上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

L3-1. 线程

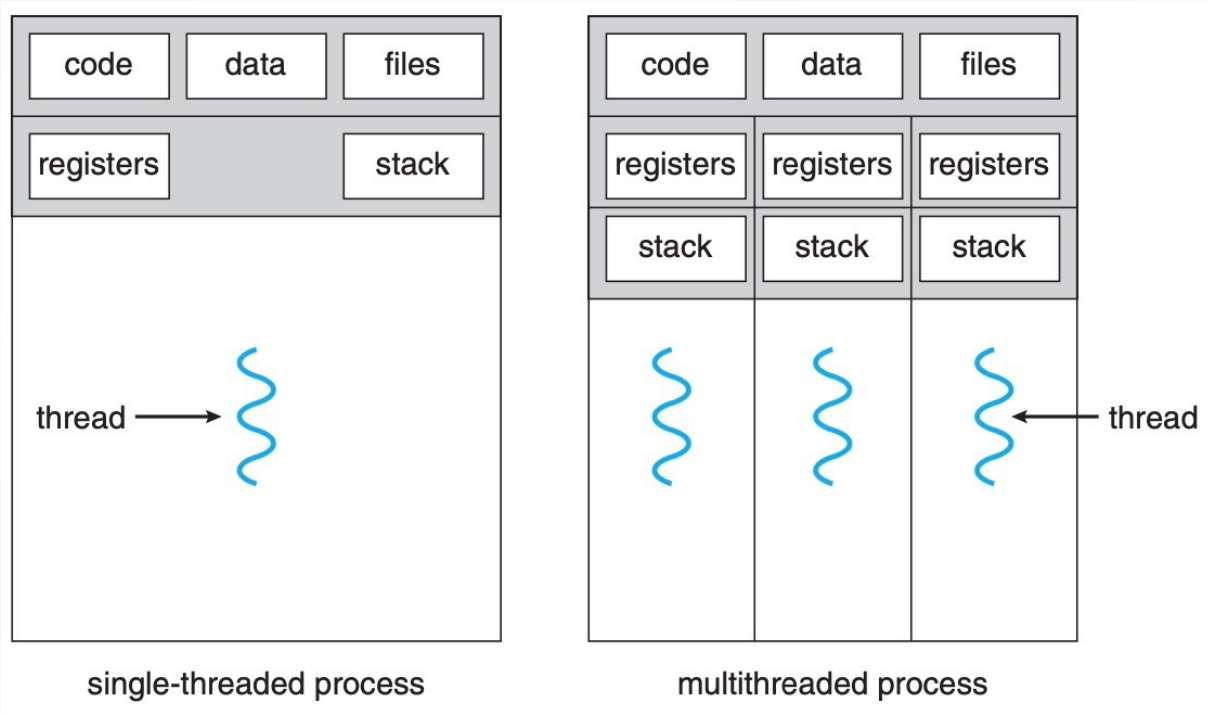
宋卓然

上海交通大学计算机系

songzhuoran@sjtu.edu.cn

什么是线程

- 每个线程是CPU使用的一个基本单元；它包括线程ID、程序计数器、 寄存器组和堆栈。
- 它与同一进程的其他线程共享代码段、数据段和其他操作系统资源



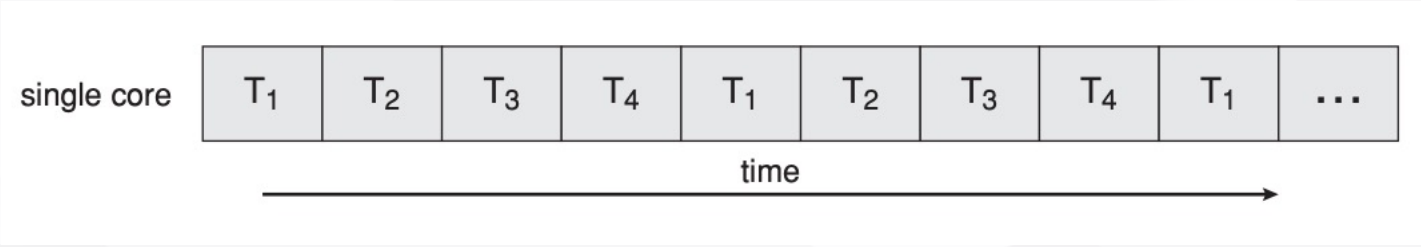
为什么要有线程

- 现代计算机运行的大多数应用软件都是多线程的
- 一个应用程序通常作为具有多个控制线程的一个进程来实现
  - 显示图像、文本
  - 接收数据
  - 拼写检查
- 进程创建往往是重量级的，而线程创建是轻量级的
- 操作系统的内核往往都是多线程的

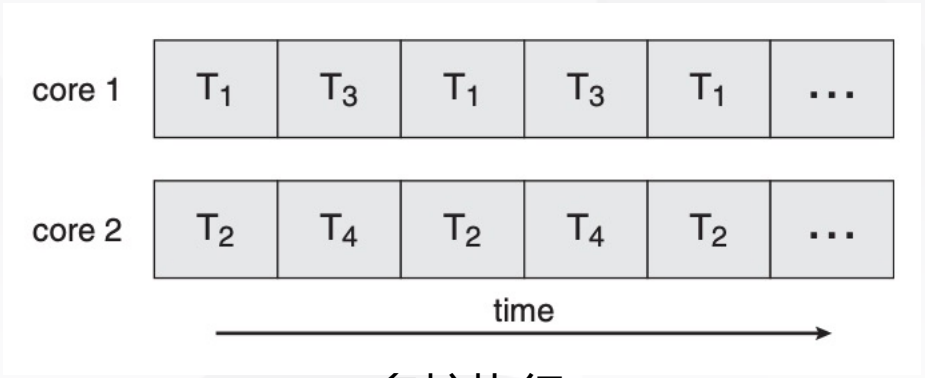
多线程的优点

- 响应性:如果一个交互程序采用多线程，那么即使部分阻塞或者执行冗长操作，它仍可以继续执行，从而增加对用户的响应程度
- 资源共享:进程只能通过如共享内存和消息传递之类的技术共享资源；而线程默认共享它们所属进程的内存和资源
- 经济:进程创建所需的内存和资源分配非常昂贵。由于线程能够共享它们所属进程的资源，所以创建和切换线程更加经济
- 可伸缩性:对于多处理器体系结构，多线程的优点更大，因为线程可在多处理核上并行运行。不管有多少可用CPU，单线程进程只能运行在一个CPU上

多核系统



单核执行



多核执行

多核速度更快

注意**并行性与并发性**的区别：并行系统可以同时执行多个任务。相比之下，并发系统支持多个任务，允许所有任务都能取得进展。因此，没有并行，并发也是可能的。在SMP 和多核架构出现之前，大多数计算机系统只有单个处理器。CPU调度器通过快速切换系统内的进程，以便允许每个进程取得进展，从而提供并行假象。这些进程并发运行，而非并行运行

Amdahl定律

- Amdahl定律是一个公式。对于既有串行也有并行组件的应用程序，该公式确定由于额外计算核的增加而潜在的性能改进。如果S是应用程序的一部分，它在具有N个处理核的系统上可以串行执行，那么该公式如下：

$$\text{加速比} \leqslant \frac{1}{S + \frac{1-S}{N}}$$

- 一个例子，假设我们有一个应用程序，其75%为并行而25%为串行。如果我们在具有两个处理核的系统上运行这个程序，我们能得到1.6 倍的加速比。如果我们再增加两核 (一共有4个)， 加速比是2.28倍。
  - S=0.25 N=2

Amdahl定律

- Amdahl定律是一个公式。对于既有串行也有并行组件的应用程序，该公式确定由于额外计算核的增加而潜在的性能改进。如果S是应用程序的一部分，它在具有N个处理核的系统上可以串行执行，那么该公式如下：

$$\text{加速比} \leqslant \frac{1}{S + \frac{1-S}{N}}$$

- Amdahl定律的一个有趣事实是，当N趋于无穷大时，加速比收敛到1/S。例如，如果应用程序的40%为串行执行，无论我们添加多少处理核，那么最大加速比为2.5倍。这是Amdahl定律背后的根本原则:对于通过增加额外计算资源而获得的性能，应用程序的串行部分可能具有不成比例的效果

进程 vs. 线程

进程

- 独立
- 运行时携带更多状态信息
- 拥有独立的地址空间
- 上下文切换通常较慢

线程

- 为进程的子集
- 共享进程状态、存储、资源
- 共享进程的地址空间
- 上下文切换往往较快

线程类别

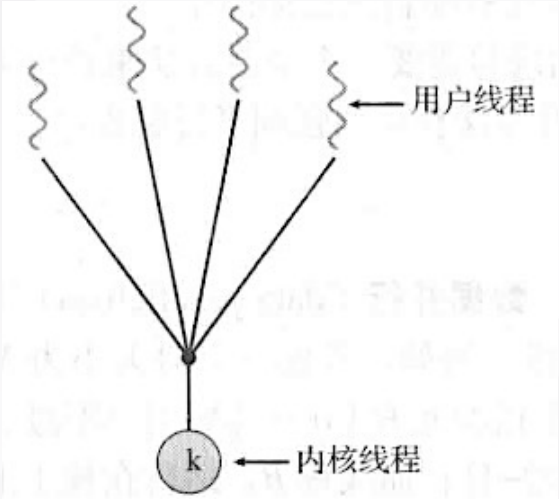
- 用户级线程
  - 由用户级线程库完成线程管理
  - 三种主要的线程库：POSIX Pthreads、Win32 threads、Java threads
- 内核级线程
  - 由操作系统内核完成线程管理
  - 例子：Windows XP/2000, Solaris, Linux, Tru64 UNIX, Mac OS X

多线程模型

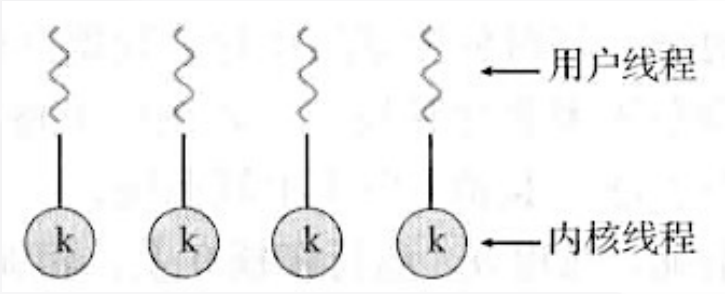
- 用户级线程与内核级线程的关系
  - 多对一模型
  - 一对一模型
  - 多对多模型
  - 双层模型

多对一模型

- 多对一模型映射多个用户级线程到一个内核线程
- 优点
  - 线程管理可由用户空间的线程库来完成的
- 缺点
  - 如果一个线程执行阻塞系统调用，那么整个进程将会阻塞
  - 因为任一时间只有一个线程可以访问内核，所以多个线程不能并行运行在多处理核系统



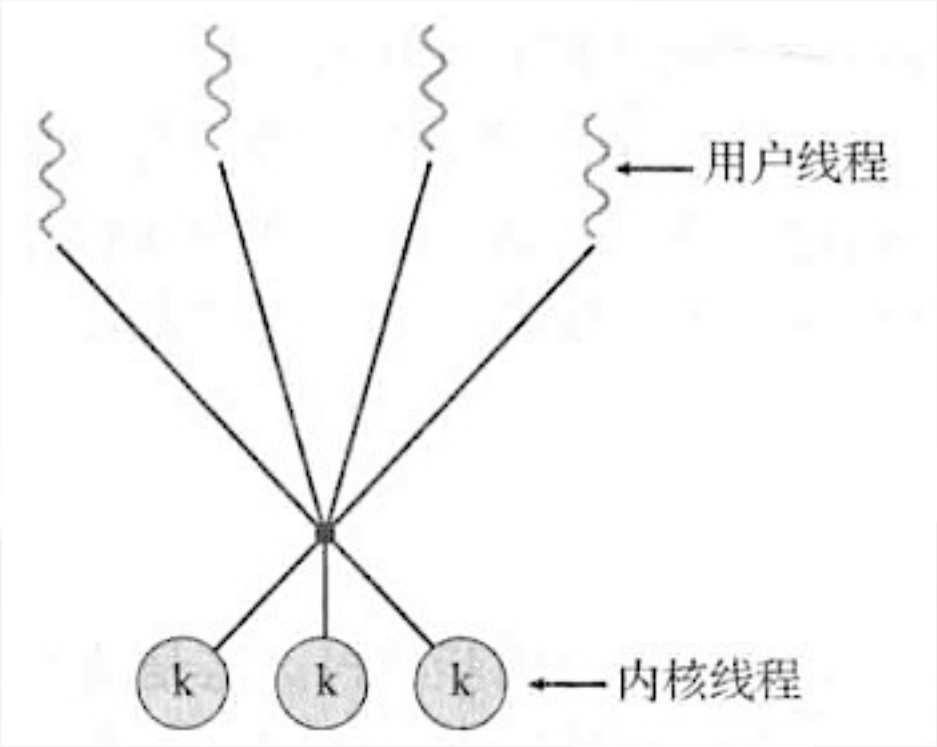
一对一模型



- 一对一模型映射每个用户线程到一个内核线程
- 优点
  - 该模型在一个线程执行阻塞系统调用时，能够允许另一个线程继续执行，所以它提供了比多对一模型更好的并发功能
- 缺点
  - 创建一个用户线程就要创建一个相应的内核线程，有一定的开销，所以这种模型的大多数实现限制了系统支持的线程数量
- 例子：Linux、Window XP/2000

多对多模型

- 多对多模型映射多个用户级线程到同样数量或更少数量的内核线程，其中，内核线程的数量可能与特定应用程序或特定机器有关
- 例子：Windows NT/2000

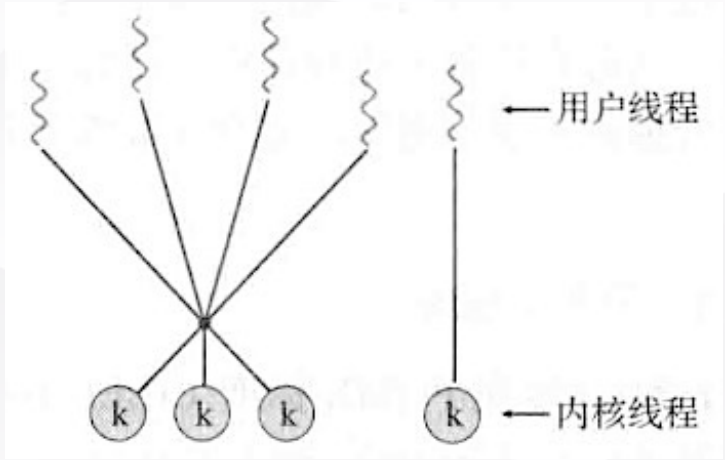


对并发性的影响

- 各模型对并发性的影响
  - 多对一模型允许开发人员创建任意多的用户线程，但是由于内核只能一次调度一个线程，所以并未增加并发性
  - 虽然一对一模型提供了更大的并发性，但是开发人员应小心，不要在应用程序内创建太多线程
  - 多对多模型没有这两个缺点:开发人员可以创建任意多的用户线程，并且相应内核线程能在多处理器系统上并发执行

双层模型

- 与多对多模型相似，同时可以允许一个用户线程与一个内核线程绑定
- 例子
  - IRIX
  - HP-UX
  - Tru64 UNIX
  - Solaris 8 and earlier



线程库

- 线程库（thread library）为程序员提供创建和管理线程的API
- 两种实现线程库的方法
  - 用户级线程库
    - 针对用户级线程库的所有代码、数据结构都存在于用户空间
    - 调用库内的一个函数只是导致了用户空间内的一个本地函数的调用，而不是系统调用
  - 内核级线程库
    - 针对内核级线程库的所有代码、数据结构都存在于内核空间
    - 调用库中的一个API函数通常会导致对内核的系统调用
- 三种主要线程库：POSIX Pthreads, Win32 threads, Java threads

Pthreads

- Pthreads是POSIX标准定义的线程创建与同步API
- 可提供用户级或内核级的库
- 用于线程创建、同步
- 广泛用于UNIX类型的操作系统
  - Solaris, Linux, Mac OS X

Pthreads实例

```
#include <pthread.h>
#include <stdio.h>

int sum; /* this data is shared by the thread(s) */
void *runner(void *param); /* threads call this function */

int main(int argc, char *argv[])
{
   pthread_t tid; /* the thread identifier */
   pthread_attr_t attr; /* set of thread attributes */

   if (argc != 2) {
      fprintf(stderr,"usage: a.out <integer value>\n");
      return -1;
   }
   if (atoi(argv[1]) < 0) {
      fprintf(stderr,"%d must be >= 0\n",atoi(argv[1]));
      return -1;
   }

   /* get the default attributes */
   pthread_attr_init(&attr);
   /* create the thread */
   pthread_create(&tid,&attr,runner,argv[1]);
   /* wait for the thread to exit */
   pthread_join(tid,NULL);

   printf("sum = %d\n",sum);
}
```

本程序已有两个线程:初始(父)线程，即main();执行累加和(子)线程，即runner()

pthread_create创建ruuner线程
pthread_join等待ruuner线程完成

Pthreads实例

```
/* The thread will begin control in this function */
void *runner(void *param)
{
   int i, upper = atoi(param);
   sum = 0;

   for (i = 1; i <= upper; i++)
      sum += i;

   pthread_exit(0);
}
```

子进程负责完成非负整数的累加

Pthreads实例 多个线程

```
#define NUM_THREADS 10

/* an array of threads to be joined upon */
pthread_t workers[NUM_THREADS];

for (int i = 0; i < NUM_THREADS; i++)
   pthread_join(workers[i], NULL);
```

pthread_join等待多个线程的方法：将这个操作包含在for循环中

课堂习题

1.在多处理器系统上采用多个用户级线程的多线程解决方案，比在单处理机系统上，能够提高更好的性能吗?请解释

2.在同一进程的多线程之间， 下列哪些程序状态部分会被共享?

a.寄存器值 b.堆内存 c.全局变量 d.堆栈内存

3.设有一个应用，其60%为并行部分，而处理核数量分别为(a)2个和(b)4个。利用Amdahl定 律，计算加速增益

4.有可能有并发但无并行吗?请解释