多线程软件执行效率与改进方法研究

杨 坤
(中国人民解放军 91404 部队 河北 秦皇岛 066000))

[摘 要] 将软件进行多线程改进,可以解决软件并行性问题,能够显著提升软件的运行效率。但如果软件改进的方法不当很容易造成系统不稳定。该文简要介绍了线程与进程的特点与差异,对在 Linux 操作系统环境下软件多线程与多进程的执行效率进行了对比,分析了产生这种执行效率差异的原因以及多线程与多进程技术应用在软件各方面改进时的优劣,并提出了实施软件改进的策略与实现方法。

[关键词] 多进程 多线程 软件优化 执行效率
中图分类号:TP311.5 文献标志码:A 文章编号:1008-1739(2011)11-38-3

Research on Multithreading Software Execution Efficiency and Improvement Approach

YANG Kun
(The Unit 91404 of PLA, Qinhuangdao Hebei 066000, China)

Abstract: The improvement on multithreading software can resolve the question of software parallelism and significantly improve software efficiency. But improper software modification may lead to system instability. This paper makes a contrast for execution efficiency between software multithreading multiprocess under Linux operation system context, analyzes the causes of execution efficiency difference and advantage and disadvantages in software of multithreading and multiprocess application, and suggests the strategy and practical approach for software improvement.

Key words multiprocess; multithreading; software optimization; execution efficiency

1 引言

线程(thread),有时被称为轻量级进程(Lightweight Process, LWP),是程序执行流的最小单元。一个标准的线程由线程 ID,当前指令指针(PC),寄存器集合和堆栈组成。进程(process)是一个具有独立功能的程序关于某个数据集合的一次运行活动。它可以申请和拥有系统资源,是一个动态的概念,是一个活动的实体。它不只是程序的代码,还包括当前的活动,通过程序计数器的值和处理寄存器的内容来表示。简单的说:进程是资源分配的最小单位,线程是 CPU 调度的最小单位。线程和进程有着迥异的区别,也有着千丝万缕的联系,一个显著的区别就是进程是能够独立运行的,但线程却不可以,进程可以创建、拥有子进程,子进程和父进程同时拥有着不同的代码空间和数据空间,也正是由于这个原因导致了进程间的通信比较复杂,会涉及到管道、共享内存、消息队列等等,线程可以共享数据空间,采用多线程技术可以使一个应用程序的多个逻辑单元并发执行,但并不是多个用户进程在执行,操作系统也不把每个线程作为独立的进程来分配独立的系统资源。这也就决定了线程间的通信可以简便、快捷,同时对于系统资源的开销也就大幅减小。线程仅是过程调用,每个线程有自己的执行堆栈和程序计数器为其执行上下文。

定稿日期:2011-04-26

2 多线程与多进程的执行效率对比

对于程序员来说,CPU、内存与程序运行时间是 3 个重要资源,下面使用一个程序对实际应用中的线程和进程的执行效率进行分析和比较。可以使用函数 pthread_create 动态创建 255 个线程,在每个线程中重复进行打印"hello world"语句,模拟多线程并发执行,如果是在多处理器机器上,系统就可为每个线程分配一个 CPU,多个线程可以并发执行。测试代码如下:

```
FILE *logFile=NULL;
print_hello_world ()
```

```
{ int i=0; for(i=0;i<a;i++)
{ printf("hello world\n");
fprintf(logFile,"hello world\n");}
pthread_exit(0); }
main()
{ int i=0;
pthread_t pid[255];
logFile=fopen(Test_Log,"a+");
for(i=0;i<255;i++)
pthread_create (&pid[i],NULL,print_hello_world,NULL);
for(i=0;i<255;i++)
pthread_join(pid[i],NULL);
return 0;}
```

然后再编制一段程序在 main 函数中调用 fork 函数创建 255 个进程,在每个进程中去完成与多线程相同的任务,重复进行打印"hello world"语句。测试代码如下:

```
FILE *logFile=NULL;
main()
{ int i=0,j=0;
logFile=fopen(TEST_LOGFILE,"a+");
for(i=0;i<255;i++)
{ if(fork()==0)
{ for(j=0;j<10;j++)
{ printf("hello world\n");
fprintf(logFile,"hello world\n");}
exit(0);}}
wait(0);
return 0;}
```

测试环境:CPU 主频 2.4 G,内存 2 G,软件:Suse Linunx Enterprise 10,内核版本:linux-2.6.16。经过测试后得到了执行效率对比图如图 1 所示。

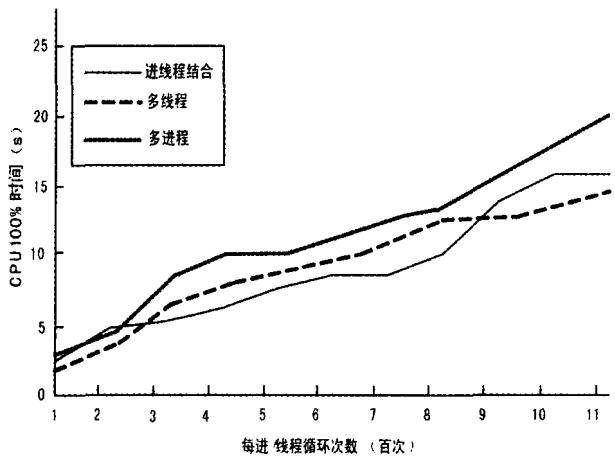

图 1 多线程与多进程软件执行效率对比图

图 1 中的短虚线和长虚线分别表示采用多进程和多线程完成任务时,CPU 的工作时间。从上图中不难看出,当采用多线程和多进程分别执行相同的数据处理任务,其效率与完成的任务量有着直接的关系,当完成的处理任务量较大时,多线程的优势并不明显,而完成的任务量较小时,多线程要优于多进程。

多线程具备创建快捷、资源开销很小等优点,它可以在使用过程中节省大量的 CPU 时间,从而使得在创建过程中十倍乃至百倍优于进程。但在进行的任务较为繁重时,建立多个线程不一定就可以获得更高得执行效率,反而会使 CPU 的任务量大幅增加,所以当任务量较大时可以考虑进线程相结合的思路,而对于对系统资源要求不高的任务则可以倾向于多线程设计。针对不同的业务、使用难易和可靠性在不同维度上对多进程与多线程进行了对比,其结果如表 1 所示。

表 1 进程与线程对比表

对比项	多进程	多线程	结论
数据共享、同步	数据共享复杂,需要用 IPC;数据是分开的,同步简单。	数据共享简单,数据同步复杂	各有优势
系统资源	占用系统资源多,利用率低	占用系统资源少、利用率高	线程占优
创建、销毁与切换	程序复杂、速度慢	程序简单、速度快	线程占优
编程与调试	简单	复杂	进程占优
可靠性	多进程间不会互扰	如出现异常,会引起进程异常	进程占优

根据表 1 中列出的情况,针对在多线程与多进程的选择上可以得出如下的结论:如需要进行频繁地创建与销毁可以优先使用多线程,需要进行大量的数据计算优先使用多线程,进行消息收发与消息处理优先使用多进程,进行消息解码和业务处理优先使用多线程。

3 IPTV 组播应用技术问题

(1) 进程控制优化

具体优化思路包括进程启动改为启动线程;终止改为从函数返回;进程标识创建线程时将线程 ID 保存在一个全局数组中;用 pthread_join()函数和 pthread_detach()函数实现父子线程同步;将所有线程的 sleep()改为 Pthread_de-lay_np()。

(2) 信号处理机制的优化

要实现多进程结构向多线程结构的转变,首先要对进线程的信号类型建立相应的处理函数,将进程的信号分成多份,做好相应标记后,分发给相应的线程进行处理,在处理过程中根据相应的标记调用相关的处理函数,最后将处理结果提交进程。

(3) 全局变量的使用

如果多线程中使用全局和静态变量如果处理不好，就会出现互斥问题。而且全局变量是在多个线程间共享使用的,而每个线程都可以对全局变量进行操作,这就很容易造成冲突。可以采用如下方法解决此问题:

① 可以申请环境变量。在线程里用这个命令设置环境变量的值,实现线程之间的参数传递;

② 定义私有指针变量,并使其指向结构的指针变量;

③ 在线程启动过程中,对结构的内存空间进行指定,同时在指针变量中保存结构的地址。当线程退出时,完成对空间和指针变量的释放;

④ 将程序中涉及到的全局变量进行结构化组合,这样一来作为结构的成员名称就是全局变量名。

(4) 动态内存异常

人们无法通过编译过程发现动态内存分配过程中发生的异常,这些异常只能在程序的执行过程中才能够被捕获发现,从而给程序员对程序的编制调试与测试带来了很多困难。要解决问题,可考虑使用动态全局内存回收链表,在线程建立时对各个线程的内存指针进行保存。并在线程退出时,由线程释放该动态内存。

4 结束语

目前几乎所有的操作系统都支持多线程和多进程,在Window操作系统和linux操作系统中更是广泛地应用了多线程、多进程技术。主要对相同条件下多线程和多进程技术的执行效率进行了实际比较，并提出了软件优化与改进的实施方法。同时发现软件的多线程与多进程的使用会根据应用的领域不同存在着很大的执行效率差异，软件的具体改进方法还需要在实际环境下进行进一步测试与研究。

参考文献

[1] 郭玉东《linux操作系统结构分析》[M].西安电子科技大学出版社,2002.

[2] Mark GS.obell[美]《A Practical Guide to Red Hat Linux 8》[M]电子工业出版社,2004.

[3] 朱响斌,徐时亮《Linux的实时性能测试》[J]微电子学与计算机,2004,(21)56-58.

Asigra推出首款企业级云备份方案

始创于1986年的云备份和恢复(BURR)软件的领先供应商Asigra Inc.日前宣布推出Asigra Cloud Backup v11新品,该产品具备云恢复功能,且具有以下优势:业界首批能够对手提设备(如平板电话和智能手机)进行数据保护的方案;首个具有多用户端能够显著削减管理资源的产品；并且是业界自动化许可程度最高的服务器。该新型软件也是第一款企业级云备份平台,可保护整个数字领域（整个企业局域网内的存储、服务器、台机、笔记本以及远程平板电脑、智能手机和笔记本),并且为客户提供数据恢复和恢复保证(R2A)以及建立新的性能基准。

Asigra云备份V11的新特征

业界首款中小型企业和企业级云备份和恢复方案,可为平板电脑和智能手机提供数字领域端到端的保护。

2011版本的新特性为DS-Consumer?,即具备恢复和恢复保证(R2A)的消费者云备份。目前市场上有数不清的消费者备份解决方案,这些方案能够进行大量的数据备份工作,但是这些方案却不能保证,所需要恢复的备份数据一定能够实现恢复。Asigra云备份为用户提供了本地和远程数据的副本,并且允许终端用户及时恢复到任意点。Asigra助力的云备份服务供应商现在为消费者提供了一个颠覆传统观念的解决方案。

版本11为多用户管理环境的简化和综合管理提供了更高的自动化。优势包括极大的减少了部署和维护时间需求,更高的运营效率,并且有充足的许可以实现自动的"许可重填",从而确保服务级别协议,使客户永远不受云许可管理的影响。

Asigra Cloud Backup v11同时引进了综合和直观的Web命令中心,通过DS-NOC(网络运营中心)对整体Asigra环境提供任意时间、任意地点的可见性、控制和管理。

性能提升400%,可满足不断增长的数据需求,并完成分配的备份窗口内的备份。Asigra Cloud Backup v11支持存储硬件、网络交换机和服务器上的10 Gbps局域网接口。此外，内部Asigra进程已得到改善以处理更快的数据读\写。这些内部的改善包括多线程数据重复复制删除、读I/O和数据吞吐的平行处理等。

25年来,Asigra全心全意的关注于其服务供应商团体(SPs),后者为其终端用户提供由Asigra助力的服务,或销售私有云或混合云部署的软件。Asigra致力于帮助其合作伙伴成为更加灵活和盈利的企业。此外,虽然大多数SaaS应用仍然保持同硬件的紧密耦合,迫使管理服务供应商(MSPs)被动地来适应市场环境,但是Asigra的云备份完全不依赖于硬件,部署的灵活性更强,还降低了总的投入成本。