Sena Engin/se042092

Multicore Programming Task 1

Used Google cloud service Ubuntu 20.04 disturbion and using the SSH connection for this task.

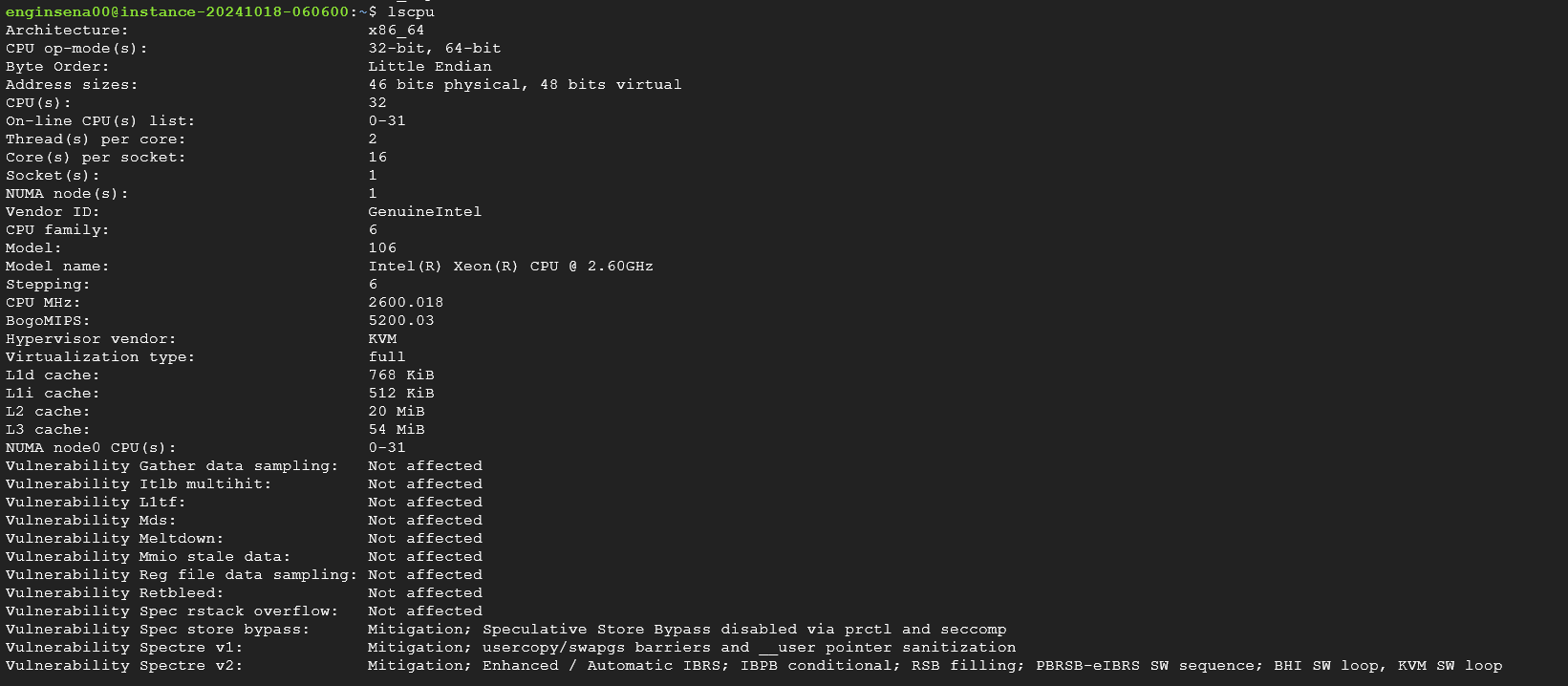
Only for the memory access analyses , I used the Valgrind because Vtune does not give the permission for this analysis.

Used these link steps for adding the profile with Intel VTune :  
<https://www.intel.com/content/www/us/en/docs/vtune-profiler/installation-guide/2024-2/package-managers.html>

QUESTION 1

Because of i am using Cloud service for this Project i have to write my remote computer .

With lscpu:



It can be seen that my processor architecture is x86- based .

X86\_64 architecture means that my processor can handle 64-bit instructions.

CPU number is 32 , represents the total number of virtual CPUs on my VM.

Cores per socket means that 16 physical core machine has got.

Threads per core meaning each core supports these number of threads.

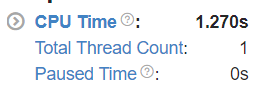
Model name: Intel(R) Xeon(R) CPU @ 2.60GHz, is my server processor model.

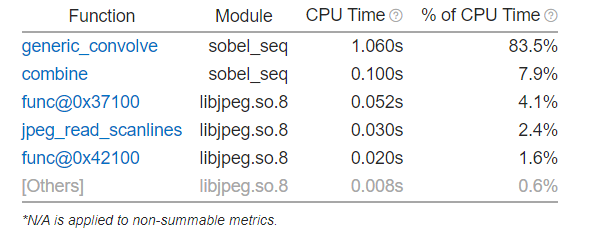
L1,L2 and L3 caches are caches from smallest to the biggest storage but faster to the slowest caches.

QUESTION 2

a) Perform an analysis to see how much time the sequential version of sobel filter spends on various part of the code (e.g., in Intel Vtune that corresponds to “hotspots” analysis). Report the result of the analysis and explain which part of the program is the most time consuming

vtune -collect hotspots -result-dir vtune\_hotspots\_seq -- ./sobel\_seq input1.jpg

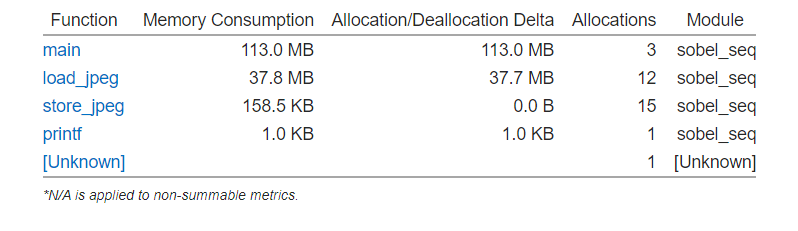




It can be seen that generic\_convolve function is the most time consuming at the serial program .

b) Perform an analyis to see how much memory “consumption” the sequential version of sobel filter has, and report the findings.

vtune -collect memory-consumption -result-dir vtune\_memory\_consumption\_seq -- ./sobel\_seq input1.jpg

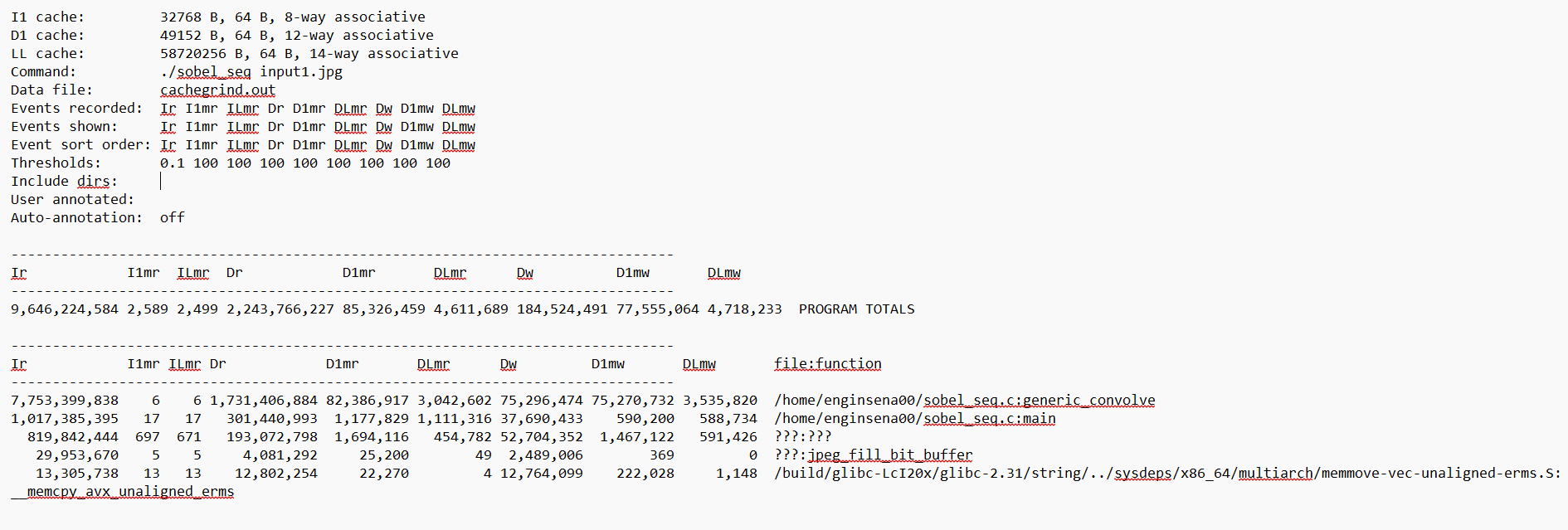


In general , main functions taken most part of the consumption.

c) Perform “memory access” analysis to identify cache misses and bandwidth usage of the sequential version of sobel filter, and report the findings.

valgrind --tool=cachegrind --cachegrind-out-file=cachegrind.out ./sobel\_seq input1.jpg

cg\_annotate cachegrind.out > cachegrind\_report.txt



I put the cache misses txt files into the reports folder.

QUESTION 3

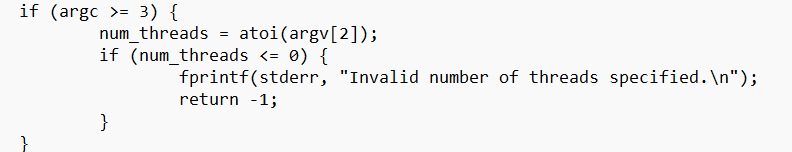
Because of the generic\_convolve function is the most time consuming ,I parallelized the loop part of this function.

My strategy is the dividing the sobel\_X and sobel\_Y edge threads as a two threads and apply this for multiple threads.

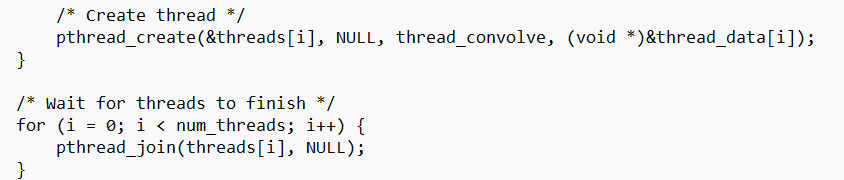
As a difference of the serial code,

I added the start and end rows for each thread.

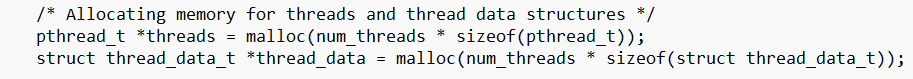
Getting the thread number from command line:

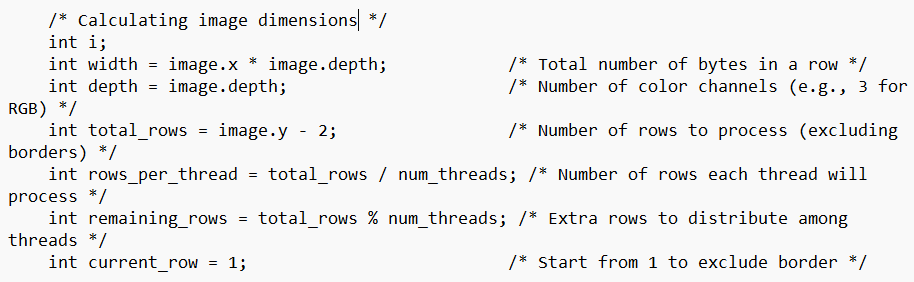


Pthread\_create and join to wait all threads to finish.

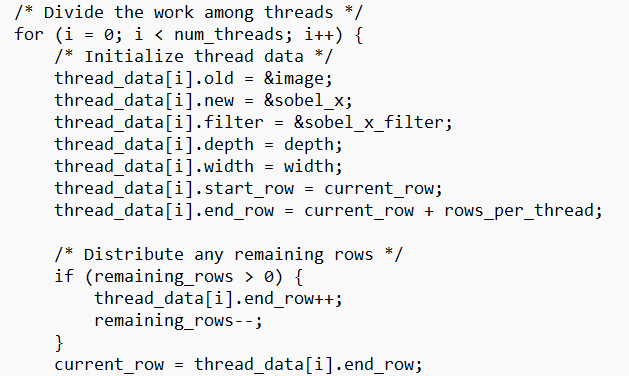


Allocated the threads with malloc.

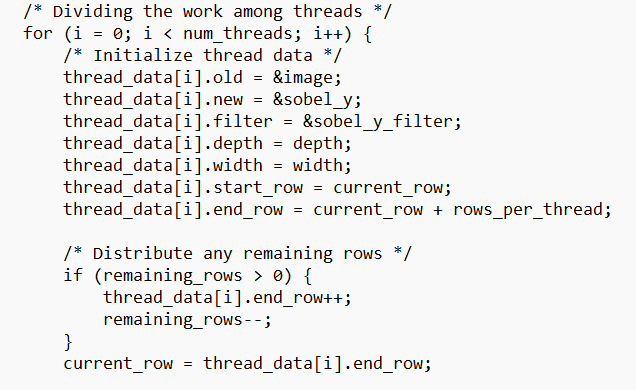


It is necessary to calculate image dimensions for all the threads:  


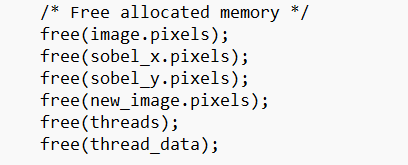
Then dividing the work among threads for sobel\_x filter:



And dividing the sobel\_y filter work among threads too:



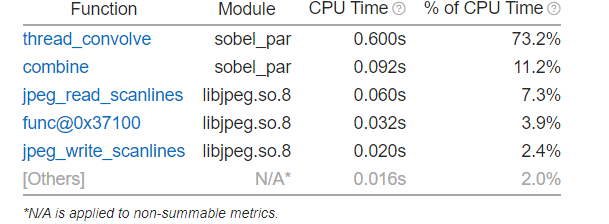
Lastly , freed the memory.



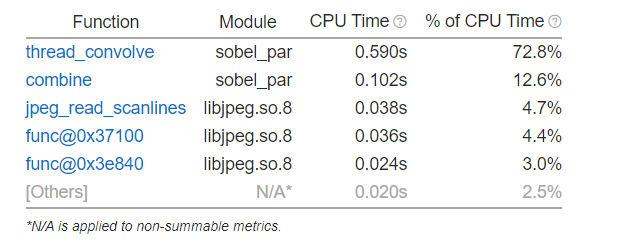
QUESTION 4

1. Perform an analysis to see how much time the parallel version of sobel filter spends on various functions (e.g., “hotspots” analysis in Vtune) when different number of threads are used. You should perform separate analysis for thread counts of 2, 4, 8, 16 and 32. Report the results similar to above.

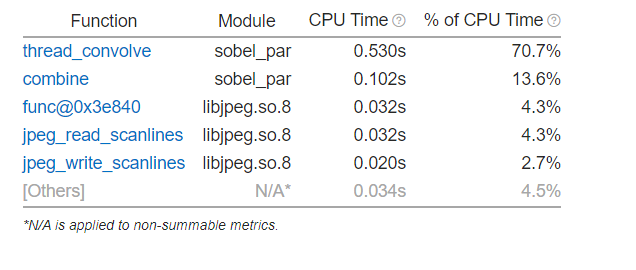
For 2 Thread:



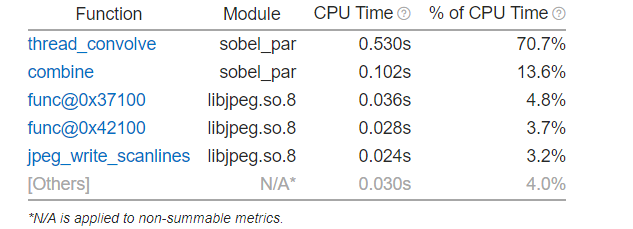
For 4 Thread:



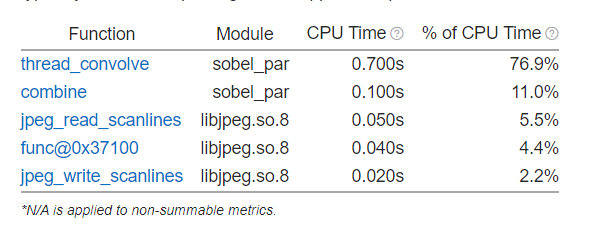
For 8 Thread:



For 16 Thread:



For 32 Thread:

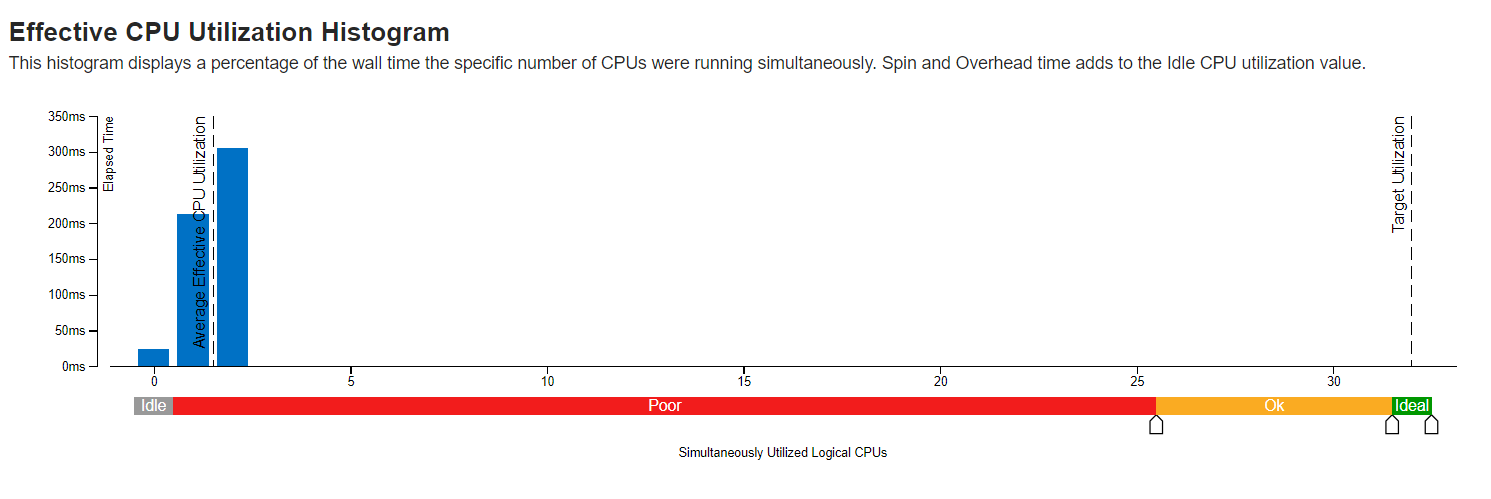


We can see that increasing the thread counts, decreases the operation time.

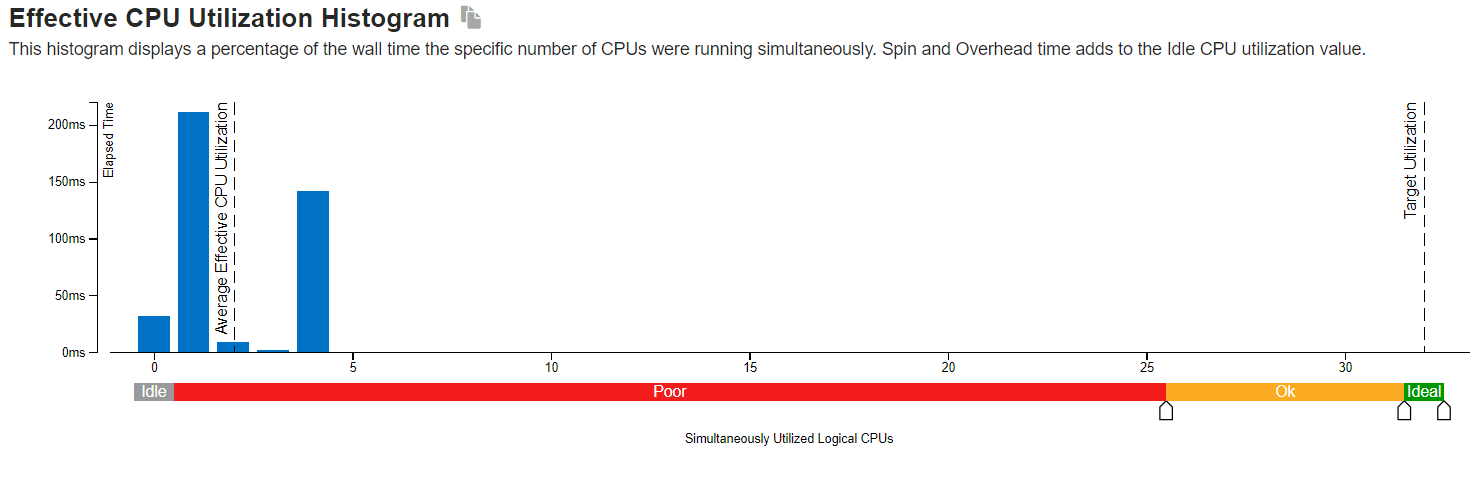
This means that my parallelization is effectively works.

1. Perform “threading” analysis for thread counts of 2, 4, 8, 16 and 32, to see how efficiently an application uses available cores. Report the results similar to above.

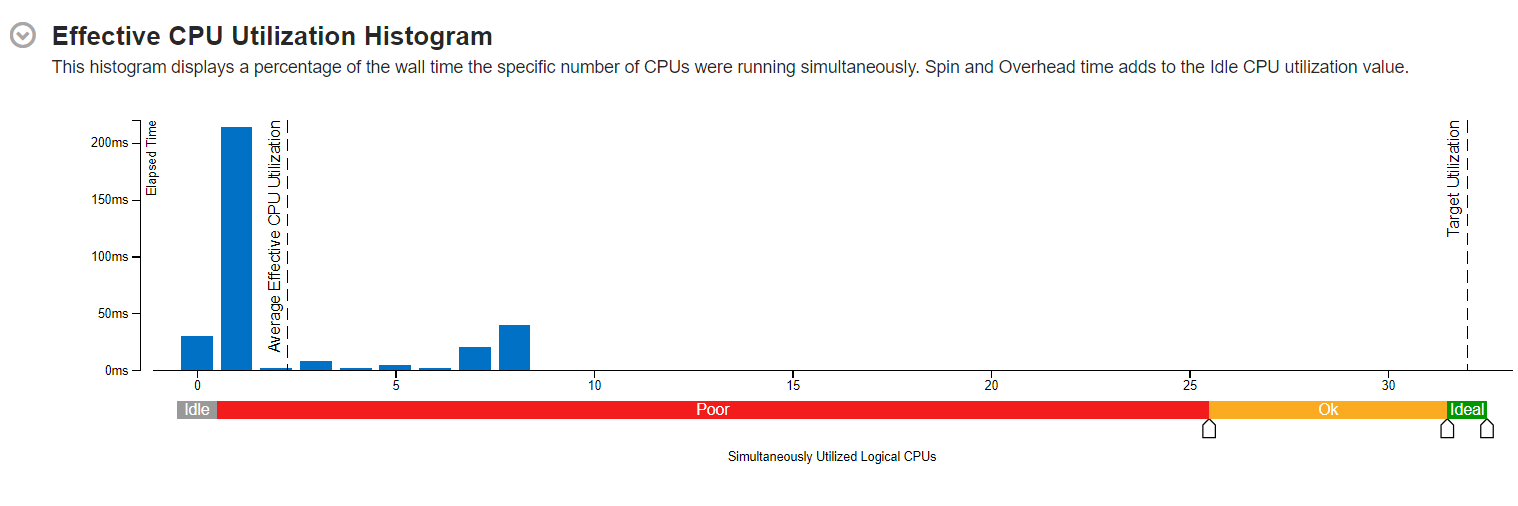
For 2 thread:



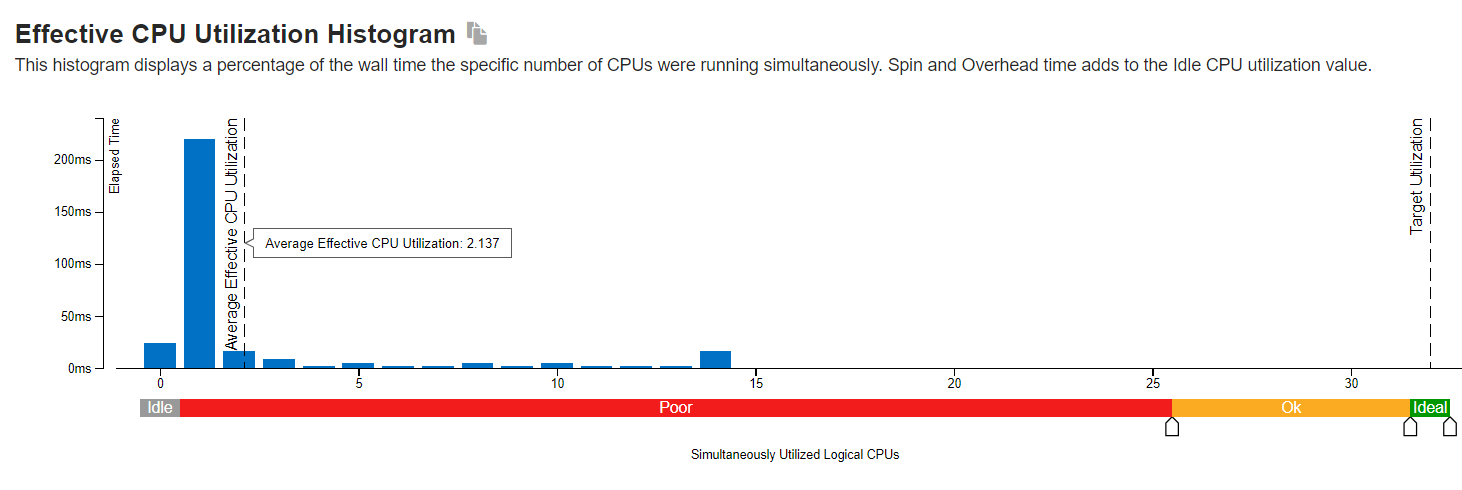
For 4 thread:



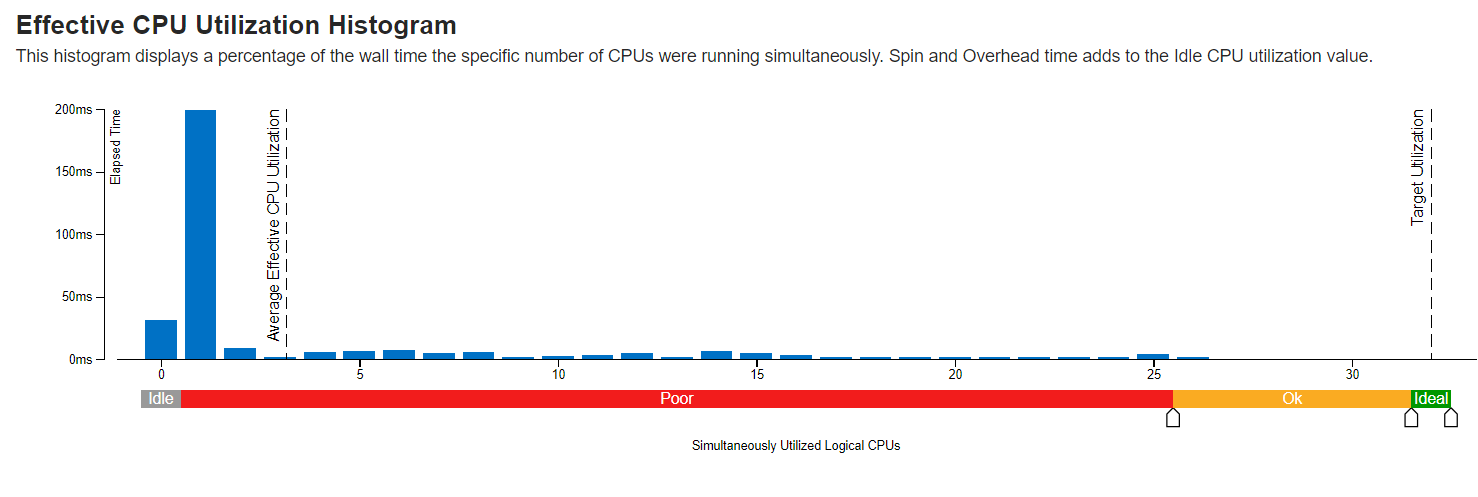
For 8 thread:



For 16 thread:



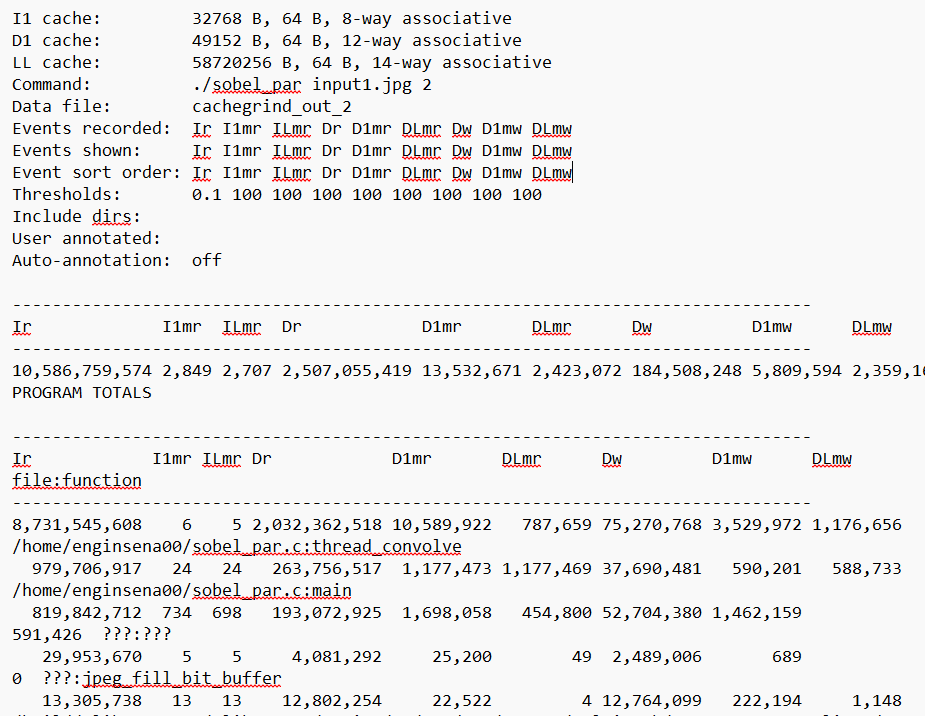
For 32 thread:



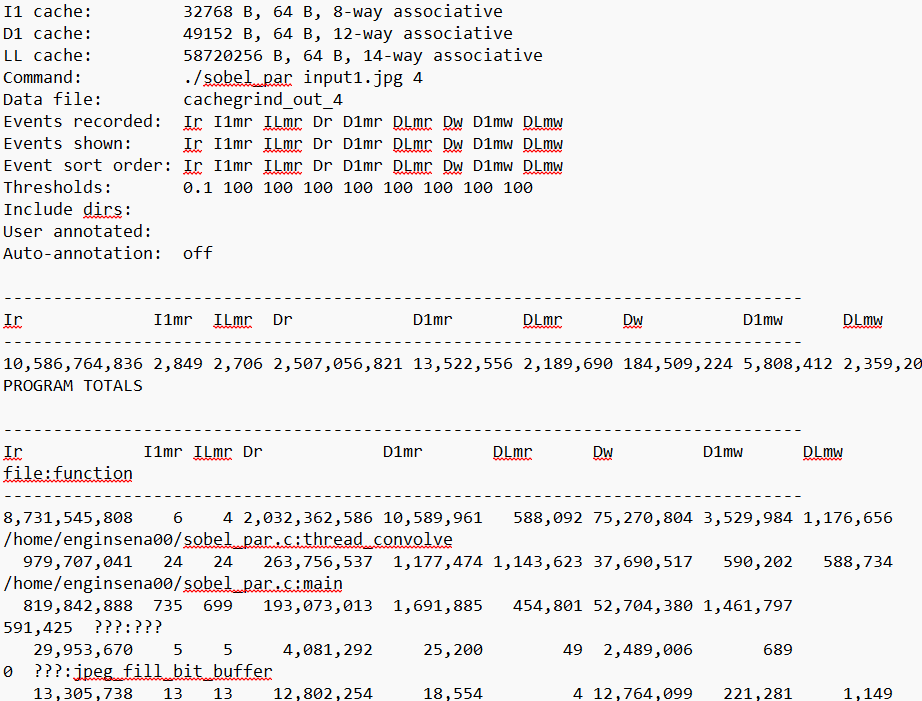
If we look at the CPU diagrams, CPU utilization peaks at the 2 threads using but with the 32 threads may be all of the threads are not actively running so not reach the peak and as we see it is decreasing.

1. Perform “memory access” analyis for thread counts of 2, 4, 8, 16 and 32. Report the results similar to above.

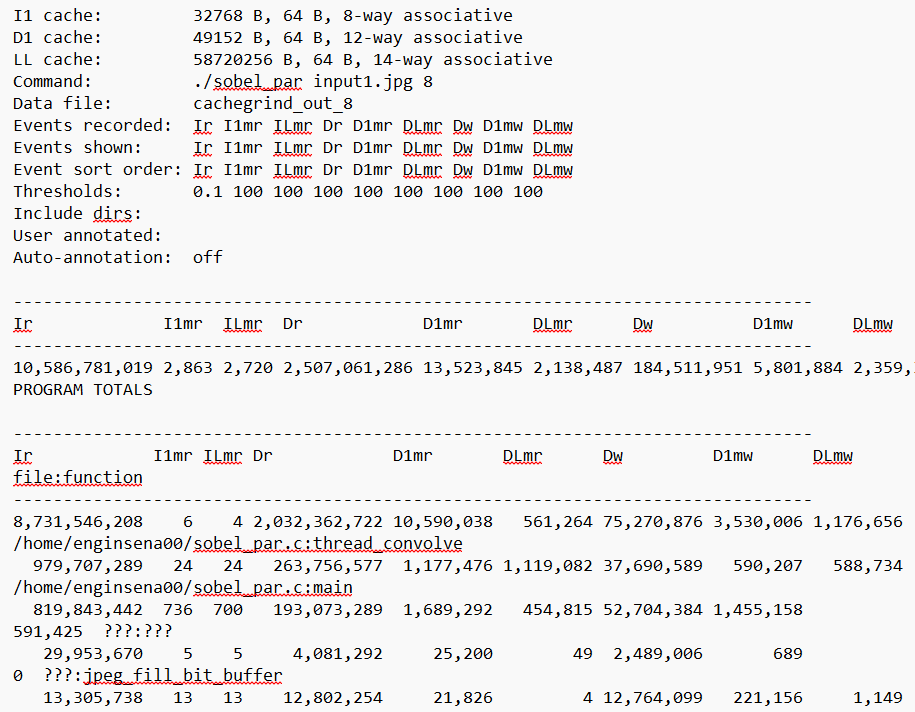
For 2 thread:



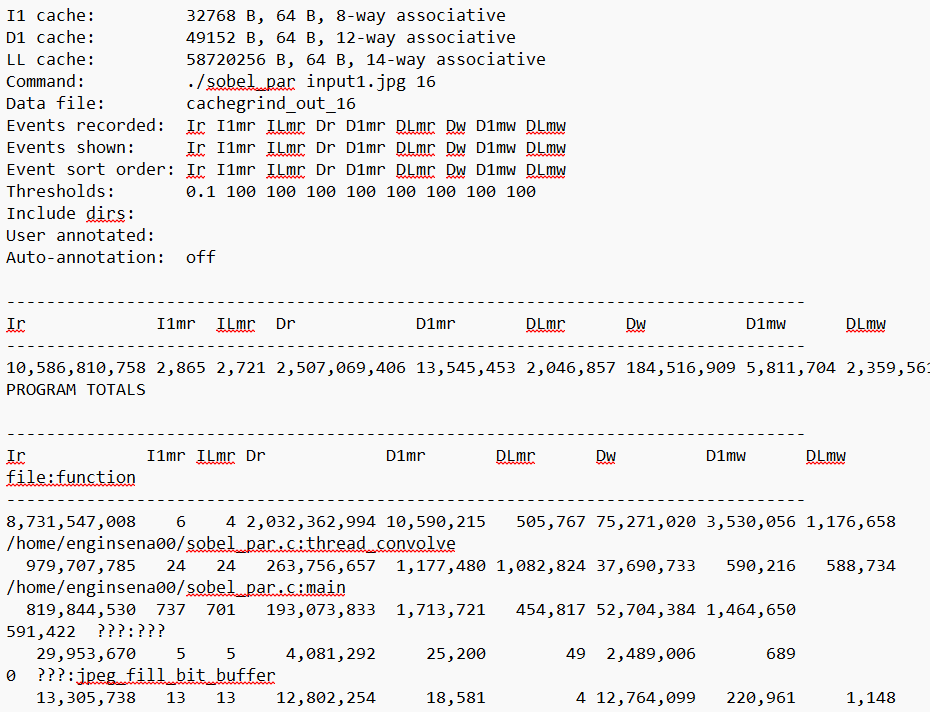
For 4 thread:



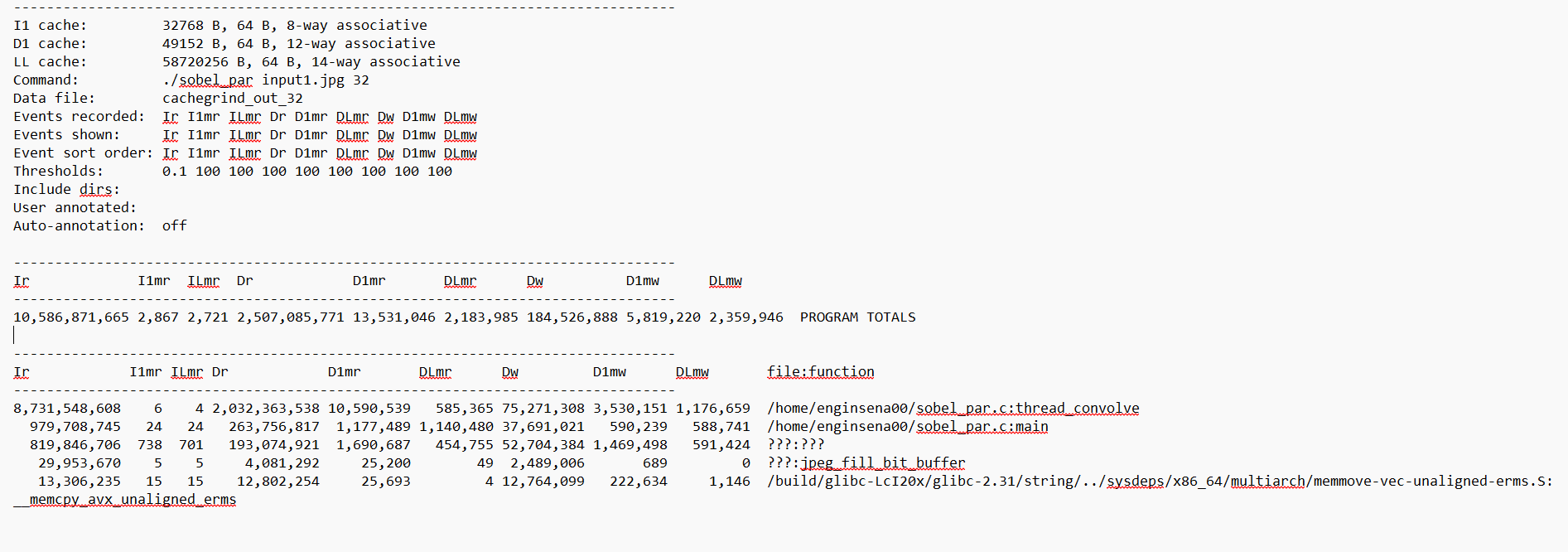
For 8 thread:



For 16 thread:



For 32 thread:



Ir means instruction counts and actually it is increasing from 2 threads to the 32 threads.

I1 and IL means that level 1 and IL is the last level instruction caches.

D1 and DL is the data caches.

Mr-read misses

Mw-write misses

We can see that increasing the number of threads does not significantly much impact cache misses.