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:

```
enginsena00@instance-20241018-060600:~$ lscpu
Architecture:                       x86_64
CPU op-mode(s):                     32-bit, 64-bit
Byte Order:                         Little Endian
Address sizes:                      46 bits physical, 48 bits virtual
CPU(s):                             32
On-line CPU(s) list:                0-31
Thread(s) per core:                 2
Core(s) per socket:                 16
Socket(s):                          1
NUMA node(s):                       1
Vendor ID:                          GenuineIntel
CPU family:                         6
Model:                              106
Model name:                         Intel(R) Xeon(R) CPU @ 2.60GHz
Stepping:                           6
CPU MHz:                            2600.018
BogoMIPS:                           5200.03
Hypervisor vendor:                  KVM
Virtualization type:                full
L1d cache:                          768 KiB
L1i cache:                          512 KiB
L2 cache:                           20 MiB
L3 cache:                           54 MiB
NUMA node0 CPU(s):                  0-31
Vulnerability Gather data sampling: Not affected
Vulnerability Itlb multihit:        Not affected
Vulnerability L1tf:                 Not affected
Vulnerability Mds:                  Not affected
Vulnerability Meltdown:             Not affected
Vulnerability Mmio stale data:      Not affected
Vulnerability Reg file data sampling: Not affected
Vulnerability Retbleed:             Not affected
Vulnerability Spec rstack overflow: Not affected
Vulnerability Spec store bypass:    Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:           Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:           Mitigation; Enhanced / Automatic IBRS; IBPB conditional; RSB filling; PBRSB-eIBRS SW sequence; BHI SW loop, KVM SW loop
```

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

CPU Time:	**1.270s**
Total Thread Count:	1
Paused Time:	0s

Function	Module	CPU Time	% of CPU Time
generic_convolve	sobel_seq	1.060s	83.5%
combine	sobel_seq	0.100s	7.9%
func@0x37100	libjpeg.so.8	0.052s	4.1%
jpeg_read_scanlines	libjpeg.so.8	0.030s	2.4%
func@0x42100	libjpeg.so.8	0.020s	1.6%
[Others]	libjpeg.so.8	0.008s	0.6%

N/A is applied to non-summable metrics.

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

Function	Memory Consumption	Allocation/Deallocation Delta	Allocations	Module
main	113.0 MB	113.0 MB	3	sobel_seq
load_jpeg	37.8 MB	37.7 MB	12	sobel_seq
store_jpeg	158.5 KB	0.0 B	15	sobel_seq
printf	1.0 KB	1.0 KB	1	sobel_seq
[Unknown]			1	[Unknown]

N/A is applied to non-summable metrics.

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

```
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         49152 B, 64 B, 12-way associative
LL cache:         58720256 B, 64 B, 14-way associative
Command:          ./sobel_seq input1.jpg
Data file:        cachegrind.out
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:     |
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
Ir            I1mr  ILmr  Dr            D1mr        DLmr       Dw          D1mw        DLmw
--------------------------------------------------------------------------------
9,646,224,584 2,589 2,499 2,243,766,227 85,326,459 4,611,689 184,524,491 77,555,064 4,718,233  PROGRAM TOTALS

--------------------------------------------------------------------------------
Ir            I1mr ILmr Dr            D1mr       DLmr      Dw         D1mw       DLmw       file:function
--------------------------------------------------------------------------------
7,753,399,838    6    6 1,731,406,884 82,386,917 3,042,602 75,296,474 75,270,732 3,535,820  /home/enginsena00/sobel_seq.c:generic_convolve
1,017,385,395   17   17   301,440,993  1,177,829 1,111,316 37,690,433    590,200   588,734  /home/enginsena00/sobel_seq.c:main
  819,842,444  697  671   193,072,798  1,694,116   454,782 52,704,352  1,467,122   591,426  ???:???
   29,953,670    5    5     4,081,292     25,200        49  2,489,006        369         0  ???:jpeg_fill_bit_buffer
   13,305,738   13   13    12,802,254     22,270         4 12,764,099    222,028     1,148  /build/glibc-LcI20x/glibc-2.31/string/../sysdeps/x86_64/multiarch/memmove-vec-unaligned-erms.S:
__memcpy_avx_unaligned_erms
```

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:

```
if (argc >= 3) {
        num_threads = atoi(argv[2]);
        if (num_threads <= 0) {
                fprintf(stderr, "Invalid number of threads specified.\n");
                return -1;
        }
}
```

Pthread_create and join to wait all threads to finish.

```
        /* Create thread */
        pthread_create(&threads[i], NULL, thread_convolve, (void *)&thread_data[i]);
    }

    /* Wait for threads to finish */
    for (i = 0; i < num_threads; i++) {
        pthread_join(threads[i], NULL);
    }
```

Allocated the threads with malloc.

```
   /* Allocating memory for threads and thread data structures */
   pthread_t *threads = malloc(num_threads * sizeof(pthread_t));
   struct thread_data_t *thread_data = malloc(num_threads * sizeof(struct thread_data_t));
```

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

```
    /* Calculating image dimensions */
    int i;
    int width = image.x * image.depth;            /* Total number of bytes in a row */
    int depth = image.depth;                      /* Number of color channels (e.g., 3 for
RGB) */
    int total_rows = image.y - 2;                 /* Number of rows to process (excluding
borders) */
    int rows_per_thread = total_rows / num_threads; /* Number of rows each thread will
process */
    int remaining_rows = total_rows % num_threads; /* Extra rows to distribute among
threads */
    int current_row = 1;                          /* Start from 1 to exclude border */
```

Then dividing the work among threads for sobel_x filter:

```
 /* Divide the work among threads */
 for (i = 0; i < num_threads; i++) {
     /* Initialize thread data */
     thread_data[i].old = &image;
     thread_data[i].new = &sobel_x;
     thread_data[i].filter = &sobel_x_filter;
     thread_data[i].depth = depth;
     thread_data[i].width = width;
     thread_data[i].start_row = current_row;
     thread_data[i].end_row = current_row + rows_per_thread;

     /* Distribute any remaining rows */
     if (remaining_rows > 0) {
         thread_data[i].end_row++;
         remaining_rows--;
     }
     current_row = thread_data[i].end_row;
```

And dividing the sobel_y filter work among threads too:

```
/* Dividing the work among threads */
for (i = 0; i < num_threads; i++) {
    /* Initialize thread data */
    thread_data[i].old = &image;
    thread_data[i].new = &sobel_y;
    thread_data[i].filter = &sobel_y_filter;
    thread_data[i].depth = depth;
    thread_data[i].width = width;
    thread_data[i].start_row = current_row;
    thread_data[i].end_row = current_row + rows_per_thread;

    /* Distribute any remaining rows */
    if (remaining_rows > 0) {
        thread_data[i].end_row++;
        remaining_rows--;
    }
    current_row = thread_data[i].end_row;
```

Lastly , freed the memory.

```
/* Free allocated memory */
free(image.pixels);
free(sobel_x.pixels);
free(sobel_y.pixels);
free(new_image.pixels);
free(threads);
free(thread_data);
```

QUESTION 4

a) 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:

Function	Module	CPU Time	% of CPU Time
thread_convolve	sobel_par	0.600s	73.2%
combine	sobel_par	0.092s	11.2%
jpeg_read_scanlines	libjpeg.so.8	0.060s	7.3%
func@0x37100	libjpeg.so.8	0.032s	3.9%
jpeg_write_scanlines	libjpeg.so.8	0.020s	2.4%
[Others]	N/A*	0.016s	2.0%

*N/A is applied to non-summable metrics.

For 4 Thread:

Function	Module	CPU Time	% of CPU Time
thread_convolve	sobel_par	0.590s	72.8%
combine	sobel_par	0.102s	12.6%
jpeg_read_scanlines	libjpeg.so.8	0.038s	4.7%
func@0x37100	libjpeg.so.8	0.036s	4.4%
func@0x3e840	libjpeg.so.8	0.024s	3.0%
[Others]	N/A*	0.020s	2.5%

*N/A is applied to non-summable metrics.

For 8 Thread:

Function	Module	CPU Time	% of CPU Time
thread_convolve	sobel_par	0.530s	70.7%
combine	sobel_par	0.102s	13.6%
func@0x3e840	libjpeg.so.8	0.032s	4.3%
jpeg_read_scanlines	libjpeg.so.8	0.032s	4.3%
jpeg_write_scanlines	libjpeg.so.8	0.020s	2.7%
[Others]	N/A*	0.034s	4.5%

*N/A is applied to non-summable metrics.

For 16 Thread:

Function	Module	CPU Time	% of CPU Time
thread_convolve	sobel_par	0.530s	70.7%
combine	sobel_par	0.102s	13.6%
func@0x37100	libjpeg.so.8	0.036s	4.8%
func@0x42100	libjpeg.so.8	0.028s	3.7%
jpeg_write_scanlines	libjpeg.so.8	0.024s	3.2%
[Others]	N/A*	0.030s	4.0%

*N/A is applied to non-summable metrics.

For 32 Thread:

Function	Module	CPU Time	% of CPU Time
thread_convolve	sobel_par	0.700s	76.9%
combine	sobel_par	0.100s	11.0%
jpeg_read_scanlines	libjpeg.so.8	0.050s	5.5%
func@0x37100	libjpeg.so.8	0.040s	4.4%
jpeg_write_scanlines	libjpeg.so.8	0.020s	2.2%

**N/A is applied to non-summable metrics.*

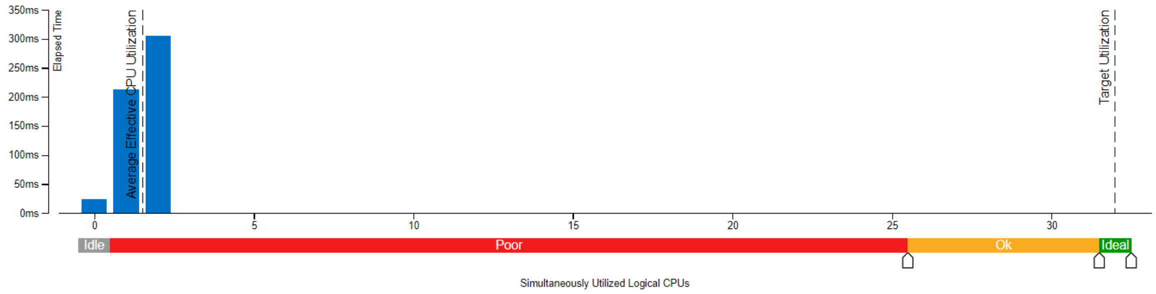
We can see that increasing the thread counts, decreases the operation time. This means that my parallelization is effectively works.

b) 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:

Effective CPU Utilization Histogram

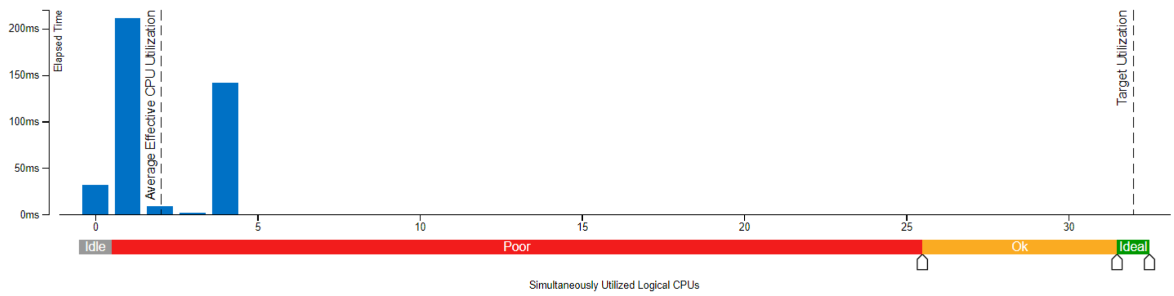
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



For 4 thread:

Effective CPU Utilization Histogram

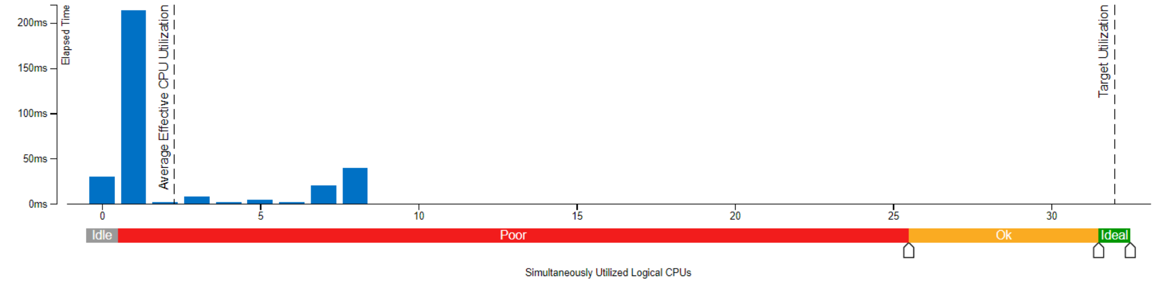
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



For 8 thread:

Effective CPU Utilization Histogram

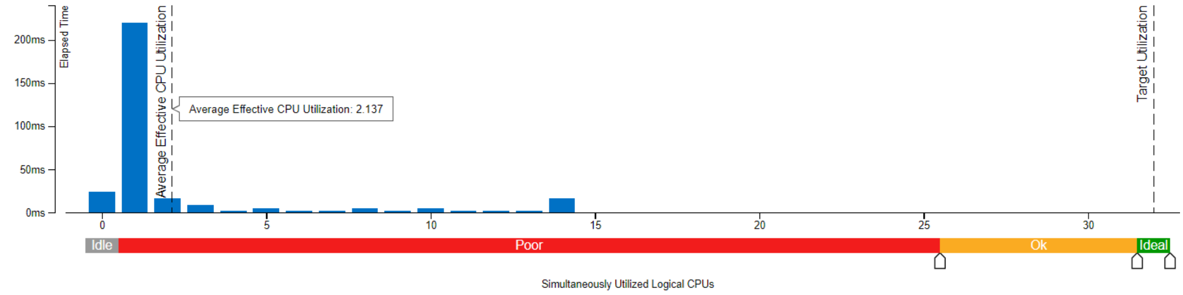
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



For 16 thread:

Effective CPU Utilization Histogram

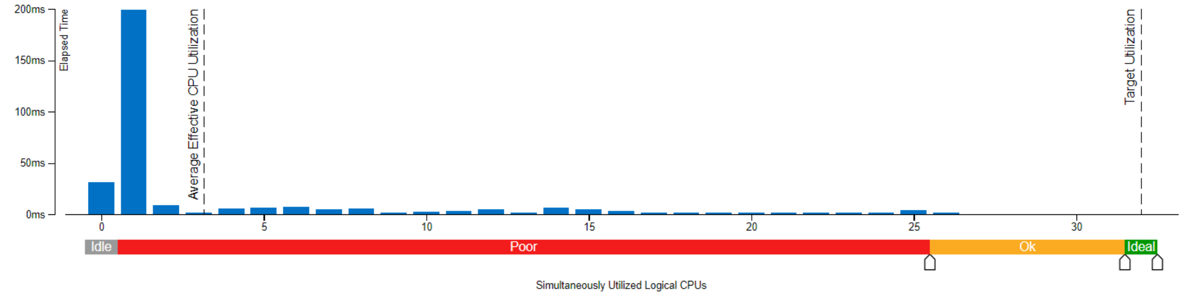
This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



For 32 thread:

Effective CPU Utilization Histogram

This histogram displays a percentage of the wall time the specific number of CPUs were running simultaneously. Spin and Overhead time adds to the Idle CPU utilization value.



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.

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

For 2 thread:

```
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         49152 B, 64 B, 12-way associative
LL cache:         58720256 B, 64 B, 14-way associative
Command:          ./sobel_par input1.jpg 2
Data file:        cachegrind_out_2
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
Ir             I1mr  ILmr  Dr            D1mr       DLmr      Dw          D1mw      DLmw
--------------------------------------------------------------------------------
10,586,759,574 2,849 2,707 2,507,055,419 13,532,671 2,423,072 184,508,248 5,809,594 2,359,1
PROGRAM TOTALS

--------------------------------------------------------------------------------
Ir            I1mr ILmr Dr            D1mr       DLmr      Dw          D1mw      DLmw
file:function
--------------------------------------------------------------------------------
8,731,545,608    6    5 2,032,362,518 10,589,922   787,659 75,270,768 3,529,972 1,176,656
/home/enginsena00/sobel_par.c:thread_convolve
  979,706,917   24   24   263,756,517  1,177,473 1,177,469 37,690,481   590,201   588,733
/home/enginsena00/sobel_par.c:main
  819,842,712  734  698   193,072,925  1,698,058   454,800 52,704,380 1,462,159
591,426  ???:???
   29,953,670    5    5     4,081,292     25,200        49  2,489,006       689
0  ???:jpeg_fill_bit_buffer
   13,305,738   13   13    12,802,254     22,522         4 12,764,099   222,194     1,148
```

For 4 thread:

```
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         49152 B, 64 B, 12-way associative
LL cache:         58720256 B, 64 B, 14-way associative
Command:          ./sobel_par input1.jpg 4
Data file:        cachegrind_out_4
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
Ir             I1mr  ILmr  Dr            D1mr       DLmr      Dw          D1mw      DLmw
--------------------------------------------------------------------------------
10,586,764,836 2,849 2,706 2,507,056,821 13,522,556 2,189,690 184,509,224 5,808,412 2,359,20
PROGRAM TOTALS

--------------------------------------------------------------------------------
Ir            I1mr ILmr Dr            D1mr       DLmr      Dw          D1mw      DLmw
file:function
--------------------------------------------------------------------------------
8,731,545,808    6    4 2,032,362,586 10,589,961   588,092 75,270,804 3,529,984 1,176,656
/home/enginsena00/sobel_par.c:thread_convolve
  979,707,041   24   24   263,756,537  1,177,474 1,143,623 37,690,517   590,202   588,734
/home/enginsena00/sobel_par.c:main
  819,842,888  735  699   193,073,013  1,691,885   454,801 52,704,380 1,461,797
591,425  ???:???
   29,953,670    5    5     4,081,292     25,200        49  2,489,006       689
0  ???:jpeg_fill_bit_buffer
   13,305,738   13   13    12,802,254     18,554         4 12,764,099   221,281     1,149
```

For 8 thread:

```
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         49152 B, 64 B, 12-way associative
LL cache:         58720256 B, 64 B, 14-way associative
Command:          ./sobel_par input1.jpg 8
Data file:        cachegrind_out_8
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
Ir             I1mr  ILmr  Dr            D1mr        DLmr       Dw          D1mw       DLmw
--------------------------------------------------------------------------------
10,586,781,019 2,863 2,720 2,507,061,286 13,523,845 2,138,487 184,511,951 5,801,884 2,359,
PROGRAM TOTALS

--------------------------------------------------------------------------------
Ir            I1mr ILmr Dr            D1mr       DLmr      Dw          D1mw      DLmw
file:function
--------------------------------------------------------------------------------
8,731,546,208    6    4 2,032,362,722 10,590,038   561,264 75,270,876 3,530,006 1,176,656
/home/enginsena00/sobel_par.c:thread_convolve
  979,707,289   24   24   263,756,577  1,177,476 1,119,082 37,690,589   590,207   588,734
/home/enginsena00/sobel_par.c:main
  819,843,442  736  700   193,073,289  1,689,292   454,815 52,704,384 1,455,158
591,425  ???:???
   29,953,670    5    5     4,081,292     25,200        49  2,489,006       689
0  ???:jpeg_fill_bit_buffer
   13,305,738   13   13    12,802,254     21,826         4 12,764,099   221,156     1,149
```

For 16 thread:

```
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         49152 B, 64 B, 12-way associative
LL cache:         58720256 B, 64 B, 14-way associative
Command:          ./sobel_par input1.jpg 16
Data file:        cachegrind_out_16
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
Ir             I1mr  ILmr  Dr            D1mr        DLmr       Dw          D1mw       DLmw
--------------------------------------------------------------------------------
10,586,810,758 2,865 2,721 2,507,069,406 13,545,453 2,046,857 184,516,909 5,811,704 2,359,56
PROGRAM TOTALS

--------------------------------------------------------------------------------
Ir            I1mr ILmr Dr            D1mr       DLmr      Dw          D1mw      DLmw
file:function
--------------------------------------------------------------------------------
8,731,547,008    6    4 2,032,362,994 10,590,215   505,767 75,271,020 3,530,056 1,176,658
/home/enginsena00/sobel_par.c:thread_convolve
  979,707,785   24   24   263,756,657  1,177,480 1,082,824 37,690,733   590,216   588,734
/home/enginsena00/sobel_par.c:main
  819,844,530  737  701   193,073,833  1,713,721   454,817 52,704,384 1,464,650
591,422  ???:???
   29,953,670    5    5     4,081,292     25,200        49  2,489,006       689
0  ???:jpeg_fill_bit_buffer
   13,305,738   13   13    12,802,254     18,581         4 12,764,099   220,961     1,148
```

For 32 thread:

```
--------------------------------------------------------------------------------
I1 cache:         32768 B, 64 B, 8-way associative
D1 cache:         49152 B, 64 B, 12-way associative
LL cache:         58720256 B, 64 B, 14-way associative
Command:          ./sobel_par input1.jpg 32
Data file:        cachegrind_out_32
Events recorded:  Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Events shown:     Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Event sort order: Ir I1mr ILmr Dr D1mr DLmr Dw D1mw DLmw
Thresholds:       0.1 100 100 100 100 100 100 100 100
Include dirs:
User annotated:
Auto-annotation:  off

--------------------------------------------------------------------------------
Ir             I1mr  ILmr  Dr            D1mr        DLmr       Dw          D1mw       DLmw
--------------------------------------------------------------------------------
10,586,871,665 2,867 2,721 2,507,085,771 13,531,046 2,183,985 184,526,888 5,819,220 2,359,946  PROGRAM TOTALS

--------------------------------------------------------------------------------
Ir            I1mr ILmr Dr            D1mr       DLmr      Dw          D1mw      DLmw       file:function
--------------------------------------------------------------------------------
8,731,548,608    6    4 2,032,363,538 10,590,539   585,365 75,271,308 3,530,151 1,176,659  /home/enginsena00/sobel_par.c:thread_convolve
  979,708,745   24   24   263,756,817  1,177,489 1,140,480 37,691,021   590,239   588,741  /home/enginsena00/sobel_par.c:main
  819,846,706  738  701   193,074,921  1,690,687   454,755 52,704,384 1,469,498   591,424  ???:???
   29,953,670    5    5     4,081,292     25,200        49  2,489,006       689         0  ???:jpeg_fill_bit_buffer
   13,306,235   15   15    12,802,254     25,693         4 12,764,099   222,634     1,146  /build/glibc-LcI20x/glibc-2.31/string/../sysdeps/x86_64/multiarch/memmove-vec-unaligned-erms.S:
__memcpy_avx_unaligned_erms
```

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.