

因此,对于至少为8个字的步长来说,读吞吐量是一个常数速率,是由从L3传送高速缓存块到L2的速率决定的。

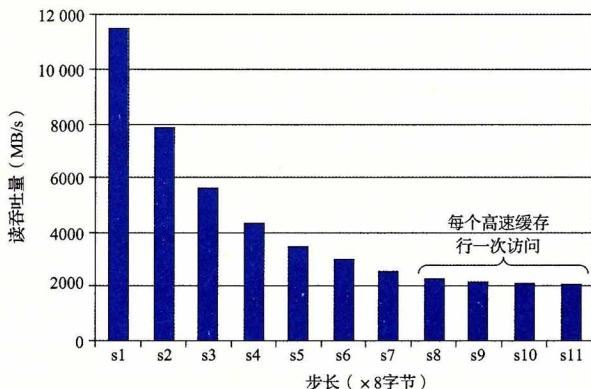



图 6-43 一个空间局部性的斜坡。这幅图展示了图 6-41 中大小=4MB 时的一个片段

总结一下我们对存储器山的讨论,存储器系统的性能不是一个数字就能描述的。相反,它是一座时间和空间局部性的山,这座山的上升高度差别可以超过一个数量级。明智的程序员会试图构造他们的程序,使得程序运行在山峰而不是低谷。目标就是利用时间局部性,使得频繁使用的字从L1中取出,还要利用空间局部性,使得尽可能多的字从一个L1高速缓存行中访问到。

 **练习题 6.21** 利用图 6-41 中的存储器山来估计从 L1 d-cache 中读一个 8 字节的字所需要的时间(以 CPU 周期为单位)。

### 6.6.2 重新排列循环以提高空间局部性

考虑一对  $n \times n$  矩阵相乘的问题:  $C=AB$ 。例如,如果  $n=2$ ,那么

$$\begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

其中

$$c_{11} = a_{11}b_{11} + a_{12}b_{21}$$

$$c_{12} = a_{11}b_{12} + a_{12}b_{22}$$

$$c_{21} = a_{21}b_{11} + a_{22}b_{21}$$

$$c_{22} = a_{21}b_{12} + a_{22}b_{22}$$

矩阵乘法函数通常是用 3 个嵌套的循环来实现的,分别用索引  $i$ 、 $j$  和  $k$  来标识。如果改变循环的次序,对代码进行一些其他的小改动,我们就能得到矩阵乘法的 6 个在功能上等价的版本,如图 6-44 所示。每个版本都以它循环的顺序来唯一地标识。

在高层次来看,这 6 个版本是非常相似的。如果加法是可结合的,那么每个版本计算出的结果完全一样<sup>①</sup>。每个版本总共都执行  $O(n^3)$  个操作,而加法和乘法的数量相同。A

① 正如我们在第 2 章中学到的,浮点加法是可交换的,但是通常是不可结合的。实际上,如果矩阵不把极大的数和极小的数混在一起——存储物理属性的矩阵常常这样,那么假设浮点加法是可结合的也是合理的。