	线程数				
版本	1	2	4	8	16
psum-mutex	68.00	432.00	719.00	552.00	599.00
psum-array	7.26	3.64	1.91	1.85	1.84
psum-local	1.06	0.54	0.28	0.29	0.30

code/conc/psum-local.c

```
1   /* Thread routine for psum-local.c */
2   void *sum_local(void *vargp)
3   {
4       long myid = *((long *)vargp);          /* Extract the thread ID */
5       long start = myid * nelems_per_thread; /* Start element index */
6       long end = start + nelems_per_thread;  /* End element index */
7       long i, sum = 0;
8
9       for (i = start; i < end; i++) {
10          sum += i;
11      }
12      psum[myid] = sum;
13      return NULL;
14  }
```

code/conc/psum-local.c

图 12-34　psum-local 的线程例程。每个对等线程把它的部分和累积在一个局部变量中

从这个练习可以学习到一个重要的经验，那就是写并行程序相当棘手。对代码看上去很小的改动可能会对性能有极大的影响。

刻画并行程序的性能

图 12-35 给出了图 12-34 中程序 psum-local 的运行时间，它是线程数的函数。在每个情况下，程序运行在一个有四个处理器核的系统上，对一个 $n=2^{31}$ 个元素的序列求和。我们看到，随着线程数的增加，运行时间下降，直到增加到四个线程，此时，运行时间趋于平稳，甚至开始有点增加。

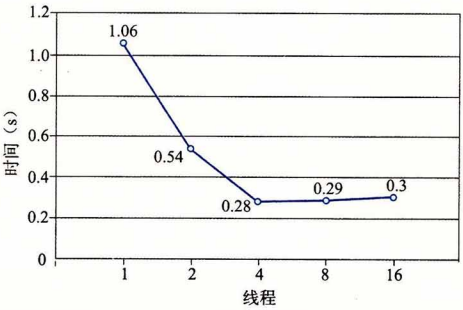

图 12-35　psum-local 的性能(图 12-34)。用四个处理器核对一个 2^{31} 个元素序列求和

在理想的情况中，我们会期望运行时间随着核数的增加线性下降。也就是说，我们会期望线程数每增加一倍，运行时间就下降一半。确实是这样，直到到达 $t>4$ 的时候，此时四个核中的每一个都忙于运行至少一个线程。随着线程数量的增加，运行时间实际上增加了一点儿，这是由于在一个核上多个线程上下文切换的开销。由于这个原因，并行程序常常被写为每个核上只运行一个线程。

虽然绝对运行时间是衡量程序性能的终极标准，但是还是有一些有用的相对衡量标准能够说明并行程序有多好地利用了潜在的并行性。并行程序的加速比(speedup)通常定义为