实验六——缓存体系结构对性能的影响

一、 实验目的

1) 学习在微体系结构变化时算法的不同行为

2) 学习在相同微体系结构下改变算法如何改变性能

3) 提高对缓存体系结构的理解

二、 实验步骤

步骤 I：工作集大小

1）分析矩阵相乘应用的工作集大小

2）对于三种块配置中的每一种，分析一个块相乘的活动工作集大小

步骤 II：模拟和性能比较

1）在 HW4SmallCache 上运行所有的工作负载

2）在所有缓存层次下运行 BlockIJMatMulWorkload

步骤III：找到最佳设置

利用上一步结论来预测表现最佳的分块方案和缓存层次结构组合。

步骤Ⅳ：在本地硬件上运行

在 workloads/matmul 中使用命令 make all-native 来构建所有二进制文件，并在本地硬件上运行。

三、 实验结果

（一）步骤 I

1) 矩阵相乘应用的工作集大小是多少？将工作集大小描述为矩阵大

小（matrix_size）和双精度大小（double_size）的函数。使用 128 作为 matrix_size、8 作为 double_size，计算工作集大小。

工作集大小 = (3 * matrix_size * matrix_size) * double_size = 393216 字节

2) 对于三种块配置中的每一种，将一个块相乘的活动工作集是多少？将工作集大小描述为矩阵大小（matrix_size）、块大小 block_size）和双精度大小（double_size）的函数。使用 128 作为 matrix_size，8 作为 block_size，以及 8 作为 double_size 进行计算。

工作集大小 = (block_size * block_size + 2 * block_size * matrix_size) * double_size = 16896 字节

（二）步骤 II

设置 matrix_size 为 128、block_size 为 8 进行所有模拟。

1）对于 HW4SmallCache，哪种分块方案表现最好？为什么？

分块方案	BlockIJ	BlockIK	BlockKJ
运行时间（ms）	6.326	6.521	8.663
L1D 命中率	98.70%	97.18%	98.33%
L1I 命中率	99.87%	99.98%	99.66%
平均内存访问时间	1.763083	1.547919	2.351979

Cache 命中率数据来源：

```
board.cache_hierarchy.ruby_system.l1_controllers.L1Dcache.m_demand_hits     12547719
# Number of cache demand hits (Unspecified)
board.cache_hierarchy.ruby_system.l1_controllers.L1Dcache.m_demand_misses       165328
# Number of cache demand misses (Unspecified)
board.cache_hierarchy.ruby_system.l1_controllers.L1Dcache.m_demand_accesses     12713047
# Number of cache demand accesses (Unspecified)
board.cache_hierarchy.ruby_system.l1_controllers.L1Icache.m_demand_hits       747910
# Number of cache demand hits (Unspecified)
board.cache_hierarchy.ruby_system.l1_controllers.L1Icache.m_demand_misses          969
# Number of cache demand misses (Unspecified)
board.cache_hierarchy.ruby_system.l1_controllers.L1Icache.m_demand_accesses       748879
# Number of cache demand accesses (Unspecified)
```

BlockIJ 分块方案表现最好，因为其平均内存访问时间明显低于 BlockKJ，虽然略高于 BlockIK，但其 L1Dcache 命中率最高。

2)对于 BlockIJMatMulWorkload，哪种缓存层次表现最好？为什么？

缓存层次	Small	Medium	Large
时间（ms）	6.326	6.720	6.720
L1D 命中率	98.70%	99.32%	99.32%
L1I 命中率	99.87%	99.87%	99.87%
延迟	较低	中等	较高

HW4SmallCache 表现最好，虽然其 Cache 命中率略低于另外两种缓存层次，但其延迟较低，因而表现最好。

（三）步骤III

1）哪种分块方案和缓存层次结构的组合表现最佳？在你的答案中描述你找到这个组合的方法。请记住，不要穷尽搜索，而是尝试利用前一步的信息和统计数据找到最佳组合。

BlockIJ 分块方案和 HW4SmallCache 缓存层次结构的组合表现最佳。

原因：从步骤 II 的（1）中可以发现，在最小的 Cache 大小下，

BlockIJ 分块方案取得了最佳性能，因而可以推测即使 Cache 大小增加，BlockIJ 分块方案仍能表现最佳；步骤Ⅱ的（2）可以进一步验证，对于 BlockIJ 分块方案，在所有缓存层次结构中，HW4SmallCache 延迟最小、表现最佳。

2）在缓存的大小和延迟之间，你发现哪一个对性能有更显著的影响？

通过步骤Ⅱ的（2）可以发现，HW4SmallCache 的缓存最小，导致其 Cache 命中率略低于 HW4MediumCache 和 HW4LargeCache，但由于其延迟最小，HW4SmallCache 表现最佳。此外，HW4LargeCache 相比 HW4MediumCache，虽然缓存大小增加，但由于延迟较大，其性能没有得到提升。因此，相比较于缓存大小，延迟对性能有更显著的影响。

（四）步骤Ⅳ

1）你运行的处理器的 L1/L2/L3 大小是多少？

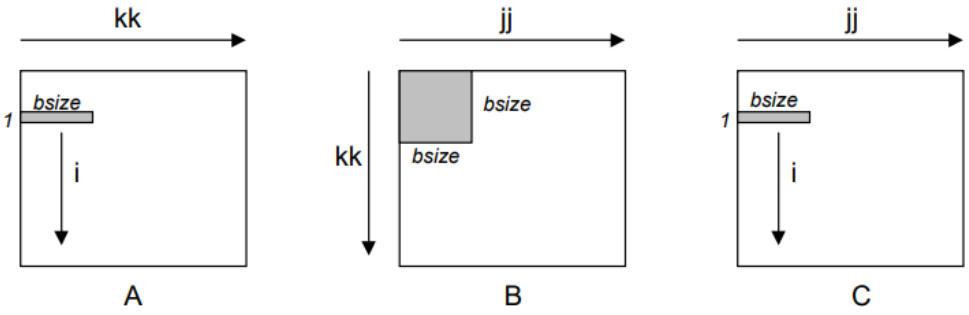
运行 lscpu 指令后结果如下所示：

```
Caches (sum of all):
  L1d:                    64 KiB (2 instances)
  L1i:                    64 KiB (2 instances)
  L2:                     512 KiB (2 instances)
  L3:                     12 MiB (2 instances)
```

2）你能否使用有关缓存大小的信息来预测最佳性能的分块方案和大小？

假设 matrix_size=512，double_size=8，当分块大小为 8 时，工作集大小为 66048 字节，与 L1Dcache 相差不大。而分块大小增加

后，工作集将远大于 L1Dcache。因此，最佳分块大小为 8。



由于数组存储方式是行主序，故最佳分块方案为 BlockKJ。

3）哪种分块方案和大小表现最佳？与 gem5 上的结果是否相同？为什么或为什么不？

矩阵大小：512，下表显示的是本地运行时间（单位 ms）。

方案 \ 大小	8	16	32	64
BlockIJ	410.105	560.862	804.782	702.073
BlockIK	529.925	643.004	739.292	717.855
BlockKJ	234.266	294.795	319.004	351.481

表现最佳：分块方案为 BlockKJ，分块大小为 8。

gem5 上的最佳分块方案是 BlockIJ，二者结果不同可能是延迟不同、缓存大小不同、以及真实硬件上工作负载不止一个等多种原因导致的。

四、 实验心得体会

通过本次实验，我了解了缓存体系结构对程序性能的影响，并掌握了如何利用矩阵分块的方法来充分利用缓存层次结构，以提升数据的局部性，从而优化矩阵乘法的运行时间。