GHENT UNIVERSITY

PARALLEL AND DISTRIBUTED SOFTWARE SYSTEMS

Homework Assignment 1

Tibo Vanheule

GHENT
UNIVERSITY

Academic Year 2023-2024

Q1.1: Explain how these performance differences are related to the CPU's cache sizes in a quantitative manner.

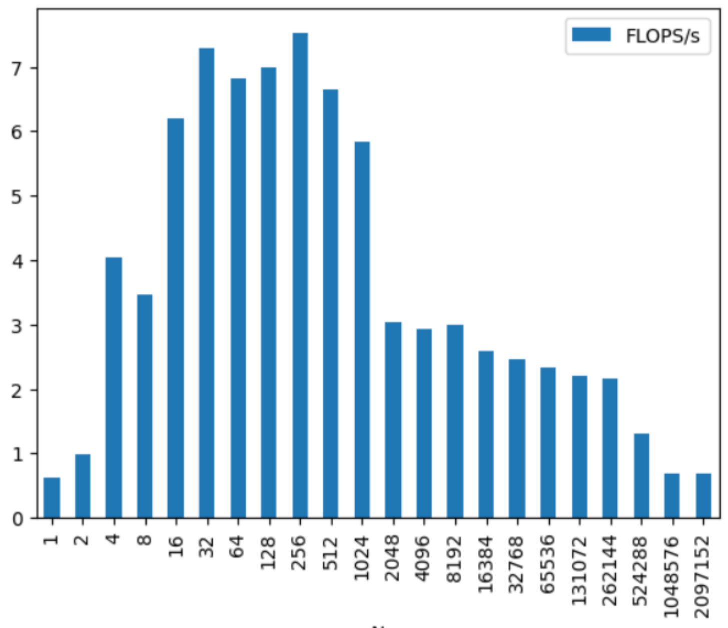

Figure 1: FLOPS/s per N, N in log scale

We see two big performance drops. First between 1024 and 2048. Second between 262.144 and 524.288. This gives an quick idea of which cache are used at point N.

We also see that there is significant loop overhead and hence poor performance for very small N.

L1 has a size of 32 KiB or 32768 Bytes. A element is 32 Bytes so L1 can store 32768/32=1024 elements. Afters this L1 cache misses occur which results in worse performance.

L2 has a size of 512 KiB or 524288 Bytes, so the number of elements L2 can store is 16.384. For this point L2 misses occur. A very huge drop in performance is absent. Probably due the similar retrieval speeds of L2 and L3.

L3 has a size of 16 MiB or 16777216 Bytes, so the number of elements L3 can store is 524.288. From this point DRAM Memory is also used resulting in even bad performance as n=1 (Loop overhead).

Q2.1: Report the runtime and the FLOPS/s for each of the four implementations.

Technology	**Running time**	**FLOPS/s**	**Control digit**
Non-contiguous access	66.7359s	1.13681e09	4.24661e13
Lineair access	25.3171s	2.99664e09	4.24661e13
Blocked	8.65364s	8.76696e09	4.24661e13
BLAS	0.315244s	2.40659e11	4.24661e13

Table 1: Running times and FLOP/s for each method of doing matrix-matrix multiplication

Q2.2: Report the D1 miss rate (as a percentage) for each of the four implementations and explain the results.

Technology	D1 Miss rate	D1 Write miss rate
Non-contiguous access	35.6%	0.6%
Lineair access	8.3%	0.1%
Blocked	2.1%	0.3%
BLAS	8.2%	9.1%

Table 2: D1 miss rate in percentage for each method of doing matrix-matrix multiplication

The results are in line with what is to be expected. The high running time is directly caused by the high D1 miss rate[1]. Except the openBLAS implementation is higher than initially expected.

BLAS: Since the running time of BLAS is higher than the other implementations, I think most misses will occur in L1 and fetches to the actual DRAM are rather limited. This idea is confirmed when looking at the LLd[2] metric of valgrind: 0.7%. So Most misses occur in either L1 or L2.

Non-contiguous access Strided memory access is the causes that the algorithm can not take advantage of temporal locality. A new cache line has to be red causing an eviction of another that is needed in the future.

Lineair This largely solves the temporal locality issue of Non-contiguous access. Moreover if we can assume that a complete row of matrix C can be kept in cache, the number of write-misses are significantly reduced. This is also confirmed in the table, as the Lineair access implementation has the lowest D1 write misses.

This does not provide perfect temporal locality for matrices bigger than our cache.

Blocked: We keep subsequent accesses to the same element close in time, to that the algorithm can benefit from temporal locality.

If the three matrices would fit in cache, we would have perfect temporal locality. Therefore, we reformulate the complete matrix-matrix product as a sequence of smaller matrix-matrix products of size bxb (where b is small enough such that 3b^{2} elements fit in cache.

Q3.1 How much time did you spend on this assignment, excluding the time spent studying the lecture notes.

About to 2-3 hours.

[1]The D1 miss rate of valgrind is a metric that indicates the percentage of data references that miss the first level of cache (L1) in a program.

[2]The LLd miss rate of valgrind is a metric that indicates the percentage of data references that miss the last level of cache (LL) in a program.