《编程之美——微软技术面试心得》

《编程之美——微软技术面试心得》（http://www.china-pub.com/38070）是微软亚洲研究院技术创新组研发主管邹欣继《移山之道——VSTS 软件开发指南》后的最新力作。它传达给读者：微软重视什么样的能力，需要什么样的人才。但它更深层的意义在于引导读者思考，提倡一种发现问题、解决问题的思维方式，充分挖掘编程的乐趣，展示编程之美。本书 3 月份上市。网上讨论和解答在：www.msra.cn/bop

题目《让 CPU 占用率曲线听你指挥》

问题

写一个程序，让用户来决定 Windows 任务管理器（Task Manager）的 CPU 占用率。程序越精简越好，计算机语言不限。例如，可以实现下面三种情况：

1. CPU的占用率固定在50%，为一条直线；

2. CPU的占用率为一条直线，但是具体占用率由命令行参数决定（参数范围1~ 100）；

3. CPU的占用率状态是一个正弦曲线。

分析与解法[1]

有一名学生写了如下的代码：

```
while (true)
{
   if (busy)
      i++;
   else

}
```

然后她就陷入了苦苦思索：else 干什么呢？怎么才能让电脑不做事情呢？CPU 使用率为 0 的时候，到底是什么东西在用 CPU？另一名学生花了很多时间构想如何“深入内核，以控制 CPU 占用率”——可是事情真的有这么复杂么？

MSRA TTG（Microsoft Research Asia, Technology Transfer Group）的一些实习生写了各种解法，他们写的简单程序可以达到如图 1-1 所示的效果。

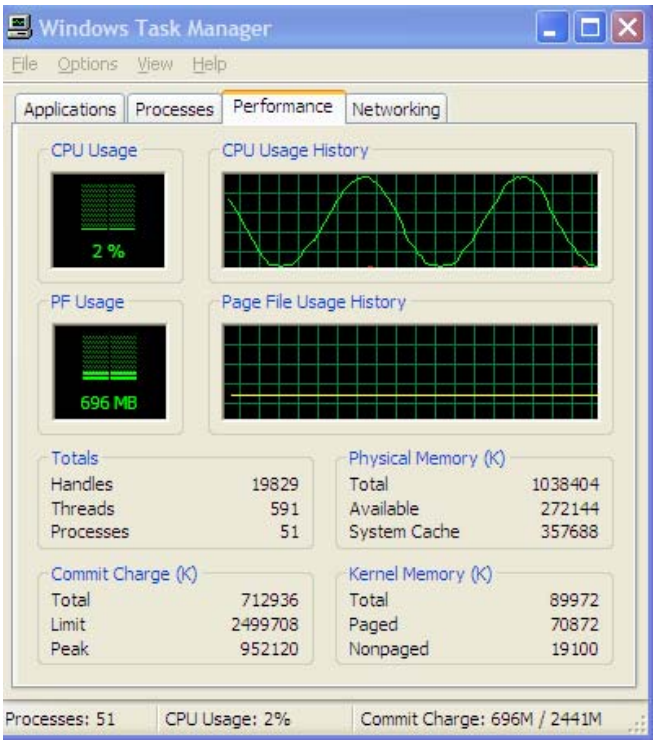

图 1-1 编码控制 CPU 占用率呈现正弦曲线形态

看来这并不是不可能完成的任务。让我们仔细地回想一下写程序时曾经碰到的问题，如

[1] 作者注：当面试的同学听到这个问题的时候，很多人都有点意外。我把我的笔记本电脑交给他们说，这是开卷考试，你可以上网查资料，干什么都可以。大部分面试者在电脑上的第一个动作就是上网搜索“CPU 控制 50%”这样的关键字，当然没有找到什么直接的结果。不过这本书出版以后，情况可能就不一样了。

果我们不小心写了一个死循环，CPU 占用率就会跳到最高，并且一直保持 100%。我们也可以打开任务管理器[2]，实际观测一下它是怎样变动的。凭肉眼观察，它大约是 1 秒钟更新一次。一般情况下，CPU 使用率会很低。但是，当用户运行一个程序，执行一些复杂操作的时候，CPU 的使用率会急剧升高。当用户晃动鼠标时，CPU 的使用率也有小幅度的变化。

那当任务管理器报告 CPU 使用率为 0 的时候，谁在使用 CPU 呢？通过任务管理器的“进程（Process）”一栏可以看到，System Idle Process 占用了 CPU 空闲的时间——这时候大家该回忆起在“操作系统原理”这门课上学到的一些知识了吧。系统中有那么多进程，它们什么时候能“闲下来”呢？答案很简单，这些程序或者在等待用户的输入，或者在等待某些事件的发生（`WaitForSingleObject()`），或者进入休眠状态（通过 `Sleep()`来实现）。

在任务管理器的一个刷新周期内，CPU 忙（执行应用程序）的时间和刷新周期总时间的比率，就是 CPU 的占用率，也就是说，任务管理器中显示的是每个刷新周期内 CPU 占用率的统计平均值。因此，我们写一个程序，让它在任务管理器的刷新期间内一会儿忙，一会儿闲，然后通过调节忙/闲的比例，就可以控制任务管理器中显示的 CPU 占用率。

【解法一】简单的解法

步骤 1 要操纵 CPU 的 usage 曲线，就需要使 CPU 在一段时间内（根据 Task Manager 的采样率）跑 busy 和 idle 两个不同的 loop，从而通过不同的时间比例，来获得调节 CPU Usage 的效果。

步骤 2 Busy loop 可以通过执行空循环来实现，idle 可以通过 `Sleep()`来实现。

问题的关键在于如何控制两个 loop 的时间，方法有二：

Sleep一段时间，然后以for循环n次，估算n的值。

那么对于一个空循环 `for(i = 0; i < n; i++)`；又该如何来估算这个最合适的 n 值呢？我们都知道 CPU 执行的是机器指令，而最接近于机器指令的语言是汇编语言，所以我们可以先把这个空循环简单地写成如下汇编代码后再进行分析：

```
loop:
mov dx i        ;将i置入dx寄存器
inc dx          ;将dx寄存器加1
mov i dx        ;将dx中的值赋回i
cmp i n         ;比较i和n
jl loop         ;i 小于 n 时则重复循环
```

假设这段代码要运行的 CPU 是 P4 2.4Ghz（2.4 * 10 的 9 次方个时钟周期每秒）。现代 CPU 每个时钟周期可以执行两条以上的代码，那么我们就取平均值两条，于是让（2 400 000 000 * 2）/5=960 000 000（循环/秒），也就是说 CPU 1 秒钟可以运行这个空循环 960 000 000 次。不过我们还是不能简单地将 n = 60 000 000，然后 `Sleep(1000)`了事。如果我们让 CPU 工

[2] 如果应聘者从来没有琢磨过任务管理器，那还是不要在简历上说“精通 Windows”为好。

作 1 秒钟 ,然后休息 1 秒钟 ,波形很有可能就是锯齿状的——先达到一个峰值(大于>50%) , 然后跌到一个很低的占用率。

我们尝试着降低两个数量级，令 $n = 9\ 600\ 000$，而睡眠时间相应改为 10 毫秒（Sleep(10)）。用 10 毫秒是因为它不大也不小，比较接近 Windows 的调度时间片。如果选得太小（比如 1 毫秒），则会造成线程频繁地被唤醒和挂起，无形中又增加了内核时间的不确定性影响。最后我们可以得到如下代码：

代码清单 1-1

```
int main()
{
   for(;;)
   {
      for(int i = 0; i < 9600000; i++);
      Sleep(10);
   }
   return 0;
}
```

在不断调整 9 600 000 的参数后，我们就可以在一台指定的机器上获得一条大致稳定的 50% CPU 占用率直线。

使用这种方法要注意两点影响：

1. 尽量减少sleep/awake的频率，如果频繁发生，影响则会很大，因为此时优先级更高的操作系统内核调度程序会占用很多CPU运算时间。

2. 尽量不要调用system call（比如I/O这些privilege instruction），因为它也会导致很多不可控的内核运行时间。

该方法的缺点也很明显：不能适应机器差异性。一旦换了一个 CPU，我们又得重新估算 n 值。有没有办法动态地了解 CPU 的运算能力，然后自动调节忙/闲的时间比呢？请看下一个解法。

【解法二】使用 GetTickCount()和 Sleep()

我们知道 GetTickCount()可以得到“系统启动到现在”的毫秒值，最多能够统计到 49.7 天。另外，利用 Sleep()函数，最多也只能精确到 1 毫秒。因此，可以在“毫秒”这个量级做操作和比较。具体如下：

利用 GetTickCount()来实现 busy loop 的循环，用 Sleep()实现 idle loop。伪代码如下：

代码清单 1-2

```
int busyTime = 10;  //10 ms
int idleTime = busyTime;  //same ratio will lead to 50% cpu usage
```

```
Int64 startTime = 0;
while (true)
{
    startTime = GetTickCount();
    // busy loop的循环
    while ((GetTickCount() - startTime) <= busyTime) ;

    //idle loop
    Sleep(idleTime);
}
```

这两种解法都是假设目前系统上只有当前程序在运行，但实际上，操作系统中有很多程序都会在不同时间执行各种各样的任务，如果此刻其他进程使用了 10% 的 CPU，那我们的程序应该只能使用 40%的 CPU（而不是机械地占用 50%），这样可达到 50%的效果。

怎么做呢？

我们得知道"当前 CPU 占用率是多少"，这就要用到另一个工具来帮忙——Perfmon.exe。

Perfmon 是从 Windows NT 开始就包含在 Windows 服务器和台式机操作系统的管理工具组中的专业监视工具之一（如图 1-2 所示）。Perfmon 可监视各类系统计数器，获取有关操作系统、应用程序和硬件的统计数字。Perfmon 的用法相当直接，只要选择您所要监视的对象（比如：处理器、RAM 或硬盘），然后选择所要监视的计数器（比如监视物理磁盘对象时的平均队列长度）即可。还可以选择所要监视的实例，比如面对一台多 CPU 服务器时，可以选择监视特定的处理器。

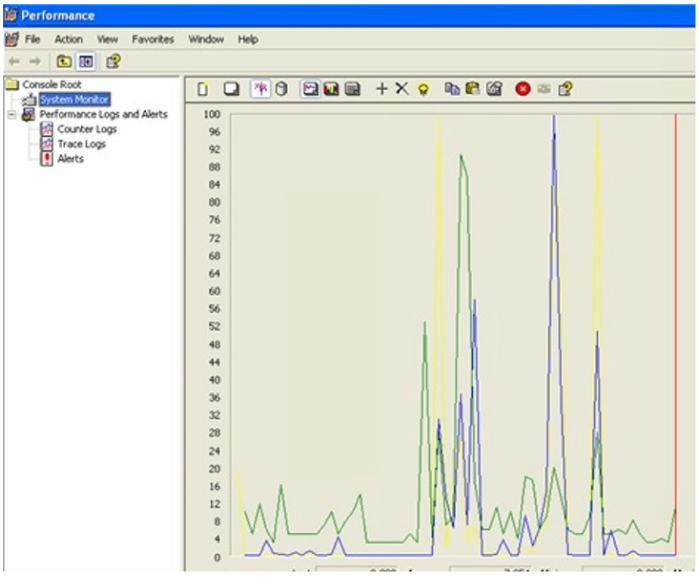

图 1-2 系统监视器（Perfmon）

我们可以写程序来查询 Perfmon 的值，Microsoft .Net Framework 提供了 PerformanceCounter()这一类型，从而可以方便地拿到当前各种计算机性能数据，包括 CPU 的使用率。例如下面这个程序——

【解法三】能动态适应的解法

代码清单 1-3

```
//C# code
static void MakeUsage(float level)
{
   PerformanceCounter p = new PerformanceCounter("Processor", "% Processor Time",
     "_Total");

   while (true)
   {
      if (p.NextValue() > level)
         System.Threading.Thread.Sleep(10);
    }
}
```

可以看到，上面的解法能方便地处理各种 CPU 使用率参数。这个程序可以解答前面提到的问题 2。

有了前面的积累，我们应该可以让任务管理器画出优美的正弦曲线了，见下面的代码。

【解法四】正弦曲线

代码清单 1-4

```
//C++ code  to make task manager generate sine graph
#include "Windows.h"
#include "stdlib.h"
#include "math.h"

const double SPLIT = 0.01;
const int COUNT = 200;
const double PI = 3.14159265;
const int INTERVAL = 300;

int _tmain(int argc, _TCHAR* argv[])
{
    DWORD busySpan[COUNT];  //array of busy times
    DWORD idleSpan[COUNT];  //array of idle times
    int half = INTERVAL / 2;
    double radian = 0.0;
    for(int i = 0; i < COUNT; i++)
    {
        busySpan[i] = (DWORD)(half + (sin(PI * radian) * half));
        idleSpan[i] = INTERVAL - busySpan[i];
        radian += SPLIT;
    }

    DWORD startTime = 0;
    int j = 0;
    while (true)
    {
        j = j % COUNT;
        startTime = GetTickCount();
        while ((GetTickCount() - startTime) <= busySpan[j]) ;
        Sleep(idleSpan[j]);
        j++;
    }
    return 0;
}
```

讨论

如果机器是多 CPU，上面的程序会出现什么结果？如何在多个 CPU 时显示同样的状态？例如，在双核的机器上，如果让一个单线程的程序死循环，能让两个 CPU 的使用率达到 50%的水平么？为什么？

多 CPU 的问题首先需要获得系统的 CPU 信息。可以使用 GetProcessorInfo()获得多处理器的信息,然后指定进程在哪一个处理器上运行。其中指定运行使用的是 SetThreadAffinityMask()函数。

另外，还可以使用 RDTSC 指令获取当前 CPU 核心运行周期数。

在 x86 平台上定义函数：

```
inline __int64 GetCPUTickCount()
{
    __asm
    {
       rdtsc;
    }
}
```

在 x64 平台上定义：

```
#define GetCPUTickCount() __rdtsc()
```

使用 CallNtPowerInformation API 得到 CPU 频率，从而将周期数转化为毫秒数，例如：

代码清单 1-5

```
_PROCESSOR_POWER_INFORMATION info;

CallNTPowerInformation(11,   //query processor power information
   NULL,               //no input buffer
   0,                 //input buffer size is zero
   &info,             //output buffer
   Sizeof(info));    //outbuf size

   __int64 t_begin = GetCPUTickCount();

   //do something

   __int64 t_end = GetCPUTickCount();
   double millisec = ((double)t_end –
      (double)t_begin)/(double)info.CurrentMhz;
```

RDTSC 指令读取当前 CPU 的周期数，在多 CPU 系统中，这个周期数在不同的 CPU 之间基数不同，频率也有可能不同。用从两个不同的 CPU 得到的周期数作计算会得出没有意义的值。如果线程在运行中被调度到了不同的 CPU，就会出现上述情况。可用 SetThreadAffinityMask 避免线程迁移。另外 ,CPU 的频率会随系统供电及负荷情况有所调整。

总结

能帮助你了解当前线程/进程/系统效能的 API 大致有以下这些：

1. Sleep()——这个方法能让当前线程“停”下来。
2. WaitForSingleObject()——自己停下来，等待某个事件发生
3. GetTickCount()——有人把 Tick 翻译成“嘀嗒”，很形象。
4. QueryPerformanceFrequency()、QueryPerformanceCounter()——让你访问到精度更高的 CPU 数据。
5. timeGetSystemTime()——是另一个得到高精度时间的方法。
6. PerformanceCounter——效能计数器。
7. GetProcessorInfo()/SetThreadAffinityMask()。遇到多核的问题怎么办呢？这两个方法能够帮你更好地控制 CPU。
8. GetCPUTickCount()。想拿到 CPU 核心运行周期数吗？用用这个方法吧。

了解并应用了上面的 API，就可以考虑在简历中写上“精通 Windows”了。