给出了一对测量某个读序列读吞吐量的函数。

test 函数通过以步长 stride 扫描一个数组的头 elems 个元素来产生读序列。为了提高内循环中可用的并行性，使用了 4×4 展开(见 5.9 节)。run 函数是一个包装函数，调用 test 函数，并返回测量出的读吞吐量。第 37 行对 test 函数的调用会对高速缓存做暖身。第 38 行的 fcyc2 函数以参数 elems 调用 test 函数，并估计 test 函数的运行时间，以 CPU 周期为单位。注意，run 函数的参数 size 是以字节为单位的，而 test 函数对应的参数 elems 是以数组元素为单位的。另外，注意第 39 行将 MB/s 计算为 10^6 字节/秒，而不是 2^{20} 字节/秒。

run 函数的参数 size 和 stride 允许我们控制产生出的读序列的时间和空间局部性程度。size 的值越小，得到的工作集越小，因此时间局部性越好。stride 的值越小，得到的空间局部性越好。如果我们反复以不同的 size 和 stride 值调用 run 函数，那么我们就能得到一个读带宽的时间和空间局部性的二维函数，称为*存储器山*(memory mountain)[112]。

每个计算机都有表明它存储器系统的能力特色的唯一的存储器山。例如，图 6-41 展示了 Intel Core i7 系统的存储器山。在这个例子中，size 从 16KB 变到 128KB，stride 从 1 变到 12 个元素，每个元素是一个 8 个字节的 long int。

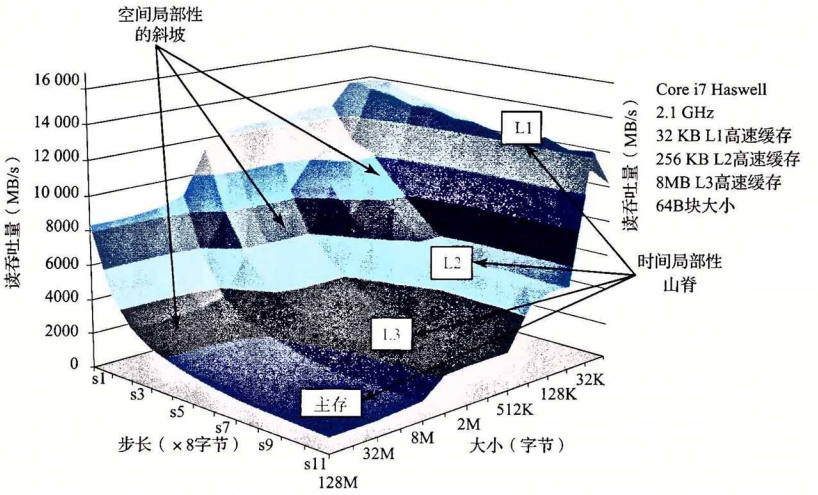

图 6-41　存储器山。展示了读吞吐量，它是时间和空间局部性的函数

这座 Core i7 山的地形地势展现了一个很丰富的结构。垂直于大小轴的是四条山脊，分别对应于工作集完全在 L1 高速缓存、L2 高速缓存、L3 高速缓存和主存内的时间局部性区域。注意，L1 山脊的最高点(那里 CPU 读速率为 14GB/s)与主存山脊的最低点(那里 CPU 读速率为 900MB/s)之间的差别有一个数量级。

在 L2、L3 和主存山脊上，随着步长的增加，有一个空间局部性的斜坡，空间局部性下降。注意，即使当工作集太大，不能全都装进任何一个高速缓存时，主存山脊的最高点也比它的最低点高 8 倍。因此，即使是当程序的时间局部性很差时，空间局部性仍然能补救，并且是非常重要的。