如，即使类 *BC* 例程(2 个加载和 1 个存储)在内循环中比类 *AB* 例程(2 个加载)执行更多的内存引用，类 *BC* 例程(每次迭代有 0.5 个不命中)比类 *AB* 例程(每次迭代有 1.25 个不命中)性能还是要好很多。

- 对于大的 n 值，最快的一对版本(*kij* 和 *ikj*)的性能保持不变。虽然这个数组远大于任何 SRAM 高速缓存存储器，但预取硬件足够聪明，能够认出步长为 1 的访问模式，而且速度足够快能够跟上内循环中的内存访问。这是设计这个内存系统的 Intel 的工程师所做的一项极好成就，向程序员提供了甚至更多的鼓励，鼓励他们开发出具有良好空间局部性的程序。

网络旁注 MEM:BLOCKING　使用分块来提高时间局部性

有一项很有趣的技术，称为分块(blocking)，它可以提高内循环的时间局部性。分块的大致思想是将一个程序中的数据结构组织成的大的片(chunk)，称为块(block)。(在这个上下文中，“块”指的是一个应用级的数据组块，而不是高速缓存块。)这样构造程序，使得能够将一个片加载到 L1 高速缓存中，并在这个片中进行所需的所有的读和写，然后丢掉这个片，加载下一个片，依此类推。

与为提高空间局部性所做的简单循环变换不同，分块使得代码更难阅读和理解。由于这个原因，它最适合于优化编译器或者频繁执行的库函数。由于 Core i7 有完善的预取硬件，分块不会提高矩阵乘在 Core i7 上的性能。不过，学习和理解这项技术还是很有趣的，因为它是一个通用的概念，可以在一些没有预取的系统上获得极大的性能收益。

6.6.3　在程序中利用局部性

正如我们看到的，存储系统被组织成一个存储设备的层次结构，较小、较快的设备靠近顶部，较大、较慢的设备靠近底部。由于采用了这种层次结构，程序访问存储位置的实际速率不是一个数字能描述的。相反，它是一个变化很大的程序局部性的函数(我们称之为存储器山)，变化可以有几个数量级。有良好局部性的程序从快速的高速缓存存储器中访问它的大部分数据。局部性差的程序从相对慢速的 DRAM 主存中访问它的大部分数据。

理解存储器层次结构本质的程序员能够利用这些知识编写出更有效的程序，无论具体的存储系统结构是怎样的。特别地，我们推荐下列技术：

- 将你的注意力集中在内循环上，大部分计算和内存访问都发生在这里。
- 通过按照数据对象存储在内存中的顺序、以步长为 1 的来读数据，从而使得你程序中的空间局部性最大。
- 一旦从存储器中读入了一个数据对象，就尽可能多地使用它，从而使得程序中的时间局部性最大。

6.7　小结

基本存储技术包括随机存储器(RAM)、非易失性存储器(ROM)和磁盘。RAM 有两种基本类型。静态 RAM(SRAM)快一些，但是也贵一些，它既可以用做 CPU 芯片上的高速缓存，也可以用做芯片下的高速缓存。动态 RAM(DRAM)慢一点，也便宜一些，用做主存和图形帧缓冲区。即使是在关电的时候，ROM 也能保持它们的信息，可以用来存储固件。旋转磁盘是机械的非易失性存储设备，以每个位很低的成本保存大量的数据，但是其访问时间比 DRAM 长得多。固态硬盘(SSD)基于非易失性的闪存，对某些应用来说，越来越成为旋转磁盘的具有吸引力的替代产品。

一般而言，较快的存储技术每个位会更贵，而且容量更小。这些技术的价格和性能属性正在以显著