基于 Linux 系统的构建高性能服务器的研究*

乔平安　颜景善　周　敏
（西安邮电大学计算机学院　西安　710121）

摘　要　研究了在 Linux 环境下构建高性能服务器的关键技术，Reactor，Proactor 事件处理模式，处理并发访问量的并发模式，考虑到现代硬件技术的发展，以空间换时间的思想构建高性能服务器，在理论研究的基础上，最后在 Linux 环境下设计了一个 Web 服务器，构建过程中充分利用了研究的高性能知识，并通过压力测试进行仿真验证，并分析仿真数据，证明构建的服务器是可以承受一定的高并发访问的。

关键词　高性能服务器技术；事件处理模式；并发模式；线程池；进程池
中图分类号　TP302.1　　**DOI**：10.3969/j.issn.1672-9722.2016.04.021

Construction of High Performance Server System Based on Linux

QIAO Pingan　YAN Jingshan　ZHOU Min
(School of Computer, Xi'an University of Posts and Telecommunications, Xi'an　710121)

Abstract　The key technology of high performance server construction based on the Linux, Reactor and Proactor event handling model, concurrent processing mode facing to amounts of accessing are introduced. Taking into account the development of modern hardware technology, so a high performance server space is put forward for time in thought. Making full use of the high performance knowledge, finally a Web server of high performance is designed. According to the experiment results, the Web server is constructed to withstand certain high concurrent accessing.

Key Words　technolgy of high performance server, event handling model, concurrent model, thread pool, process pool
Class Number　TP302.1

1　引言

随着因特网的发展，互联网的使用人数及信息量的爆炸式增长，对于服务器稳定高效的处理高并发请求越来越重要[1]，例如 Facebook 在 2012 年拥有 8.367 亿的独立访问者。如果对于同时大量的访问，不能快速处理，那么对于企业来说将是致命的打击，导致客户的大量流失。所以构建高性能服务器显得越来越重要。通过对在 Linux 环境下构建高性能服务器研究的基础上，运用所研究的知识构建一个 Web 服务器，并进行压力测试。所研究的构建高性能服务器的知识主要包括：在 Linux 环境下的 I/O 模型以及两种高效的事件处理模式，两种并发处理模式以及在半同步/半异步模式下衍生出的半同步/半反应堆模式，根据现在服务器行业的发展，硬件的发展速度明显高于软件，针对这种特点则有了以空间换取时间的思想的提出，运用在高性能服务器上的具体表现形式则是进程池，线程池、内存池、连接池的提出及应用，运用高性能的定时器可以提高服务器性能，构建稳定的服务器可以运用统一事件源的思想。

2　理论方法

构建高性能服务器的理论方法如下。

2.1　Linux 环境下 I/O 模型

选取合适的 I/O 模型对服务器的性能有着重要的影响，Linux 环境下 I/O 模型[2]分为四种：

1）阻塞 I/O，程序阻塞于读写函数。

* **收稿日期**：2015 年 9 月 1 日，**修回日期**：2015 年 10 月 27 日
作者简介：乔平安，男，副教授，硕士生导师，研究方向：数据库技术、计算机网络。颜景善，男，硕士研究生，研究方向：Web 服务器开发。周敏，女，硕士研究生，研究方向：计算机网络。

2) I/O 复用,程序阻塞于 I/O 复用系统调用,但可以同时监听多个 I/O 事件。

3) SIGIO 信号,信号触发读写就绪事件,用户程序执行读写操作,程序没有阻塞阶段。

4) 异步 I/O,内核执行读写操作并触发读写完成事件。程序没有阻塞阶段。

由各个 I/O 模型的特点可知,对于高性能服务器处理高并发访问运用的是 I/O 复用技术以及异步 I/O 技术[3]。

在 Linux 环境下的 I/O 复用技术包括:select、poll、epoll[4]系列函数。三者的区别如下:

1) 事件集合:对于 select,用户通过三个参数分别传入感兴趣的可读、可写及异常等事件,内核通过对这些参数在线修改来反馈其中的就绪事件,这使得用户每次调用 select 都要重置这三个参数。对于 poll,统一处理所有事件类型,只需要一个事件集参数,用户通过 pollfd. events 传入感兴趣的事件,内核通过修改 pollfd. revents 反馈其中就绪的事件。对于 epoll,由内核事件表直接接管用户感兴趣的所有事件,因此每次调用 epoll_wait 时,无须反复传入用户感兴趣的事件。epoll_wait 系统调用的参数 events 仅用来反馈就绪的事件。

2) 应用程序索引就绪文件描述符的时间复杂度:select:O(n),poll:O(n),epoll:O(1)。

3) 最大支持文件描述符数:select:一般有最大值限制。poll,epoll 取决于硬件。

4) 工作模式:select,poll:工作在 LT 模式下,epoll:支持 ET 高效模式。

5) 内核实现和工作效率:select,poll:采用轮询方式来检测就绪事件,算法时间复杂度为 O(n),epoll:采用回调方式来检测就绪事件,算法时间复杂度为 O(1)。

经过以上分析可知处理大规模高并发的数据访问用 I/O 复用中的 epoll[5]模型优势明显。

异步 I/O 技术:注册的是读完成或写完成事件,真正的读写由内核完成,效率高,主要的函数包括 aio_read,aio_write,异步 I/O 技术也可用于高并发的数据访问。

2.2 事件处理模式

尽管服务器的种类众多,但是其基本框架都是一样的[6],基本框架如图 1 所示。

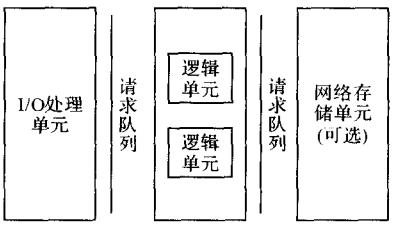

图 1 服务器基本框架

• 逻辑单元:对于单个服务器程序:一个逻辑单元通常是一个业务进程或线程,处理并分析客户数据。对于服务器集群:一个逻辑单元通常是一台逻辑服务器,集群有多台逻辑服务器,实现对客户请求的并行处理。

• 网络存储单元:对于单个服务器程序,是本地数据库,文件或缓存。对于服务器集群,是数据库服务器。

• 请求队列:对于单个服务器程序,是各单元间的通信方式。对于服务器集群,是各服务器间永久的 TCP 连接。

构建高性能服务器的网络设计模式有两种:Reactor 和 Proactor 事件处理模式。图 2 为 Reactor 处理模式,图 3 为 Proactor 处理模式[2]。

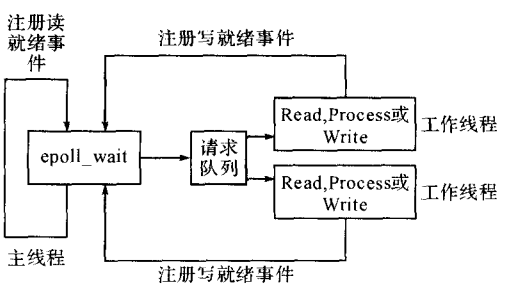

图 2 Reactor 处理模式

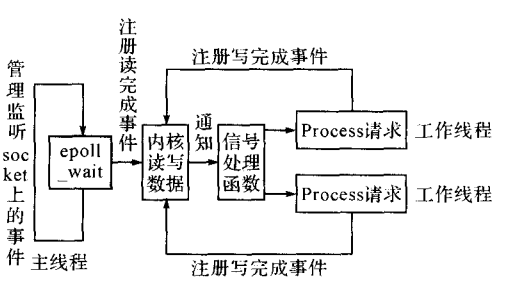

图 3 Proactor 处理模式

2.3 并发处理模式

并发模式是指 I/O 处理单元和多个逻辑单元之间协调完成任务的方法。服务器并发处理模式包括两种:半同步/半异步模式,领导者/追随者模式[7]。

高效的半同步半异步模式:在主线程中管理监听 socket,当有连接到来时,则接受连接,并将连接 socket 派发到工作线程,在工作线程中利用 epoll 系列函数注册此 socket,进行任务的处理。此模式

最大的优势在于一个工作线程可以独立地管理多个连接。

半同步/半异步模式的一个变种是半同步/半反应堆模式，它在主线程中管理监听/连接 socket，当任务到来时，将任务插入请求队列中，由于多个线程共享此队列，所以需要进行加锁，并运用信号量来通知其它线程来处理此任务。此模式由于需要加锁会浪费 CPU，并且每个工作线程同时只能处理一个任务。

领导者/追随者模式是多个工作线程轮流获得事件源集合，轮流监听、分发和处理请求任务的一种模式。在任意时刻，仅有一个领导者，领导者监听事件的发生，事件发生时，从追随者中推选出新的领导者，并处理任务，由此实现了并发。

2.4　提高服务器性能的其他建议

由于硬件技术的快速发展，现在服务器一般都不缺乏硬件资源，所以可以运用空间换取时间的思想。池的运用就是运用这一思想，池包括：线程池，进程池，内存池，连接池。

进程池[8~9]、线程池则是在服务器启动时创建，若要有任务请求时可以直接拿来用，这样就避免了因为创建进程或线程所耗费的大量资源。进程池模型如图 4 所示。

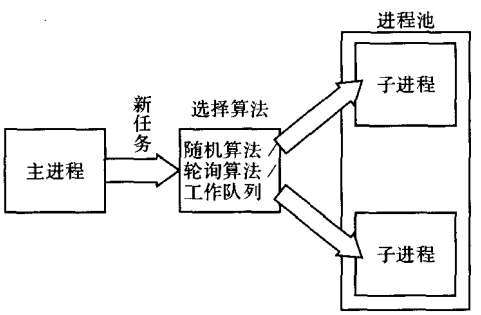

图 4　进程池模型

线程池模型与进程池类似，进程池创建子进程的数目在 3～10 个，线程池中的线程数量应该和 CPU 数量差不多。

数据复制[10]：在构建高性能的服务器时尽量避免不必要的复制操作，可以通过一些高级的 IO 函数：例如 sendfile，writev，splice，tee 等，也可以通过共享内存来避免两个进程用户空间的大量拷贝，从而提高服务器的性能。

并发程序需要考虑加锁的问题，锁不但不处理任何业务请求，而且还降低了资源的利用率，因此应尽量避免加锁，如果必须要加锁，也要尽量减小锁的粒度，例如读写锁。

使用高性能的定时器，可以高效地处理众多的定时事件。高性能定时器包括时间轮和时间堆，时间轮采用了哈希散列的思想；时间堆采用了堆排序，并且是小顶堆。

统一事件源是将 I/O 事件，定时事件，信号统一起来，通过在信号处理函数中将发生的信号通过管道来传给主线程或主进程，从而达到了统一事件源的目的，提高了服务器的稳定性。

3　构建一个高性能 Web 服务器

利用半同步/半反应堆的并发模式及线程池实现一个高性能 Web 服务器。总体的设计思路是在服务器启动时即先启动设计好的线程池，在主线程中管理监听 socket 以及连接 socket，当有数据到来时，则接受数据并初始化任务类，将任务类插入到工作队列中，工作队列通过信号量来通知空闲的工作线程，然后在工作线程中调用任务类的处理函数对 http 协议的数据进行解析，通过解析结果进行相应的处理，并将处理结果写入用户缓冲，然后修改内核事件表，注册写就绪事件，最后将处理好的数据发送给客户端。具体设计过程如下。

3.1　封装线程同步机制

在设计线程同步机制包装类时，要将实现线程同步的信号量以及互斥锁有关的线程同步函数封装进去。通过信号量来通知工作线程在任务队列中有任务需要处理；由于工作线程共享同一个工作队列所以需要使用互斥锁，对共享队列进行独占式的访问。在具体的封装时，由于在构造函数中没有返回值，所以可以通过在构造函数中抛出异常，在主线程中来进行对异常情况的捕获与处理。

3.2　封装线程池类

在线程池类中封装了线程池的构造函数、析构函数以及往请求队列中添加任务的函数，还有工作线程运行的函数，它不断地从工作队列中取出任务并执行之。在设计其构造函数时，注意 pthread_create 函数的回调函数必须是静态的，并且将类对象作为参数传递给回调函数，这样才能调用线程池中的动态方法。在任务添加函数中要先进行加锁，然后添加任务并解锁，此时才能发出信号给其他空闲工作线程来处理任务。

3.3　封装任务类

在封装的任务类中，最主要完成的任务就是通过一系列的函数完成对 http 数据的解析，以及根据解析结果对 http 方法进行处理。在解析中从状态机的状态转移图如图 5 所示。

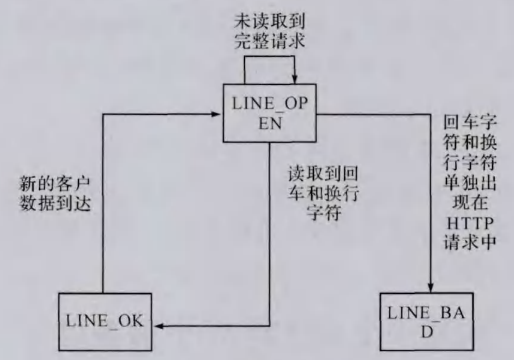

图 5 从状态机状态转移图

在主状态机中记录当前的状态,根据当前的状态来对从状态机解析出来的行进行解析,并在主状态机中完成状态的转移。

4 压力测试

运用施压程度最高的 I/O 复用方式对设计的服务器进行施压测试[11]。如果运用多线程、多进程进行压力测试时,线程和进程本身的调用会消耗一定的资源。

压力测试程序思想是通过建立指定数量的连接,并运用 epoll 模型在循环中不断的向服务器发送数据或从服务器接受数据。仿真测试如图 6~图 9 所示。

```
[yjs@yanjingshan 15]$ ./webserver 127.0.0.1 12345
create the 0th thread
create the 1th thread
create the 2th thread
create the 3th thread
create the 4th thread
create the 5th thread
create the 6th thread
create the 7th thread
```

图 6 启动服务器程序

```
[yjs@yanjingshan 16]$ ./stress_client 127.0.0.1 12345 1000
create 1 sock
build connection 0
create 1 sock
build connection 1
create 1 sock
build connection 2
create 1 sock
build connection 3
create 1 sock
```

图 7 启动压力测试程序

```
write out 80 bytes to socket 70
write out 80 bytes to socket 61
write out 80 bytes to socket 62
write out 80 bytes to socket 5
write out 80 bytes to socket 27
write out 80 bytes to socket 38
```

图 8 压力测试程序

```
got 1 http line: Connection: keep-alive
got 1 http line:
got 1 http line: GET http://localhost/index.html HTTP/1.1
got 1 http line: GET http://localhost/index.html HTTP/1.1
got 1 http line: Connection: keep-alive
```

图 9 服务器程序

仿真结果分析:压力测试程序:压力测试程序先建立了 1000 个客户连接,如图 7,然后这 1000 个客户同时向服务器端发出请求,请求内容如下:"GET http://localhost/index.html HTTP/1.1\r\nConnection: keep-alive\r\n\r\nxxxxxxxxxxxx",如图 7 所示向服务器端发送了 80 个字节的请求数据。服务器程序:服务器程序启动时如图 6 所示先建立了七个线程池以及接受存储任务类的内存池,然后等待客户请求的到来,当客户请求到来时,服务器端程序对请求进行了分析如图 9:"got 1 http line: Get http://localhost/index.html HTTP/1.1 got 1 http line: Connection keep-alive got 1 http line: ",分析之后并做了相应的应答:"HTTP/1.1 404 Not Found\r\nContent-Length: 49\r\nConnection: keep-alive\r\n\r\nThe requested file was not found on this server.\n",服务器端将应答结果返回客户端,客户端此时接收到的数据如下:"read in 119 bytes from socket 54 with content: HTTP/1.1 404 Not Found\r\nContent-Length: 49\r\nConnection: keep-alive\r\n\r\nThe requested file was not found on this server.\n ",通过对仿真结果的分析可知,在客户端高并发的访问下,服务器能够稳定并及时的处理客户请求,所以设计的 Web 服务器能够处理一定的高并发任务请求。

参 考 文 献

[1] 朱志祥,许辉辉,王雄. 基于云计算的弹性负载均衡方案[J]. 西安邮电大学学报,2013,18(6):43-47.
ZHU Zhixiang, XU Huihui, WANG Xiong. Elastic load balancing scheme based on Cloud Computing[J]. Journal of Xi'an University of Posts and Telecommunications,2013,18(6):43-47.

[2] 黄欣. 自适应网络 I/O 模型的研究与实现[D]. 广州:华南理工大学,2012:7-10.
HUANG Xin. Research and implementation of adaptive network I/O model[D]. Guangzhou: South China University of Technology,2012:7-10.

[3] 彭妮. 阶段化异步事件驱动高性能 Web 服务器[D]. 长沙:长沙理工大学,2006:12-17.
PENG Ni. Stage asynchronous event driven high performance Web server[D]. Changsha: Changsha University of Science and Technology,2006:12-17.

[4] 张超,潘旭东. Linux 下基于 EPOLL 机制的海量网络信息处理模型[J]. 强激光与粒子束,2013,25(S1):46-50.
ZHANG Chao, PAN Xudong. A mass network information processing model based on EPOLL mechanism under Linux[J]. Intense Laser and Particle Beam, 2013,25(S1):46-50.

[5] 吴敏,熊文龙. 基于 Linux 的高性能服务器端的设计与研究[J]. 交通与计算机,2007,25(1):129-131.
WU Min, XIONG Wenlong. Design and research of high performance server based on Linux[J]. Transportation and Computer,2007,25(1):129-131.

[6] Lee, Yang-Sun. High performance web server architecture with Kernel-level caching[J]. Cluster Computing, 2013,16(3):339-346.

[7] 游双. 高性能服务器编程[M]. 北京:北京机械出版社,2013:145-152.
YOU Shuang. High performance server programming [M]. Beijing: Beijing Machinery Press,2013:145-152.

[8] Xu Zongyu, Wang Xingxuan. A modified round_robin load balancing algorithm for cluster_based web servers[C]// Nanjing: IEEE Computer Society,2014:3580-3584.

[9] 谢晓燕,张静雯. 一种基于 Linux 集群技术的负载均衡算法[J]. 西安邮电大学学报,2014,19(3):64-68.
XIE Xiaoyan, ZHANG Jingwen. A load balancing algorithm based on Linux cluster technology[J]. Journal of Xi'an University of Posts and Telecommunications, 2014,19(3):64-68.

[10] Ke, Ya Ming, Wang, Yong Bin, Li, Ying. A high performance web server based on asynchronous and zero-copy [J]. Applied Mechanics and Materials, 2014,543(2):3118-3121.

[11] 赵丽娟. Web 应用程序渗透测试方法研究[D]. 长沙:中南大学,2014:4-5.
ZHAO Lijuan. Research on Web application penetration testing method[D]. Changsha: Hunan: Central South University,2014:4-5.

(上接第 624 页)

参 考 文 献

[1] 徐佳,徐琳. 航电系统的基于融合预测的健康管理[J],系统工程与电子技术,2011,22(3):428-436.
XU Jia, XU Lin. Health management based on fusion prognostics for avionics systems[J]. Systems Engineering and Electronics,2011,22(3):428-436.

[2] 杨洲,景博. 自动驾驶仪 PHM 系统健康评估方法研究[J]. 仪器仪表学报,2012,33(8):1765-1771.
YANG Zhou, JING Bo. Research of health assessment method for Autopilot PHM system[J]. Journal of Instrument,2012,33(8):1765-1771.

[3] 尹明,叶晓慧,陈少昌,故障预测与健康管理中传感器选择与定位方案[J]. 传感器和变换器,2013,158(11):230-235.
YIN Ming, YE Xiaohui, CHEN Shaochang. Sensor Selection and Location Scheme for Prognostic and Health Management [J]. Sensors and Transducers, 2013,158(11):230-235.

[4] 王红霞,潘红兵,叶晓慧. 多故障的测试序列问题研究[J]. 兵工学报,2011,32(11):1518-1523.
WANG Hongxia, PAN Hongbing, YE Xiaohui. Research on Test Sequence Problem for Multiple Fault Diagnosis[J]. Acta Armamentarii, 2011, 32(11): 1518-1523.

[5] 叶清,吴晓平,宋业新. 基于 D-S 证据理论和 AHP 的故障诊断方法研究[J]. 海军工程大学学报,2006,18(4):12-16.
YE Qing, WU Xiaoping, SONG Yexin. Fault diagnosis method based on D-S theory of evidence and AHP [J]. Journal of Naval University of Engineering,2006, 18(4):12-16.

[6] 李伟,梁玉英,朱赛. 基于神经网络和证据理论的信息融合在故障诊断中的应用[J]. 计算机测量与控制,2012,20(11):2888-2890.
LI Wei, LIANG Yuying, ZHU Sai. Fault diagnosis based on Neural Network and Evidence Theory Information Fusion[J]. Computer Measurement & Control, 2012,20(11):2888-2890.

[7] 张继军,马登武,张金春. 基于 HMM 的电子设备状态监测与健康评估[J]. 系统工程与电子技术,2013,35(8):1692-1694.
ZHANG Jijun, MA Dengwu, ZHANG Jinchun. State monitoring and health evaluation of electronic equipment using HMM[J]. Systems Engineering and Electronics,2013,35(8):1692-1694.

[8] 黄志彦,张柏书,于开山,等. D-S 证据理论数据融合算法在某系统故障诊断中的应用[J]. 电光与控制,2007,14(2),146-149.
HUANG Zhiyan, ZHANG Baishu, YU Kaishan, et al. Application of D-S evidence theory data fusion algorithm in fault diagnosis of a certain system[J]. Electronics Optics & Control,2007,14(2):146-149.

[9] 蒋黎明,何加浪,张宏. D-S 证据理论中一种新的冲突证据融合方法[J]. 计算机科学,2011,38(4):236-238.
JIANG Liming, HE Jialang, ZHANG Hong. New Fusion Approach for Conflicting Evidence in D-S Theory of Evidence[J]. Computer Science, 2011, 38(4): 236-238.

[10] 刘海燕,赵宗贵,刘熹. D-S 证据理论中冲突证据的合成方法[J]. 电子科技大学学报,2008,37(5):701-704.
LIU Haiyan, ZHAO Zonggui, LIU Xi. Combination of Conflict Evidences in D-S Theory[J]. Journal of University of Electronic Science and Technology of Chian,2008,37(5):701-704.