DOI：10.13875/j.issn.1674-0637.2020-01-0018-11

基于多核处理器的NTP服务器设计

刘岩[1,2,3,4]，武建锋[1,2,3,4]，王康[1,2,3]

（1. 中国科学院 国家授时中心，西安 710600；
2. 中国科学院 时间频率基准重点实验室，西安 710600；
3. 中国科学院 精密导航定位与定时技术重点实验室，西安 710600；
4. 中国科学院大学，北京 100049）

摘要：响应能力是NTP（network time protocol）服务器的一项重要性能指标。为了提高响应能力，基于多核处理器设计了一款NTP服务器，并对不同内核数下服务器的响应能力进行了试验分析，得到了服务器内核数与响应能力之间的定量关系，为多核NTP服务器的应用奠定了基础。

关键词：网络时间服务器；多核处理器；高响应能力

Design of NTP server based on multi-core processor

LIU Yan[1,2,3,4], WU Jian-feng[1,2,3,4], WANG Kang[1,2,3]

(1. National Time Service Center, Chinese Academy of Sciences, Xi'an 710600, China;
2. Key Laboratory of Time and Frequency Primary Standards, Chinese Academy of Sciences, Xi'an 710600, China;
3. Key Laboratory of Precise Positioning and Timing Technology, Chinese Academy of Sciences, Xi'an 710600, China;
4. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Responsibility is an important performance index of NTP (network time protocol) server. In order to improve the responsibility, a NTP server is designed based on multi-core processor in this study. The responsiveness of servers with different numbers of core is tested and analyzed, and the quantitative relationship between the numbers of server kernel and the responsibility is obtained. It lays the foundation for the application of multi-core NTP server.

Key words: NTP server; multi-core processor; high responsibility

0 引言

NTP（network time protocol），网络时间协议，是互联网TCP/IP模型中应用层的一种协议，它定义了一种基于网络的时间同步方法[1]。NTP是由美国特拉华大学的D. L. Mills教授提出的，目前已经发展到

收稿日期：2019-07-09；接受日期：2019-09-02
基金项目：装备发展部军用标准资助项目（Y833ZX1601）；中国科学院“西部之光”人才培养计划西部青年学者B类基金资助项目（Y607YR8601）
作者简介：刘岩，男，硕士，主要从事NTP网络时间服务器的研究。

第 4 版[2]。另外，NTP 既能应用于局域网内，又能用于广域网内，时间同步精度在广域网内能达到几十个毫秒，在局域网内能达到毫秒级[3-4]。NTP 应用范围广，同步精度可以满足大众要求，因此获得了广泛的应用。

NTP 在应用时通常采用客户端/服务器模式，基本系统组成结构如图 1 所示，包括一台服务器和多个客户端。客户端通过向服务器发送请求并接收服务器的应答，即可将本地时间同步到服务器。服务器从 GNSS（Global Navigation Satellite System）卫星、天文台等获取标准时间 UTC（universal time coordinate），为系统提供准确的时间参考[5]。系统内各个客户端分别将时间同步到服务器，以实现彼此之间的时间统一。

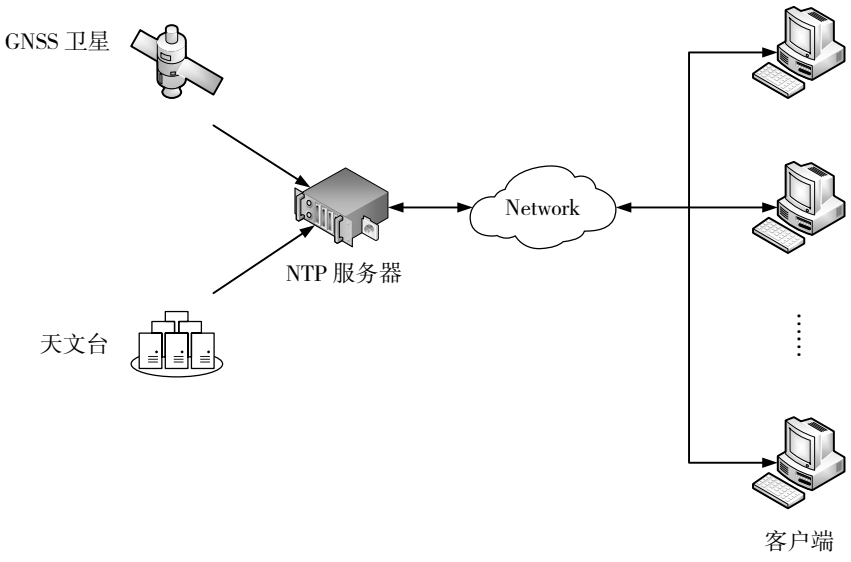

图 1　NTP 时间服务结构

如今，随着各种分布式系统规模不断扩大，系统内客户端数量越来越多。因此，对 NTP 服务器的请求响应能力提出了更高的要求。响应能力可以以单位时间内服务器能够响应的请求次数来定量的表征[6]。在互联网网络授时项目中要求实现 NTP 服务器响应能力不低于 30 000 次/s。

在 NTP 的时间同步过程中，服务器处理请求的流程可分为 3 个阶段：接收请求、处理数据、发送应答[7]。在多核处理器出现之前，NTP 服务器采用串行工作方式，当收到多个请求时，服务器只能一个一个地处理，不断地重复上述处理流程。当时提高服务器的响应能力只能靠使用更高主频的处理器来实现，而更高的主频将产生更多的热量[8]，因此，处理器的主频也就无法无限制的提高。多核技术的出现使得某一时刻多个处理器内核能同时工作。如果在每一个内核上分别运行一个客户端请求处理程序，使服务器以并行方式工作，这样在相同时间内就可以处理更多的请求[9]。比如，图 2 描述了使用两个内核时，并行方式与串行方式的请求处理过程的比较。在并行方式中，当某个内核上的处理过程处于处理数据阶段，另一内核上的程序可以去接收请求，这样就能缩短多个请求的总的处理时间，从而提高服务器的请求响应能力。

另外，在串行处理方式中，一些其他功能，比如服务器时间到 UTC 的同步，也会占用请求处理时间；而在并行方式中，这些功能可以单独运行在一个内核上，这样可以进一步提高请求响应能力。

本文使用一种多核处理器设计了一款 NTP 服务器，以满足项目对响应能力指标的要求，同时，对内核数与响应能力之间的关系进行了研究，为多核 NTP 服务器的应用奠定了基础。

图 2 串行和并行方式处理过程对比

1 NTP 时间同步原理

NTP 的时间同步过程是由客户端发起的。当需要校准本地时钟时，客户端主动向服务器发送一个请求报文，报文内容可分为控制字段和 3 个时间戳[10]。如图 3 所示，客户端发送请求时，记录下发送时刻的时间戳，在图中以 T_1 表示，T_1 被写入请求报文中。服务器收到请求后，首先记录下报文接收时刻的时间戳 T_2，接着修改报文的控制字段，最后向客户端发送应答报文并记录下发送时刻的时间戳 T_3，T_2 和 T_3 同样被记录在报文中。客户端在收到服务器的应答时，再记录下报文接收时刻的时间戳 T_4。其中，T_1 和 T_4 从客户端时钟读取，T_2 和 T_3 从服务器时钟读取。

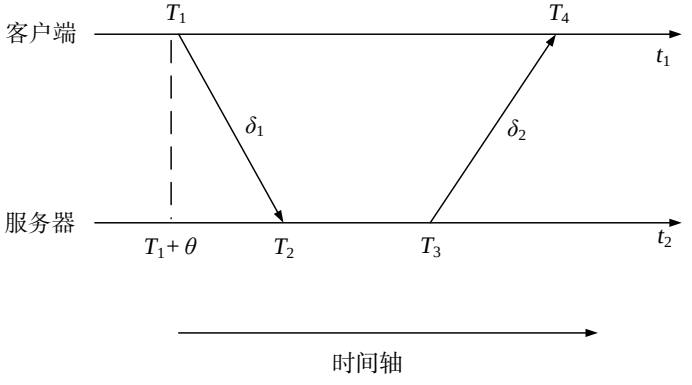

图 3 客户端/服务器的时间同步过程

客户端根据这 4 个时间戳可以得出下面 3 个等式：

$$T_2 - T_1 = \theta + \delta_1 , \tag{1}$$

$$T_4 - T_3 = \delta_2 - \theta , \tag{2}$$

$$\delta_1 + \delta_2 = \delta , \tag{3}$$

式（1）至（3）中，θ 为客户端与服务器间的时间偏差，δ_1 为报文由客户端到服务器的网络路径时延，δ_2 为反向路径时延，δ 为往返总的路径时延。假设往返路径时延相同，即 $\delta_1 = \delta_2 = \frac{\delta}{2}$，则以上 3 个等式变为：

$$T_2 - T_1 = \theta + \frac{\delta}{2} , \tag{4}$$

$$T_4 - T_3 = \frac{\delta}{2} - \theta 。 \tag{5}$$

由式（4）和式（5）可以求出：

$$\theta = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}, \tag{6}$$

$$\delta = (T_4 - T_1) - (T_3 - T_2)。 \tag{7}$$

客户端根据计算出的时间偏差 θ 调整本地时钟，从而将时间同步到服务器[2]。

2　服务器设计

服务器的组成结构如图 4 所示，其结构包括以下几个单元[11]：

网络通信单元：负责与客户端通过网络传输报文，实现方式是基于标准的以太网接口硬件和 TCP/IP 通信协议进行网络编程；

数据处理单元：负责处理客户端的请求数据并对其他单元进行控制，通过在处理器上编程来实现；

时钟单元：负责保证服务器时间与 UTC 一致，为客户端提供准确的时间参考，包括守时模块和时间源模块，守时模块负责维持一个本地时间，时间源模块用于获取 UTC 并校准守时模块的时间。

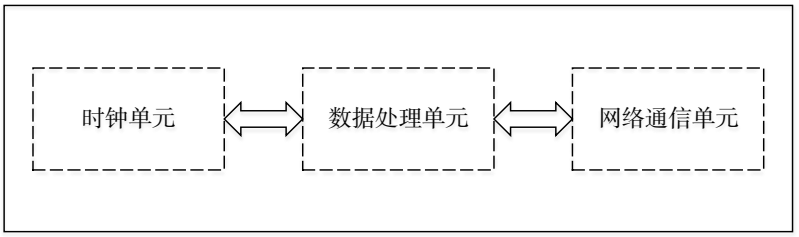

图 4　服务器组成结构

2.1　硬件设计

图 5 所示是本设计的硬件组成结构以及各部分之间的连接方式。

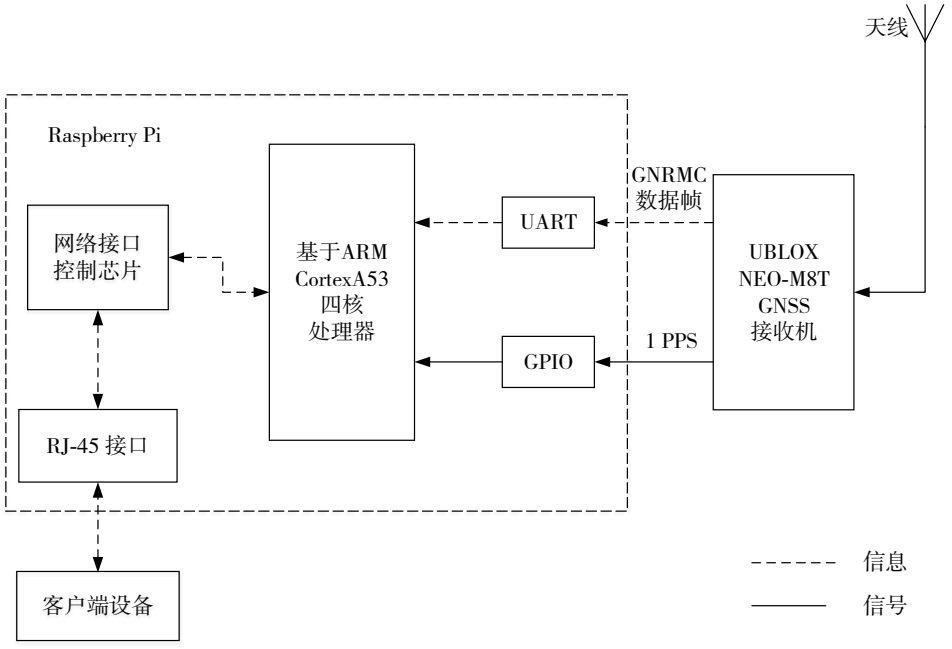

图 5　服务器的硬件组成

网络通信单元和数据处理单元的硬件整合在一种 Raspberry Pi 3B 开发板上。Raspberry Pi 是一种基于 ARM 的微型电脑主板，同时能够运行 Linux 操作系统，具备 PC 所有的基本功能。本设计使用的 Raspberry Pi 3B 采用一种基于 ARM Cortex-A53 的处理器芯片，具有 4 个内核，主频 1.2 GHz，同时拥有 1 GB 的内存。

时钟单元的守时模块采用 Raspberry Pi 上 Linux 的系统时钟，精度可以达到 1 ns[12]。时间源模块硬件使用 UBLOX NEO-M8T GNSS 接收机。UBLOX NEO-M8T 可以从 GNSS 卫星获取 UTC，同时能够输出用于时间校准的时码信息和 1 PPS 信号[13]。Raspberry Pi 分别通过串口和 I/O 口读入 GNRMC 数据帧和 1 PPS 信号，以校准系统时钟。Raspberry Pi 和 UBLOX NEO-M8T 的接口均为 TTL 电平，无需转换。

2.2 软件设计

对应于服务器的 3 个组成单元，软件可分为网络通信程序模块、数据处理程序模块和时间校准程序模块。其中重点是基于多核处理器的请求数据处理程序设计。

2.2.1 网络通信程序设计

服务器在接收和发送 NTP 报文时涉及到网络编程。网络通信过程可以使用 TCP/IP 5 层模型来描述，同时在应用层和运输层之间存在一个 socket 抽象层。socket 将应用层以下的逻辑封装起来，并为用户程序提供了接口函数，用户只需要调用相关函数即可实现网络通信[14]。

图 6 所示为客户端和服务器之间通信程序的执行流程。首先两端都需要调用 socket( )函数创建一个基于 UDP 协议的网络连接，同时服务器还需要使用 bind( )函数绑定本机的 IP 地址和端口。接着服务器使用 recvfrom( )函数接收客户端的请求报文，此函数在没有请求到来时处于阻塞、监听状态，不占用处理器资源。recvfrom( )函数在接收数据的同时还可以得到客户端的 IP 地址和端口。当客户端需要校准时钟时，通过 sendto( )函数向服务器发送请求报文，sendto( )函数可以向指定的 IP 地址和端口发送数据。收到请求后，服务器对报文进行处理，最后调用 sendto( )函数向客户端发送一个应答报文。客户端完成时间同步后，可以关闭 socket 连接，而服务器则要回到监听状态，等待下一个请求[15]。

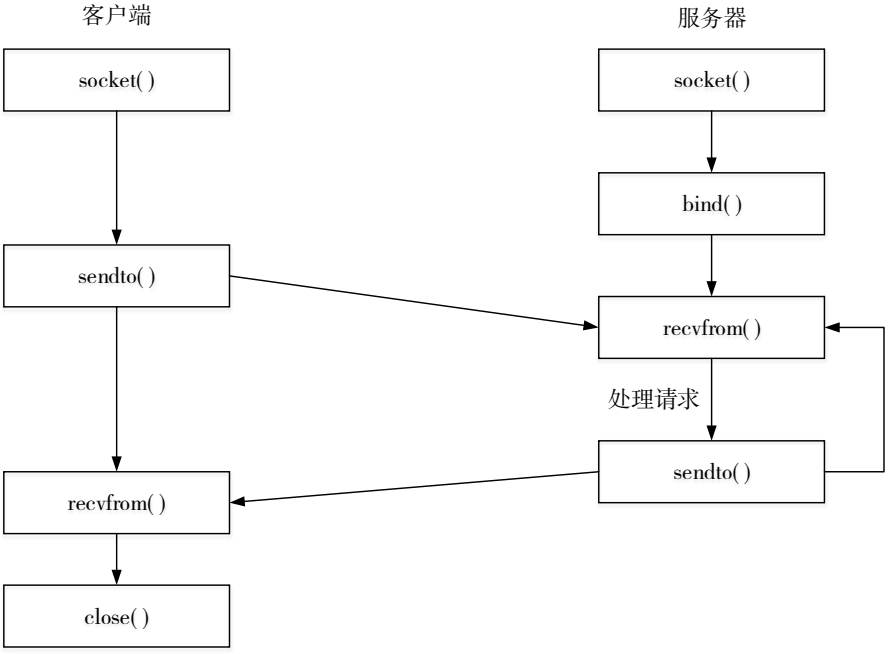

图 6 客户端/服务器通信程序流程

2.2.2　数据处理程序设计

数据处理程序的流程图如图 7 所示。程序首先建立网络连接、绑定 IP 地址和端口。接着接收客户端的请求报文，为了尽可能地保证时间戳的准确性，收到报文后立刻记录下接收时刻，之后对报文格式进行判断，如果报文格式正确，则对报文进行处理，最后向客户端发送应答报文，否则将报文直接丢弃，不予处理。

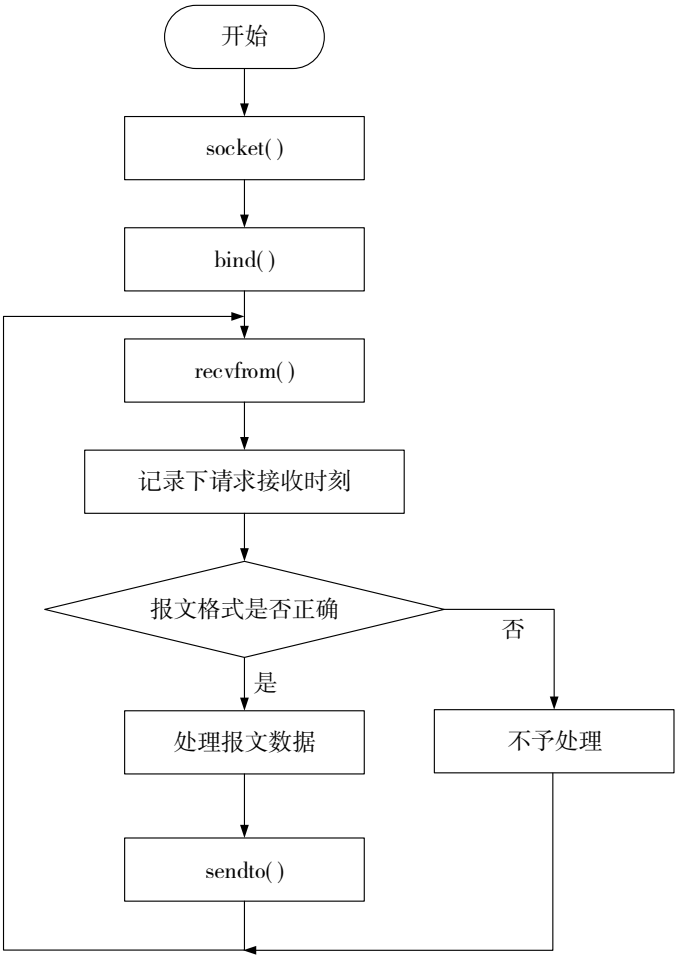

图 7　单线程程序流程

为了利用多核处理器，程序采用多线程的设计方法[16]。每个线程执行同样的处理流程，并设置为分别在不同的内核上运行。通过设置线程数，即可实现对不同内核数的利用。协议规定 123 作为 NTP 服务器的通信端口，由于一个进程只能绑定到一个端口，因此程序采用在一个进程下设置多个线程的方式，而不是使用多进程[17]。图 8 所示是双线程的流程图，多线程的原理与双线程相同。主线程建立网络连接、绑定 IP 地址和端口后，创建一个子线程。子线程与主线程共享同一个 socket 接口，接着执行与图 7 同样的请求处理流程。

服务器通过网络接收请求报文时，报文首先进入 socket 的接收缓存区，然后用户程序到这个缓存区中读取报文；发送应答时，也是将报文写入到 socket 的发送缓存区。由于多个线程共享同一个 socket 接口，因此，某一时刻不能有两个线程同时调用 recvfrom( )或 sendto( )函数访问 socket 接收或发送缓存区。为此，程序采用锁机制，线程在调用 recvfrom( )或 sendto( )函数前必须获得锁的权限，而某一时刻只能有一个线程拥有权限，其他线程只能排队等待，执行完 recvfrom( )或 sendto( )函数后再将锁释放，这样就可以保证同一时刻只有一个线程能够访问 socket 的接收或发送缓存区[17]。

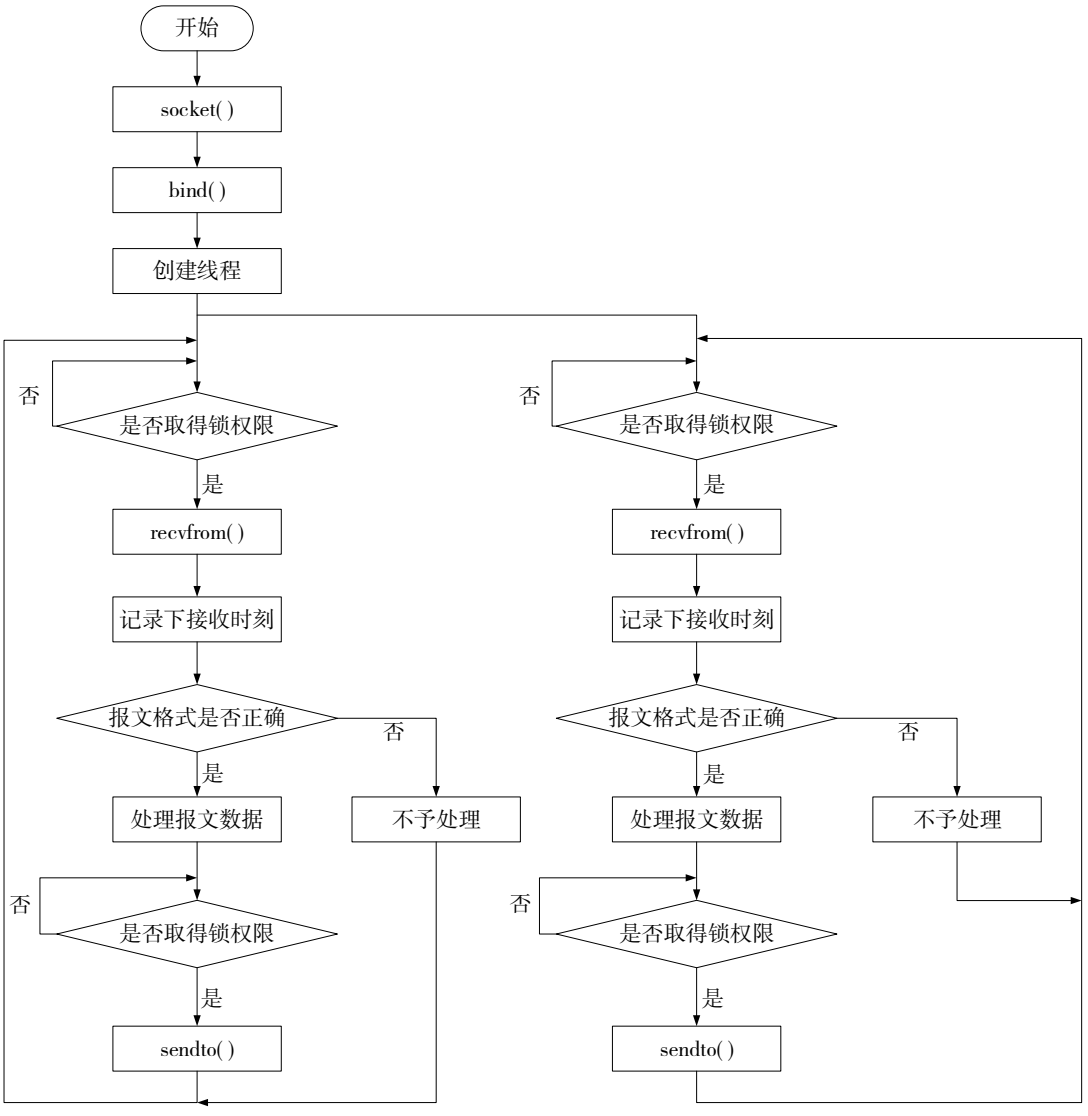

图 8 双线程程序流程

2.2.3 时间校准程序设计

时间校准程序被设置为一个线程，独立运行，不占用请求处理资源。

Linux 中含有一个系统时钟,精度可以达到纳秒级。Linux 内核为用户提供两个接口函数 clock_gettime( ) 和 clock_settime( )，前者可以从系统时钟中获取纳秒级时间，后者可以修改系统时钟，精度也在纳秒级。

UBLOX NEO-M8T 每秒在输出 GNRMC 数据帧的同时输出一个 1 PPS 信号，两种波形如图 9 所示。

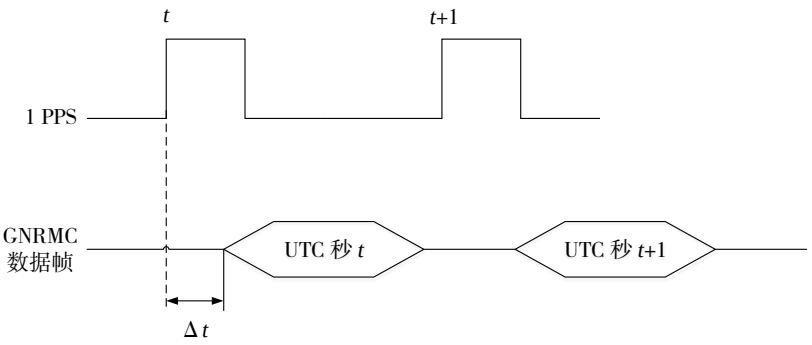

图 9 GNRMC 数据帧和 1 PPS 信号波形

1 PPS 信号的脉冲上升沿与 UTC 的秒起始时刻对齐，而 GNRMC 数据帧的输出滞后于秒起始时刻[18]。

Raspberry Pi 通过串口读入 GNRMC 数据帧，读入的数据帧可能出现以下情况：

① 由于 UBLOX NEO-M8T 接收机未收到 GNSS 卫星信号，导致输出的数据无效，无效数据形如：$GNRMC，081 253.00，V，…，270 219，…，N*61，接收机刚启动时就会发生这种情况；

② 信号接收过程中受到干扰，使接收到的数据与卫星发送的数据不一致；

③ Raspberry Pi 串口接收数据帧时出现错误，比如丢失位。

因此，在提取 UTC 信息之前，需要对收到的 GNRMC 数据帧进行校验。校验过程通过软件方式来实现，校验程序每次从硬件读入一个字符，首先要找到起始字符‘$’，然后判断接下来收到的字符是否依次是‘G’，‘N’，‘R’，‘M’，‘C’，如果中间插入其他字符，说明收到的数据有误，不再接收下面的字符，而是重新寻找新一帧的起始字符。程序还要判断是否收到表示数据帧无效的字符‘V’。如果未出现上述两种情况，那么将接收所有字符，最后对数据进行校验，校验通过则从中提取出 UTC 的秒以上时间[19]。

Raspberry Pi 通过串口每秒从 UBLOX NEO-M8T 接收机读取一次 UTC，通过 I/O 口接收 1 PPS 信号。1 PPS 的上升沿到达时，说明此时是 UTC 1 秒的开始。利用此信号的上升沿触发处理器中断，在中断处理程序中，首先根据串口读入的 UTC 校准系统时钟的秒级时间，然后将纳秒级时间归零，即可实现服务器时间到 UTC 的同步。由于每秒的 GNRMC 数据帧滞后于 1 PPS 信号，所以当前 1 PPS 上升沿引起中断时，串口读入的最新秒值对应于上一个 1 PPS 信号，所以应将当前秒值加 1 后再调整系统时钟的秒级时间。

3　服务器性能测试

服务器设计完成后，首先对时间同步精度进行了测试，以确保能达到基本要求；然后分别对不同内核数下的响应能力进行了测试，以对内核数与响应能力的关系进行分析。

3.1　时间同步精度测试

服务器的时间同步精度使用 TimeAcc-007 进行测试。测试时，TimeAcc-007 通过网线直接与服务器连接，以每秒一次的频率向服务器发送请求，一共采集了 10 000 组数据。测试结果如图 10 所示，同步精度的均值为 16.5 μs，标准差 6.93 μs，均方差 17.9 μs。

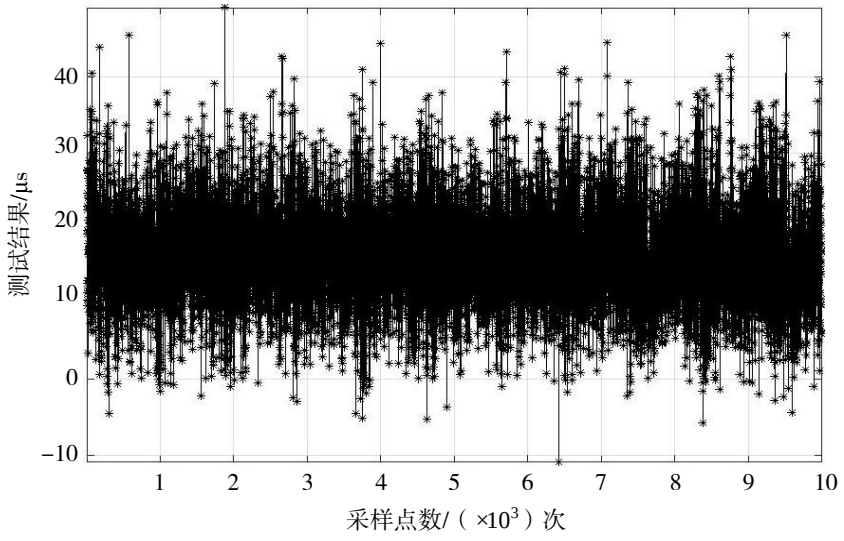

图 10　时间同步精度测试结果

3.2 响应能力测试

测试服务器的响应能力时，设计了一个多客户端模拟程序。程序运行在测试计算机上，可以在单位时间内向服务器发送一定数量的请求数据包，通过 wireshark 软件可以统计服务器每秒的响应次数。测试时，程序从 5 000 次/s 开始，以 5 000 次/s 间隔递增，向服务器发送请求，每组进行 20 次测试[20]。

测试结果如表 1 所示，在使用单核处理器时，当请求量小于 25 000 次/s 时，服务器可以响应全部请求；而当请求量增加到 30 000 次/s 时，响应量的均值在 26 272 次/s 左右，图 11 所示是 20 次测试结果曲线图。

表 1 不同请求量的响应结果 单位：次/s

请求量	单核的响应量	双核的响应量	三核的响应量	四核的响应量
5 000	5 000	5 000	5 000	5 000
10 000	10 000	10 000	10 000	10 000
15 000	15 000	15 000	15 000	15 000
20 000	20 000	20 000	20 000	20 000
25 000	25 000	25 000	25 000	25 000
30 000	26 272	30 000	30 000	30 000
35 000	–	35 000	35 000	35 000
40 000	–	40 000	40 000	40 000
45 000	–	44 102	45 000	45 000
50 000	–	44 102	46 027	46 098

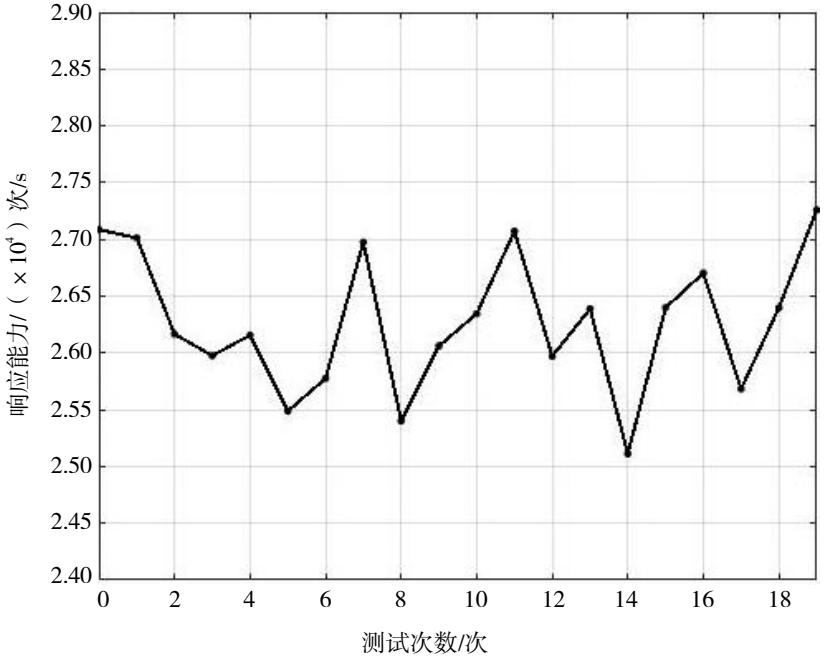

图 11 单核下 30 000 次/s 的响应结果

双核的响应能力均值在 44 102 次/s 左右；三核的响应能力均值在 46 027 次/s 左右；四核的响应能力均值在 46 089 次/s 左右，图 12 所示是双核、三核和四核的测试结果曲线。表 2 给出了不同内核数下响应能力的最小值、最大值和均值的详细结果。

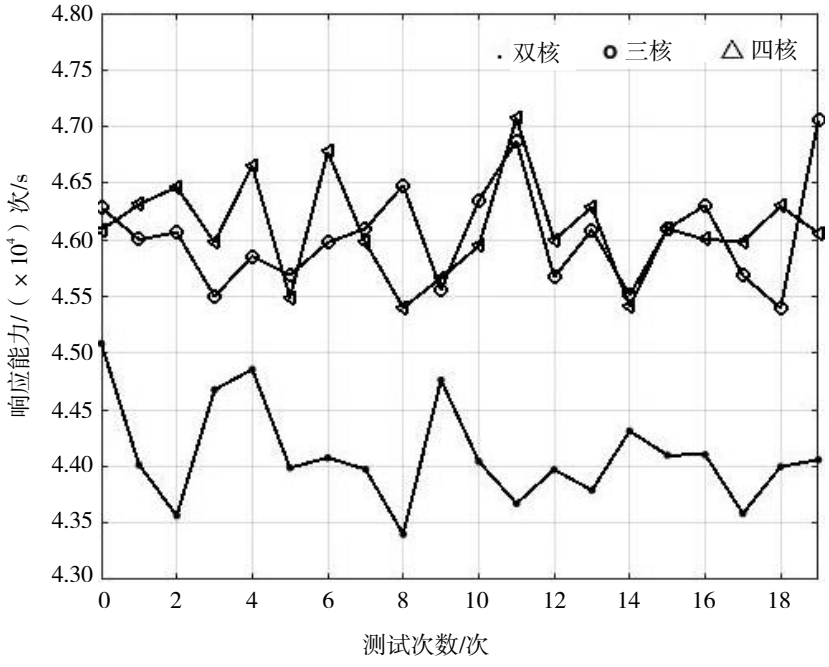

图 12　双核、三核和四核下 50 000 次/s 的响应结果

表 2　不同内核数的响应能力　　单位：次/s

	最小值	最大值	均值
单核	25 112	27 259	26 272
双核	43 401	45 086	44 102
三核	45 397	47 059	46 027
四核	45 401	47 072	46 098

4　结语

论文首先对 NTP 的时间同步原理进行了分析，确定了服务器的请求处理流程；接着介绍了基于多核处理器的 NTP 服务器设计方法，包括服务器的硬件介绍和软件设计，重点是在多线程程序的设计，同时还包括时钟单元的程序设计；最后对服务器的性能进行了测试，包括服务器的时间同步精度和不同内核数下服务器的响应能力。测试结果表明：服务器的时间同步精度约为 16.5 μs；使用双核处理器时，响应能力可以达到 40 000 次/s，能够满足项目的 30 000 次/s 的指标要求。此外，相比于单核，双核处理器的响应能力能够提高约 67.9%；而相比于双核，三核、四核的响应能力只提高了 4.4%左右。而目前 NTP 服务器可扩展的功能越来越多，比如增加校验加密算法。虽然三核和四核处理器对响应能力没有明显提高，但可以用于扩展其他功能，同时不会降低服务器的响应能力。

参考文献：

[1] MILLS D L. Network Time Protocol Specification, Impementation and Analysis-RFC 1305[S]. 3rd. 1992.

[2] 陈敏. 基于 NTP 协议的网络时间同步系统的研究与实现[D]. 武汉：华中科技大学, 2005.

[3] 陈希, 滕玲, 高强, 等. NTP 和 PTP 协议的时间同步误差分析[J]. 宇航计测技术, 2016, 36(3): 35–40.

[4] 赵龙. 基于 NTP 协议的网络授时研究[D]. 阜新：辽宁工程技术大学, 2006.

[5] 黄沛芳. 基于 NTP 的高精度时钟同步系统实现[J]. 计算机技术与应用, 2009, 35(7): 122–124+127.

[6] 吴鹏. NTP 授时服务性能监测及状态评估[D]. 北京：中国科学院大学, 2016.

[7] 宋妍, 朱爽. 基于 NTP 的网络时间服务系统的研究[J]. 计算机工程与应用, 2003, 39(36): 147–149+152.

[8] 彭岚, 周启海. UNIX 并发服务器及其几种优化改进方案[J]. 计算机应用, 2004, 24(4): 47–49.

[9] 李晓飞, 李开毅, 周洪运. 并发编程原理分析[J]. 玉溪师范学院学报, 2008, 24(4): 44–46.

[10] 胥婕, 徐亮, 董莲, 等. 基于 NTP 协议的网络时间同步系统的设计与实现[J]. 上海计量测试, 2017, 44(4): 10–13.

[11] 王瑞清. 嵌入式高精度 NTP 网络时间服务器研究与实现[D]. 武汉：华中科技大学, 2011.

[12] 李明国, 宋海娜. 计算机时间同步技术研究[J]. 系统仿真学报, 2002, 14(4): 477–480.

[13] 贺洪兵. 基于 GPS 的高精度时间同步系统的研究设计[D]. 成都：四川大学, 2005.

[14] 邓钦文. 基于 ARM 和 Linux 的嵌入式 Web 服务器研究与实现[D]. 长沙：湖南大学, 2010.

[15] 王雷, 王子淘. 基于 Linux 的 Socket 网络编程的性能优化[J]. 电子设计工程, 2009, 17(9): 101–103.

[16] 刘仕筠, 盛志伟, 黄健. Linux 环境并发服务器设计技术研究[J]. 成都信息工程学院学报, 2006, 21(5): 630–634.

[17] 申时全. 基于 Linux 多线程技术的网络并发编程及应用研究[J]. 现代计算机, 2016, 31(16): 65–70.

[18] 张治炼. 基于 GPS 授时的本地同步时钟的设计[D]. 成都：电子科技大学, 2012.

[19] 李向宇, 邹源忠, 刘玉菊. 基于 GPS 自动校正时间电子钟的设计[J]. 信息技术与信息化, 2017(3): 33–35.

[20] 吴鹏, 华宇, 张旭海. NTP 服务器的响应阈值测试软件设计[J]. 时间频率学报, 2017, 40(1): 36–42.