[J]. 信息与电脑（理论版），2017(23)：142-143.
[3] 张龙. 聊天系统的设计与实现[D]. 大连：大连理工大学，2015.
[4] 董克基. 基于 SOCKET 的网络聊天系统分析设计[J]. 电子世界，2014(16)：189.
[5] 萧泳东，肖化. 基于 Linux 的网络聊天系统设计[J]. 现代电子技术，2013，36(3)：51-54，57.