升的，然后变得平稳，之后又开始以比以前慢一些的速率下降。不过，由于多核处理器的出现(2004 年出现双核，2007 年出现四核)，有效周期时间以接近于以前的速率持续下降。

6.2　局部性

一个编写良好的计算机程序常常具有良好的*局部性*(locality)。也就是，它们倾向于引用邻近于其他最近引用过的数据项的数据项，或者最近引用过的数据项本身。这种倾向性，被称为*局部性原理*(principle of locality)，是一个持久的概念，对硬件和软件系统的设计和性能都有着极大的影响。

局部性通常有两种不同的形式：*时间局部性*(temporal locality)和*空间局部性*(spatial locality)。在一个具有良好时间局部性的程序中，被引用过一次的内存位置很可能在不远的将来再被多次引用。在一个具有良好空间局部性的程序中，如果一个内存位置被引用了一次，那么程序很可能在不远的将来引用附近的一个内存位置。

程序员应该理解局部性原理，因为一般而言，*有良好局部性的程序比局部性差的程序运行得更快*。现代计算机系统的各个层次，从硬件到操作系统、再到应用程序，它们的设计都利用了局部性。在硬件层，局部性原理允许计算机设计者通过引入称为*高速缓存存储器*的小而快速的存储器来保存最近被引用的指令和数据项，从而提高对主存的访问速度。在操作系统级，局部性原理允许系统使用主存作为虚拟地址空间最近被引用块的高速缓存。类似地，操作系统用主存来缓存磁盘文件系统中最近被使用的磁盘块。局部性原理在应用程序的设计中也扮演着重要的角色。例如，Web 浏览器将最近被引用的文档放在本地磁盘上，利用的就是时间局部性。大容量的 Web 服务器将最近被请求的文档放在前端磁盘高速缓存中，这些缓存能满足对这些文档的请求，而不需要服务器的任何干预。

6.2.1　对程序数据引用的局部性

考虑图 6-17a 中的简单函数，它对一个向量的元素求和。这个程序有良好的局部性吗？要回答这个问题，我们来看看每个变量的引用模式。在这个例子中，变量 sum 在每次循环迭代中被引用一次，因此，对于 sum 来说，有好的时间局部性。另一方面，因为 sum 是标量，对于 sum 来说，没有空间局部性。

```
1  int sumvec(int v[N])
2  {
3      int i, sum = 0;
4
5      for (i = 0; i < N; i++)
6          sum += v[i];
7      return sum;
8  }
```

a）一个具有良好局部性的程序

地址	0	4	8	12	16	20	24	28
内容	v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
访问顺序	1	2	3	4	5	6	7	8

b）向量v的引用模式（$N=8$）

图 6-17　注意如何按照向量元素存储在内存中的顺序来访问它们

正如我们在图 6-17b 中看到的，向量 v 的元素是被顺序读取的，一个接一个，按照它们存储在内存中的顺序(为了方便，我们假设数组是从地址 0 开始的)。因此，对于变量 v，函数有很好的空间局部性，但是时间局部性很差，因为每个向量元素只被访问一次。因为对于循环体中的每个变量，这个函数要么有好的空间局部性，要么有好的时间局部性，所以我们可以断定 sumvec 函数有良好的局部性。