第33卷 第17期 计 算 机 工 程 2007年9月
Vol.33 No.17 Computer Engineering September 2007
·网络与通信· 文章编号：1000—3428(2007)17—0120—03 文献标识码：A 中图分类号：TP393

基于排队论的嵌入式Web系统性能优化

习 博，方彦军
(武汉大学自动化系，武汉 430072)

摘 要：以嵌入式 Web 技术的广泛应用为背景，探讨了一种在嵌入式 Web 系统中，利用单服务窗等待式 M/M/1 排队理论，建立嵌入式 Web 系统性能评估模型的方法。就如何利用该模型对 ARM＋uCLinux 实现的嵌入式 Web 系统进行网络性能评估，对嵌入式 Web 服务器的改进设计等问题进行了描述。

关键词：嵌入式 Web；ARM7；uCLinux；排队论

Performance Improvement of Embedded Web System Based on Queuing Theory

XI Bo, FANG Yan-jun
(Department of Automation, Wuhan University, Wuhan 430072)

【Abstract】 Basing on some applications of embedded Web technology, this paper discusses one method that uses M/M/1 queuing theory of one single service window's waiting strategy to construct the evaluating model of embedded Web system performance. One embedded Web system which realized by ARM and uCLinux is realized in practice. It describes how to use the model to evaluate the network performance and improve its capability.

【Key words】 embedded Web; ARM7; uCLinux; queuing theory

随着自动化仪器仪表向数字化、智能化和网络化方向发展，传统的 8/16 位单片机的资源已不能满足需求。32 位高性能专用CPU和嵌入式操作系统的出现，使得仪器仪表能够实现复杂的控制，进行多任务的处理。同时，嵌入式Web技术的广泛应用，使得越来越多的设备能够接入Internet，通过Web页面进行远程访问和控制[1]。近几年来，国内外的许多学者对如何提高嵌入式Web设备性能问题进行了大量的研究。文献[2]提出了对TCP/IP协议簇进行约减，以提高嵌入式Web设备的接入性能；文献[3]提出通过随机概率分配方式替代固定转发分配方式，使访问负载的分布更均匀；文献[4]提出了一种由Web应用驱动的DynamicWeb页面缓存方法来减轻服务器端的CPU负荷。

针对如何在有限的硬件条件下，合理安排系统资源，完成任务调度，最大限度地提高嵌入式 Web 应用服务器(embedded Web application server, EWAS)的响应速度及吞吐能力等问题，本文利用 M/M/1 排队论对数据包接收进行了建模分析，并就 ARM+uCLinux 嵌入式 Web 系统实现提出了内核修改、应用程序编程等性能优化方法，以提高系统的吞吐率和网络响应速度。

1 系统分析

1.1 开发平台

在具体实现中，开发板处理器采用 Samsung 公司基于 ARM7TDMI 内核的 32 位高速处理器 S3C4510B，这款处理器专门针对以太网应用，它支持媒介无关接口 MII，通过配置内部的寄存器就可以实现 10M 和 100M 的自适应。另外，本文选用嵌入式 uCLinux 作为软件开发平台。uCLinux 是近年迅速发展起来的一种专门用于微控制领域的嵌入式操作系统，是嵌入式 Linux 的一个分支，已经成功移植到多种像 S3C4510B 这样不带 MMU 的微处理器平台上，在稳定性和其他方面都有上佳表现。根据分析，笔者对嵌入式 uCLinux 系统上实现的嵌入式 Web 服务器怎样接收和处理 HTTP 请求和数据包，有了深入的认识。如图 1 所示，数据包在嵌入式 Web 服务器中分别要在用户缓冲区、系统缓冲区经过处理后，才生成数据帧包。

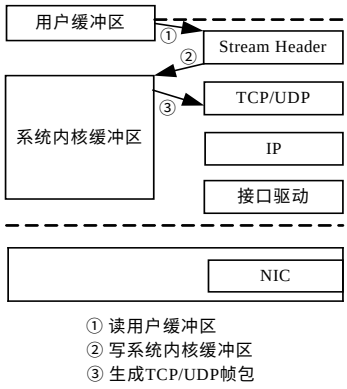

图 1 uCLinux 系统数据帧包的生成

1.2 建模分析

根据排队论理论，可以将 EWAS 服务抽象为一个 M/M/1 的排队服务系统，M/M/1 意味着到达率为 λ、服务率为 μ 的负指数服务、服务员个数为 1 的单队列服务系统。Internet 上

作者简介：习 博(1980－)，男，博士研究生，主研方向：嵌入式系统，计算机控制；方彦军，教授、博士生导师

收稿日期：2006-10-08 **E-mail：**aeroboys@sohu.com

传来的控制命令数据对 EWAS 来说就是前来排队以获得服务的顾客，而 EWAS 就是提供这个服务的窗口，服务窗口大小为 1。假设在 EWAS 上某一作业的处理时间是指数分布的，来自同一终端的在 2 个请求之间的时间间隔也被假设为按照一个指数分布，由此对这一过程建立模型如图 1 所示。

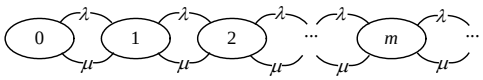

图 2　M/M/1 状态流图

其中，λ表示强度(单位时间内到达的顾客数)；μ表示服务率(在单位时间内被服务的顾客数，即服务速率)；$\rho=\lambda/\mu$(整个服务系统平均占用服务台的顾客数，即利用率)。

在正常负载下，数据包以速率为 λ 的泊松过程到达 EWAS 服务节点，服务时间服从指数分布，并且顾客按照到达的顺序接受服务，即 FCFS。则有，服务节点空闲的概率 $P_0=1-\rho$，根据状态平稳图和列哥尔莫可夫代数方程的一般规律，可计算丢包率为

$$P_x=\rho^x P_0=\rho^x\frac{1-\rho}{1-\rho^{x+1}}$$

系统长度，包括正在发送的和缓冲区中的平均数据包的个数 L 为

$$L=\sum_{x=1}^{\infty}xP_x=\sum_{x=1}^{\infty}(1-\rho)\rho^x x=\rho/(1-\rho)$$

在缓冲区中的平均数据包数 L_q 为

$$L_q=\sum_{x=1}^{\infty}xP_{x+1}=\sum_{x=2}^{\infty}(1-\rho)\rho^x(x-1)=\rho^2/(1-\rho)$$

设任一到达的数据包等待的时间为 T，当系统有 x 个数据包时，到达的数据包轮到自己发送所需的时间大于$t(t\geq 0)$的概率为 $q_x(T>t)$，设平衡时系统有 x 个数据包的概率为 P_x，那么 $T>t$ 的概率为

$$P(T>t)=\sum_{x=0}^{\infty}P_x q_x(T>t)$$

对于 q_x，当系统中有 x 个数据包时，其中一个正在发送，另外 $x-1$ 个在缓冲区内等候，即新到达的数据包就要等待 $x-1$ 个服务完成后才能发送，在 t 时间内服务的个数符合泊松分布：

$$P(T>t)=\sum_{x=1}^{\infty}P_x\sum_{i=1}^{x-1}\frac{(n\mu t)^i}{i!}e^{-n\mu t}$$

平均等待时间 W_q 则为

$$W_q=\int_0^{\infty}tdP(T\leq t)=\frac{\lambda}{\mu^2(1-\lambda/\mu)}=L_q/\lambda$$

根据上面的分析，可得出如下结论：

(1)在正常负载情况下，为了降低丢包率 P_x，应该提高节点单位时间内被服务的顾客数 μ；

(2)提高服务强度 ρ 可以降低系统处于负载时的网卡丢包率，并提高数据包处理效率，以及增大正在发送的和缓冲区中的平均数据包的个数；

(3)在超载情况下，系统负载 λ 对性能有很大的影响，λ 越大，系统的性能越低；

(4)平均等待时间W_q和缓冲区中的平均数据包数L_q的对应关系为 $L_q=\lambda W_q$，降低平均等待时间W_q，可以降低系统缓冲区中的数据包数，其性能曲线是以 λ 为斜率的一条　直线。

2　网络性能评估及优化

2.1　性能估算

已知上述开发板中ARM CPU的晶振为 50MHz，每个时钟周期为 1/(50M)，由于采用了 3 级流水线，平均每条指令需要 1.9 个时钟周期，因此执行每条指令平均时间为 1.9/(50M)= 3.8×10^{-8}s。因此确定窗口为每位顾客服务平均需 3.8×10^{-8}s。嵌入式uCLinux系统中实现EWAS的程序指令共有 3 000 条，则在ARM7 上处理需要花费的时间为 $3000\times3.8\times10^{-8}$s＝$1.14\times10^{-4}$s。根据经典数据流量模型，在 10Mb/s局域网传输数据，每次在网络上获取数据的时间间隔大约为 95.125ms，顾客到来平均间隔为 95.125ms+1.14×10^{-4}s＝95.24ms。可得：强度 λ ＝1/0.09524＝10.5 条/s；服务率 μ ＝$1/3.8\times10^{-8}$s＝2.632×10^{7}条/s，服务强度 $\rho=\lambda/\mu$＝3.99×10^{-7}。引入M/M/1 排队模型公式，计算可得服务窗空闲的概率：P_0=1−ρ＝99.9999601%；系统内顾客的均值$L=3.99\times10^{-7}$；系统内排队等候的平均顾客数：$L_q=1.592\times10^{-13}$；顾客在系统内平均逗留时间：$W_q=1.516\times10^{-14}$s。

由以上数据可知，开发板的性能到达了 99.999 9601%的工作效率；由系统内排队等候的平均顾客数 L_q 可见排队等待的可能性相当小，由顾客平均排队等待时间 W_q 也可知系统的工作效率很高，满足了网络通信的需要。

2.2　性能优化

嵌入式系统的特点是个性化比较强，不具有通用性，系统的性能不是越高越好，只要满足应用的要求即可。嵌入式Web 应用的任务不仅来自于 Internet，本地也会有一些任务搀杂其中。利用排队模型完成性能分析后，本文对 EWAS 内核中，涉及数据包接收部分的代码和通信代码进行了修改，以提高其响应速度：

(1)在涉及网络模块编程时，尽量采用 uCLinux 中 select()函数取代原先套接口阻塞的系统调用函数 fcntl()，这是因为一旦嵌入式系统通信被设置为不阻塞，就得频繁地查询套接口以检测有无信息到来，这会占用太多的 CPU 处理时间，降低系统性能，所以必须采用多路复用的 select()编程实现；

(2)在缓冲区设置方案上，本文参考了伯克利 Unix 所用的 mbuf 库缓冲区方案。考虑到开发板主要用于工业现场的数据通信，每个以太网帧包信息量最小在 64B，为了避免缓冲区中大量碎片的产生，设置缓冲区大小为 80B(多余部分用于传递模块信息)，并把多个缓冲区设置成一个链表容纳大于 64B 的帧；

(3)在了解 EWAS 的数据包接收过程后，对/uCLinux-samsung/user/boa/src 下 boa.c 文件进行修改。实现一个 net_schedule (struct net_device *dev)函数，在数据包排队进行处理前，查询网络设备状态，待返回值为 1，确定排队队列中数据小于阈值后，再利用软中断通知网络层接收数据包，交给处理器处理。

3　测试分析

完成开发后，本文将开发板接入网关地址为 192.168.1.1 的网段，IP地址设为 192.168.1.10。通过使用同一网段内IP地址为 192.168.1.12 的PC终端机进行结果测试与分析。其中，PC终端机使用WinXP操作系统和D-Link DFE-530TX PCI FAST Ethernet Adapter 10M/100M自适应网卡接入该网段，同时使用网络监听工具SpyNet Sniffer的CaptureNet进行监测。

在 EWAS 网页中加入 Request average speed、Request disposal time 和 Request disposal bytes 3 个参数用于评价嵌入式 Web 服务器处理 HTTP 网络通信协议的性能。其中，Request average speed 表示浏览器向 EWAS 发出 HTTP 请求后到嵌入式系统处理完毕并输出数据的平均速度；Request disposal time 表示 EWAS 处理 HTTP 请求所花费的平均时间；Request disposal bytes 表示 EWAS 回应 HTTP 请求时输出到 TCP 的数据数量。参数的显示是在/uCLinux-samsung/user/boa/src 下的 get.c 文件中加入语句：

```
sprintf( ext_info_string, "
<p>Request average speed: %d bytes/second <p>Request disposal time: %d ns
<p>Request disposal bytes: %d bytes
<p> \n</body>\n</html>",
speed_bytes * 1000000/ speed_time,
speed_time, speed_bytes );
```

其中，speed_time, speed_bytes 是 EWAS 收到 HTTP 请求到处理完请求时调用 speed_calc_begin 和 speed_calc_end 函数，其具体实现是在/uCLinux-samsung/user/boa/src 下 boa.c 的 main 函数中执行的：

```
//…
speed_calc_begin();
process_requests();   /* any blocked req's move from request_ready to request_block */
speed_calc_end();
//…
```

get.c 文件中 speed_calc_begin 和 speed_calc_end 使用了 uCLinux 的时间函数 gettimeofday 完成 speed_time, speed_bytes 的计算，其实现代码如下：

```
// 计算处理速度
struct timeval gtv;
struct timezone gtz;
int speed_calc_begin()
{
  gettimeofday( &gtv, &gtz );
  // gettimeofday 为 uCLinux 的时间函数
  return 0;
}
int speed_calc_end()
{
  struct timeval tv;
  struct timezone tz;
  unsigned total;
  gettimeofday( &tv, &tz );
  total = tv.tv_sec * 1000000 + tv.tv_usec;
  total -= gtv.tv_sec * 1000000 + gtv.tv_usec;
  speed_time += total;
  return 0;
}
int speed_calc_write( unsigned size )
{
  speed_bytes += size;
  return 0;
}
```

在实验室环境中对 EWAS 所在开发板进行连续长时间(24h 以上)的测试中，网络状况基本稳定，监控软件输出数据的平均速度为 1Kb/s，EWAS 处理数据请求所花费的平均时间为 90ms，EWAS 回应数据请求到输出 TCP 数据量的平均值为 900B，其 uCLinux 操作系统及应用程序性能稳定。

4 结束语

在嵌入式 Web 多任务系统设计过程中，由于嵌入式系统所使用的芯片上存储资源十分有限，既不能无限制地增强系统的处理能力，也不能让系统的吞吐率太低，使应用程序无法正常运转。作为嵌入式 Web 应用的载体及提供应用服务的核心，嵌入式 Web 应用服务器 EWAS 性能优劣直接决定整个应用的效率。因此，本文利用排队论对嵌入式 Web 系统的数据包接收过程建立了数学分析模型，并就如何优化 ARM＋uCLinux 实现的嵌入式 Web 系统进行了研究，提出了优化方法，通过实验测试表明，该方法可以较大地提高嵌入式 Web 服务器的性能以及系统通信的效率。

参考文献

1 习　博, 方彦军. 嵌入式监测系统中网络通信的研究与实现[J]. 电气自动化设备, 2004, 24(7): 68-72.

2 韩光洁, 赵　海, 王金东, 等. Embedded Internet 环境下 TCP/IP 协议簇的约减[J]. 小型微型计算机, 2004, 25(9): 1602-1606.

3 郭成城, 晏蒲柳. 一种异构 Web 服务器集群动态负载均衡算法[J]. 计算机学报, 2005, 28(2): 179-184.

4 Iyengar A, Challenger J. Improving Web Server Performance by Caching Dynamic Data[C]//Proceedings of USENIX Symposium on Internet Technologies and Systems, Berkeley, USA. 2001-12.

5 张文波, 赵　海, 王小英, 等. 基于 ARMLinux 的 EWS 过载性能研究[J]. 通信学报, 2005, 26(8): 87-92.

6 Sridhar T. 嵌入式通信软件设计[M]. 北京: 北京航空航天大学出版社, 2004.

7 陆传来. 排队论[M]. 北京: 北京邮电大学出版社, 2000.

（上接第 119 页）

5 Zhang Zhen, Wang Xiaoming, Wang Yunxiao. A P2P Global Trust Model Based on Recommendation[C]//Proc. of International Conference on Machine Learning and Cybernetics. 2005, 7: 3975.

6 Bearly T, Kumar V. Expanding Trust Beyond Reputation in Peer-to-Peer Systems[C]//Proc. of the 15th IEEE International Workshop on Database and Expert Systems Applications. 2004: 966-970.

7 Yu Bin, Singh M P, Sycara K. Developing Trust in Large-scale Peer-to-Peer Systems[C]//Proc. of the 1st IEEE Symposium on Multi-agent Security and Survivability. 2004: 1-10.

8 Dou Wen. A Recommendation-based Peer-to-Peer Trust Model[J]. Journal of Software, 2004, 15(4): 571-583.