Russell Ratcliffe

Parallel Processing

Lab 2

The purpose of this lab was to see the effect that different block sizes have on execution time for matrix multiplication, and to make predictions of cache sizes based on those results. The purpose of caching is to fetch information from main memory and store the information closer to the processor. This speeds up execution time since the processor can fetch memory from a cache much faster than it can get information from main memory.

Caching can speed of the process of matrix multiplication. To speed up matrix multiplication one would store blocks inside the cache. Execution time is the fastest when the block size is the largest size that can fit into level one cache.

Below is a table of my min execution times. The min execution time happened at a block size of 128, which is 32768 bytes (64^2 \*2 since it is a short). From these results I can infer that my processor has a level one cache size of 64kB.

|  |  |
| --- | --- |
| Block Size | Time (microseconds) |
| 32 | 10989331 |
| 64 | 10673424 |
| 128 | 10637040 |
| 256 | 10889264 |
| 512 | 10889264 |
| 1024 | 12536173 |
| 2048 | 12706305 |

Below is my code for implementing the matrix multiplication. The variable matrix\_size refers to a 1024x1024 two-dimensional array filled with 1s of type short. The variable block\_size refers to the size of the block. It was tested with all of the above block sizes.

The function gettimeofday() is used to measure the execution times. To get execution time one must subtract the start time from the end time.

struct timeval start, end;

gettimeofday(&start,NULL);

for (int k = 0; k < matrix\_size; k += block\_size)

for (int j = 0; j < matrix\_size; j += block\_size)

for (int i = 0; i < matrix\_size; i++)

for (int jj = j; jj < min(j + block\_size, matrix\_size); jj++)

for (int kk = k; kk < min(k + block\_size, matrix\_size); kk++)

C[i][jj] = C[i][jj] + A[i][kk] \* B[kk][jj];

gettimeofday(&end,NULL);

cout<<("%ld\n", ((end.tv\_sec \* 1000000 + end.tv\_usec)

- (start.tv\_sec \* 1000000 + start.tv\_usec)));

My code was written in C++ and compiled with GCC, the GNU Compiler Collection. I am running Fedora 24 with a Intel(R) Core(™) i5-2520M processor and 10gb of ram.

My hypothesis from the results of this experiment is that my computer’s cache is split up into three levels. The first level can store blocks under 64kb, and has the quickest fetch time. The second level can store blocks under 256kb and it has the second quickest fetch time. The third level stores blocks over 256kb. In my experiment, the size 128 block was the fastest since it took up a total of 32kb, and this is the largest block size out of the provided block sizes that can fit inside the level one cache. There is a large difference in execution time between a block size of 512 and 1024. This is because with a blocksize of 512, the computer will use the second level cache, and for a blocksize of 1024 the computer will use third level cache.

|  |
| --- |
|  |