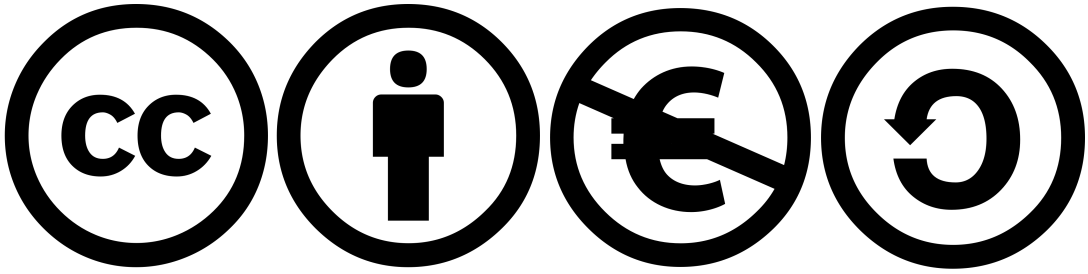


Report Contest OMP

Merge Sort

Lecturer: Francesco Moscato - fmoscato@unisa.it

Group:

- De Stefano Alessandro 0622701470 a.destefano56@studenti.unisa.it
- Della Rocca Marco 0622701573 m.dellarocca22@studenti.unisa.it

Problem Description

Parallelize and Evaluate Performances of "Merge Sort" Algorithm, by using OpenMP.

Experimental Setup

Hardware

CPU Info

```
processor           : 0
vendor_id           : GenuineIntel
cpu family          : 6
model               : 165
model name          : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz
stepping            : 5
microcode           : 0xffffffff
cpu MHz             : 3696.012
cache size          : 20480 KB
physical id         : 0
siblings            : 20
core id             : 0
cpu cores           : 10
apicid              : 0
initial apicid      : 0
fpu                 : yes
fpu_exception       : yes
cpuid level         : 22
wp                  : yes
flags               : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca
                      cmov pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx
                      pdpe1gb rdtscp lm constant_tsc rep_good nopl xtopology cpuid
                      pni pclmulqdq vmx ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic
                      movbe popcnt aes xsave avx f16c rdrand hypervisor lahf_lm
                      abm 3dnowprefetch invpcid_single ssbd ibrs ibpb stibp
                      ibrs_enhanced tpr_shadow vnmi ept vpid ept_ad fsgsbase bmi1
                      avx2 smep bmi2 erms invpcid rdseed adx smap clflushopt
                      xsaveopt xsavec xgetbv1 xsaves flush_l1d arch_capabilities
vmx flags           : vnmi invvpid ept_x_only ept_ad ept_1gb tsc_offset vtpr ept
                      vpid unrestricted_guest ept_mode_based_exec
bugs                : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit
bogomips            : 7392.02
clflush size        : 64
cache_alignment     : 64
address sizes       : 39 bits physical, 48 bits virtual
```

MEM Info

```
MemTotal           :    16317284 kB
MemFree            :    15781724 kB
MemAvailable       :    15748252 kB
Buffers            :      138776 kB
Cached             :       64368 kB
SwapCached         :           0 kB
Active             :      154828 kB
Inactive           :       72376 kB
Active(anon)       :         168 kB
Inactive(anon)     :       24552 kB
Active(file)       :      154660 kB
Inactive(file)     :       47824 kB
Unevictable        :           0 kB
Mlocked            :           0 kB
SwapTotal          :     4194304 kB
SwapFree           :     4194304 kB
Dirty              :         280 kB
Writeback          :           0 kB
AnonPages          :       24464 kB
Mapped             :       22092 kB
Shmem              :         336 kB
KReclaimable       :       25828 kB
Slab               :       64128 kB
SReclaimable       :       25828 kB
SUnreclaim         :       38300 kB
KernelStack        :        4384 kB
PageTables         :        1340 kB
NFS_Unstable       :           0 kB
Bounce             :           0 kB
WritebackTmp       :           0 kB
CommitLimit        :    12352944 kB
Committed_AS       :      308872 kB
VmallocTotal       : 34359738367 kB
VmallocUsed        :       25100 kB
VmallocChunk       :           0 kB
Percpu             :        7968 kB
AnonHugePages      :        8192 kB
ShmemHugePages     :           0 kB
ShmemPmdMapped     :           0 kB
FileHugePages      :           0 kB
FilePmdMapped      :           0 kB
HugePages_Total    :           0
HugePages_Free     :           0
HugePages_Rsvd     :           0
HugePages_Surp     :           0
Hugepagesize       :        2048 kB
Hugetlb            :           0 kB
DirectMap4k        :       19456 kB
DirectMap2M        :     4104192 kB
DirectMap1G        :    22020096 kB
```

Software

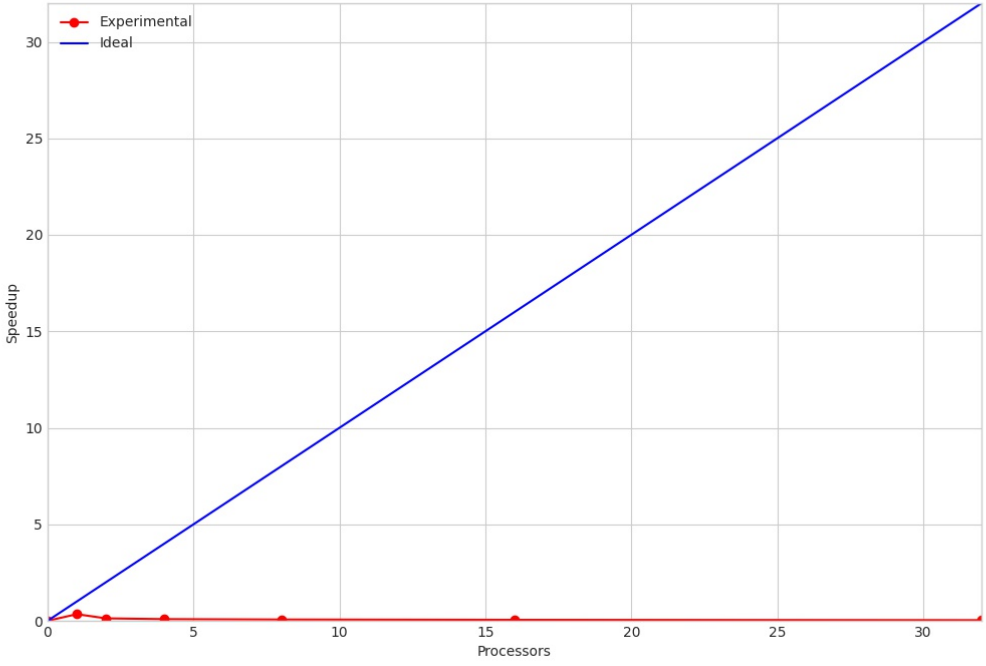
gcc version 9.3.0
OS: Ubuntu 20.04.3 LTS on Windows 10 x86_64
Kernel: 5.10.60.1-microsoft-standard-WSL2
Shell: bash 5.0.17
CPU: Intel i9-10900K (20) @ 3.696GHz
Memory: 921MiB / 15934MiB
Swap: 4194304 KiB

Performance, Speedup & Efficiency

We analyzed the performance of our program taking as a reference the execution of the sequential algorithm.
We evaluated the execution time by changing the size of the array to sort [1e5, 4e5, 16e5], the gcc optimization level [O0, O1, O2, O3], and the number of threads [1, 2, 4, 8, 16, 32].

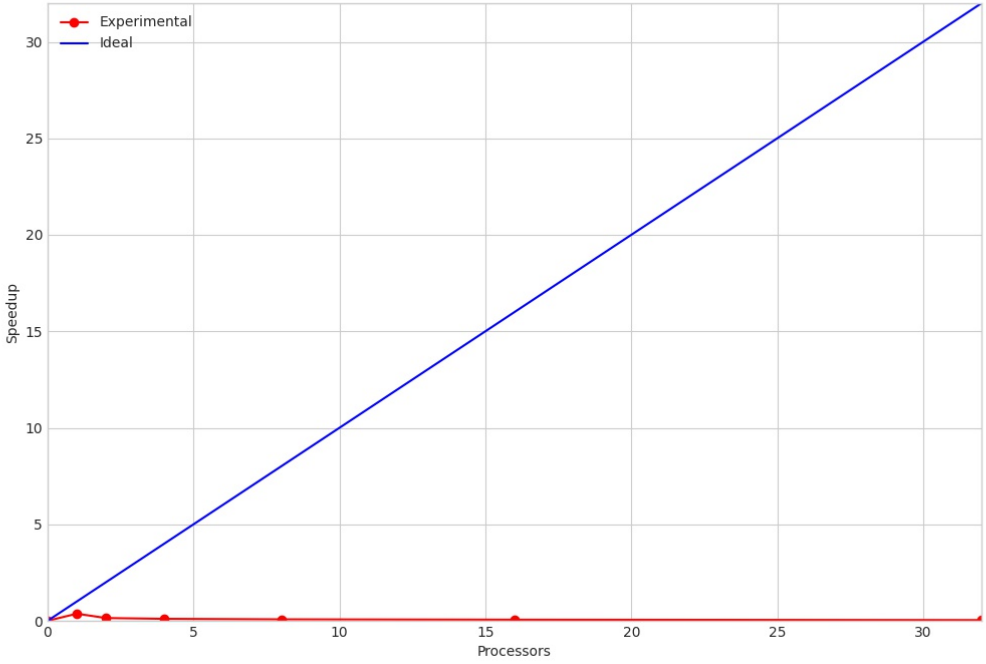
size 1e5, O2

execution	threads	time	stdev	speedup	efficiency
serial	1	0.005900	0.000000	1.000000	1.000000
parallel	1	0.017200	0.000100	0.343000	0.343000
parallel	2	0.047000	0.005000	0.125500	0.062800
parallel	4	0.068100	0.004700	0.086600	0.021700
parallel	8	0.088100	0.007500	0.067000	0.008400
parallel	16	0.117400	0.007000	0.050300	0.003100
parallel	32	0.172800	0.006400	0.034100	0.001100



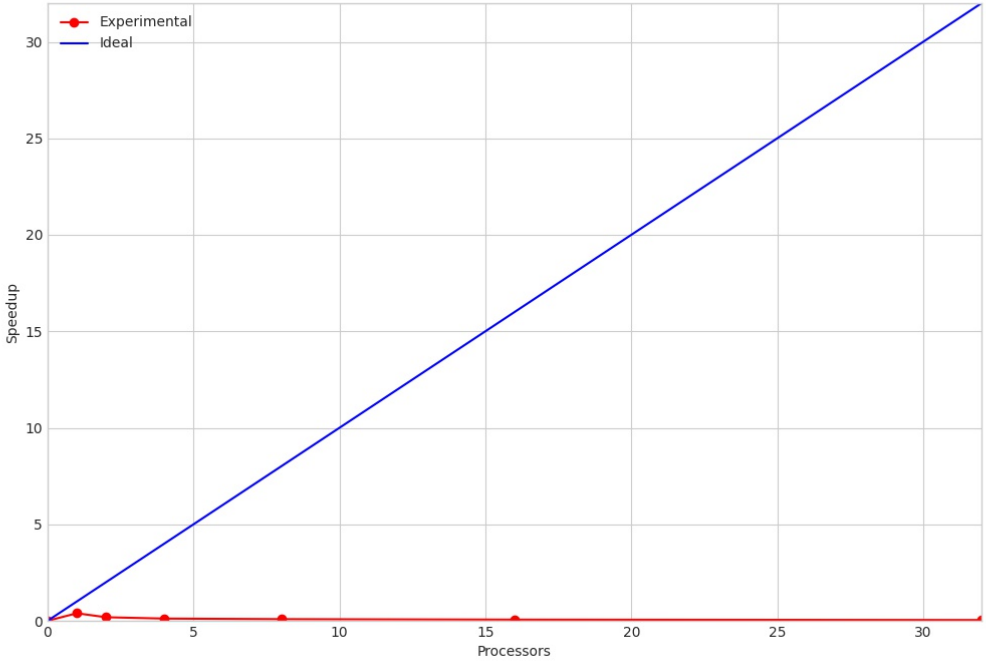
size 4e5, O2

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026300	0.000100	1.000000	1.000000
parallel	1	0.071500	0.000300	0.367800	0.367800
parallel	2	0.183900	0.017700	0.143000	0.071500
parallel	4	0.259600	0.019700	0.101300	0.025300
parallel	8	0.345000	0.015400	0.076200	0.009500
parallel	16	0.454900	0.032400	0.057800	0.003600
parallel	32	0.692600	0.031500	0.038000	0.001200



size 16e5, O2

execution	threads	time	stdev	speedup	efficiency
serial	1	0.113900	0.000800	1.000000	1.000000
parallel	1	0.292000	0.000900	0.390100	0.390100
parallel	2	0.618300	0.070600	0.184200	0.092100
parallel	4	0.996200	0.090500	0.114300	0.028600
parallel	8	1.327700	0.067700	0.085800	0.010700
parallel	16	1.862600	0.142000	0.061200	0.003800
parallel	32	2.770100	0.144100	0.041100	0.001300

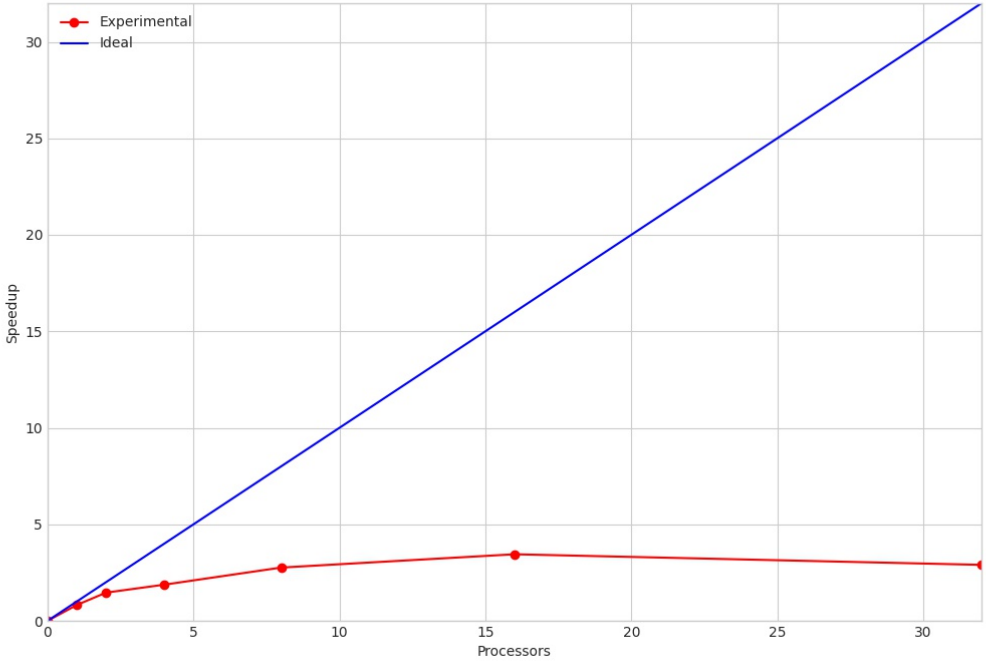


Considerations

There are no improvements whatsoever. Instead, the performances in the parallel version are much worse.
The omp library is introducing some sort of overhead that completely ruins performances. After some focused brainstorming we found out that every thread execution was working too little. To counter that, we introduced a value called “task_size” which we are willing to optimize during the next tests. By doing so the execution will become sequential when the array’s size is less than task_size.

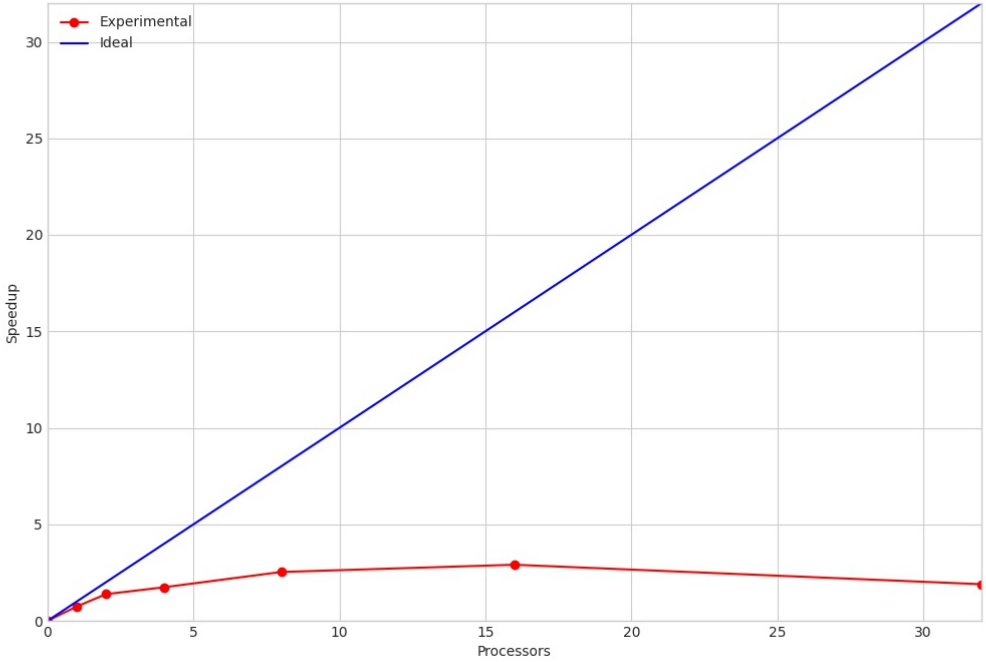
size 4e5, Task_size 100, O0

execution	threads	time	stdev	speedup	efficiency
serial	1	0.045500	0.000200	1.000000	1.000000
parallel	1	0.055600	0.000200	0.818300	0.818300
parallel	2	0.031200	0.003300	1.458300	0.729200
parallel	4	0.024300	0.000900	1.872400	0.468100
parallel	8	0.016500	0.000700	2.757600	0.344700
parallel	16	0.013200	0.000300	3.447000	0.215400
parallel	32	0.015700	0.000400	2.898100	0.090600



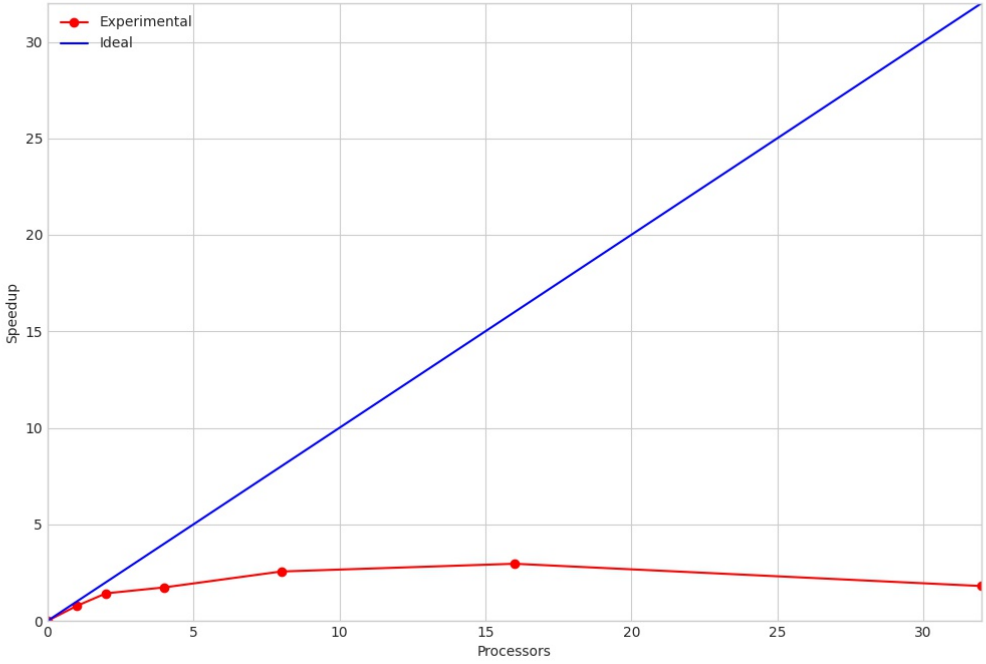
size 4e5, Task_size 100, O1

execution	threads	time	stdev	speedup	efficiency
serial	1	0.027300	0.000200	1.000000	1.000000
parallel	1	0.036800	0.000300	0.741800	0.741800
parallel	2	0.019800	0.000400	1.378800	0.689400
parallel	4	0.015700	0.000100	1.738900	0.434700
parallel	8	0.010800	0.001000	2.527800	0.316000
parallel	16	0.009400	0.000200	2.904300	0.181500
parallel	32	0.014400	0.000300	1.895800	0.059200



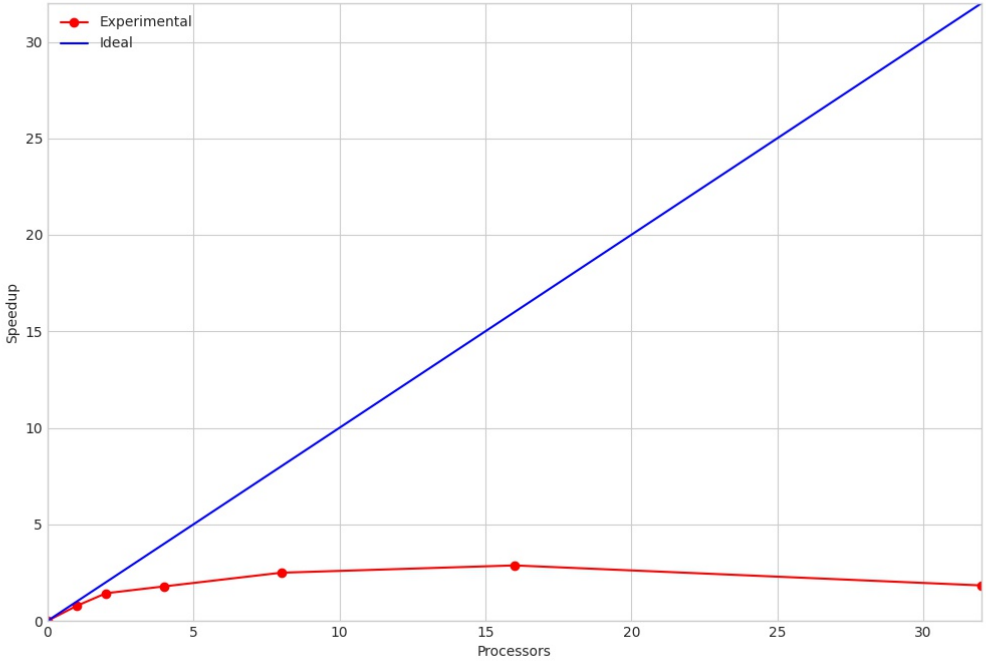
size 4e5, Task_size 100, O2

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026300	0.000200	1.000000	1.000000
parallel	1	0.034000	0.000100	0.773500	0.773500
parallel	2	0.018500	0.000100	1.421600	0.710800
parallel	4	0.015200	0.000400	1.730300	0.432600
parallel	8	0.010300	0.000700	2.553400	0.319200
parallel	16	0.008900	0.000300	2.955100	0.184700
parallel	32	0.014600	0.000700	1.801400	0.056300



size 4e5, Task_size 100, O3

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026400	0.001500	1.000000	1.000000
parallel	1	0.034000	0.000200	0.776500	0.776500
parallel	2	0.018500	0.000400	1.427000	0.713500
parallel	4	0.014800	0.000600	1.783800	0.445900
parallel	8	0.010600	0.000500	2.490600	0.311300
parallel	16	0.009200	0.000300	2.869600	0.179300
parallel	32	0.014400	0.000500	1.833300	0.057300

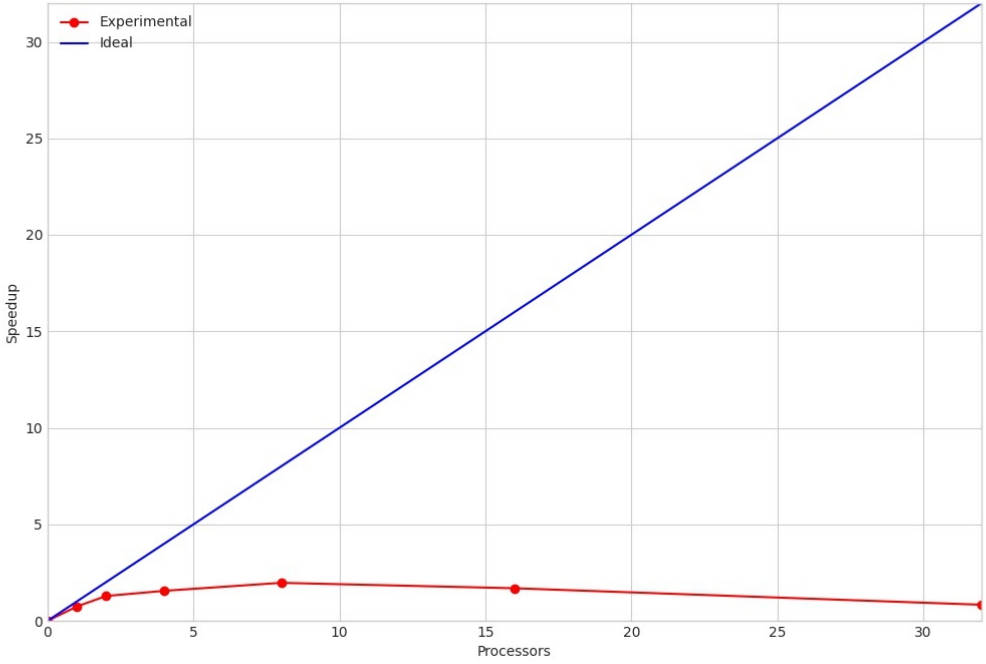


The effects of the task_size

Independently by the level of optimization, the introduction of the task_size is doing its job.
The best performances are obtained with the gcc optimization level O2.
Let us find the best task_size with respect to the problem size.

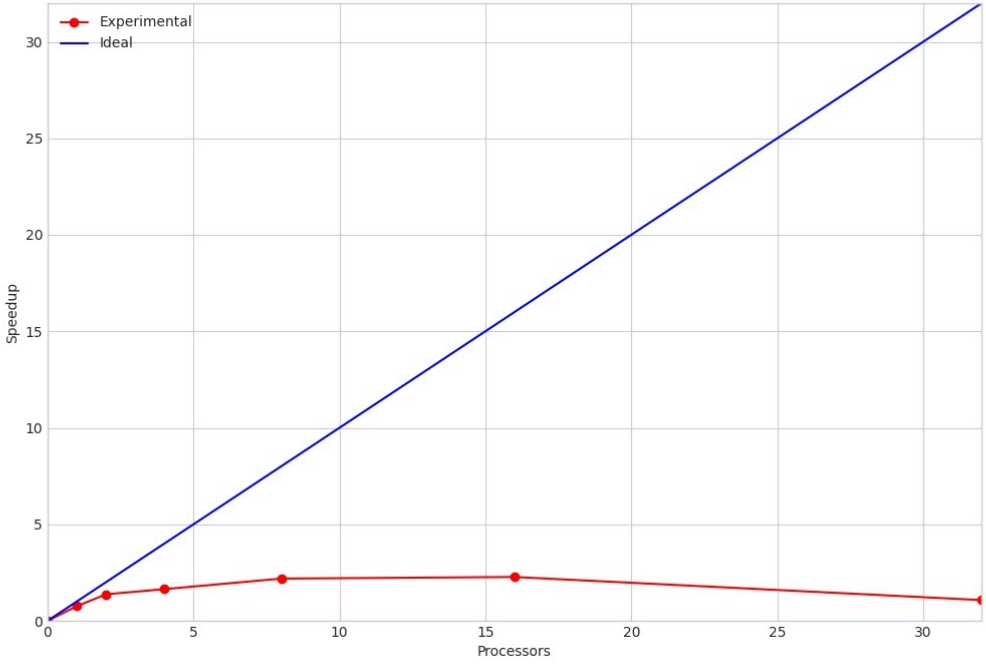
size 1e5, O2, task_size 50

execution	threads	time	stdev	speedup	efficiency
serial	1	0.005900	0.000000	1.000000	1.000000
parallel	1	0.008000	0.000000	0.737500	0.737500
parallel	2	0.004600	0.000000	1.282600	0.641300
parallel	4	0.003800	0.000100	1.552600	0.388200
parallel	8	0.003000	0.000100	1.966700	0.245800
parallel	16	0.003500	0.000200	1.685700	0.105400
parallel	32	0.007100	0.000300	0.831000	0.026000



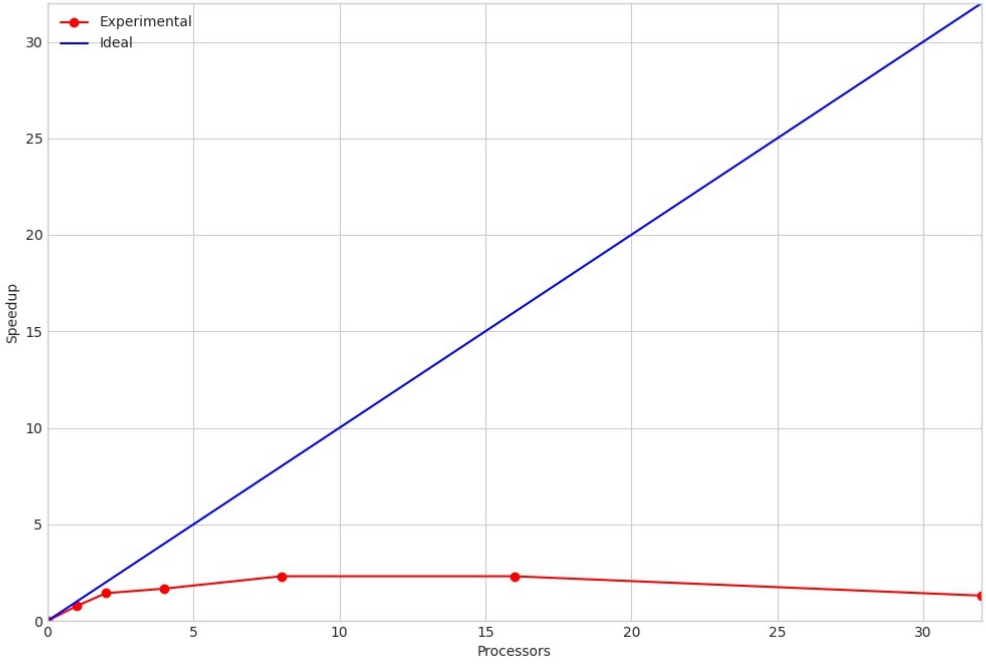
size 1e5, O2, task_size 100

execution	threads	time	stdev	speedup	efficiency
serial	1	0.005900	0.000100	1.000000	1.000000
parallel	1	0.007800	0.000000	0.756400	0.756400
parallel	2	0.004300	0.000000	1.372100	0.686000
parallel	4	0.003600	0.000100	1.638900	0.409700
parallel	8	0.002700	0.000200	2.185200	0.273100
parallel	16	0.002600	0.000000	2.269200	0.141800
parallel	32	0.005500	0.000400	1.072700	0.033500



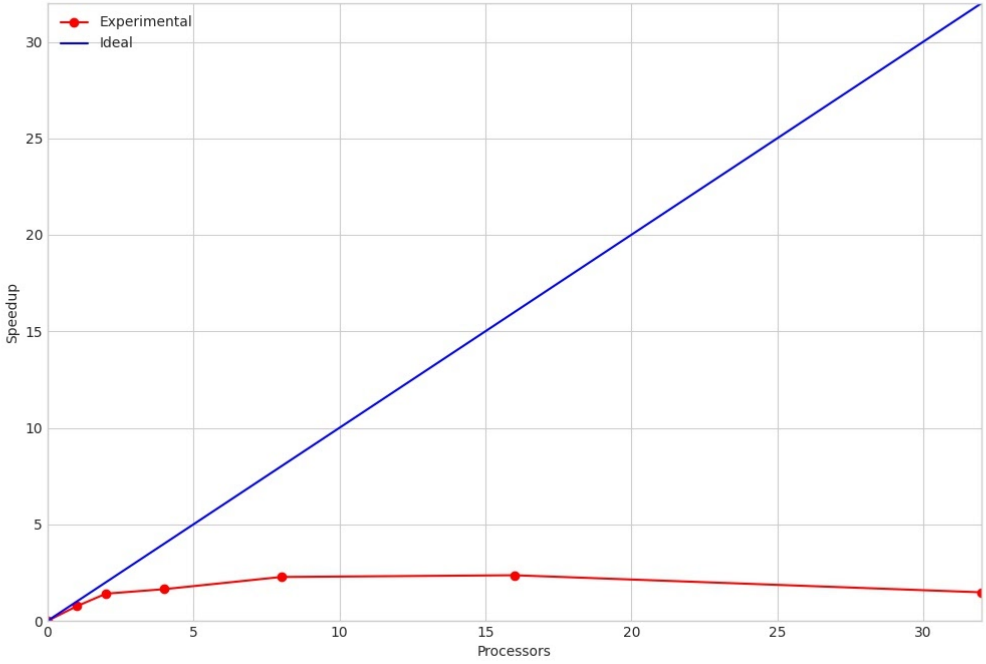
size 1e5, O2, task_size 200

execution	threads	time	stdev	speedup	efficiency
serial	1	0.006000	0.000100	1.000000	1.000000
parallel	1	0.007800	0.000100	0.769200	0.769200
parallel	2	0.004200	0.000100	1.428600	0.714300
parallel	4	0.003600	0.000100	1.666700	0.416700
parallel	8	0.002600	0.000100	2.307700	0.288500
parallel	16	0.002600	0.000300	2.307700	0.144200
parallel	32	0.004600	0.000200	1.304300	0.040800



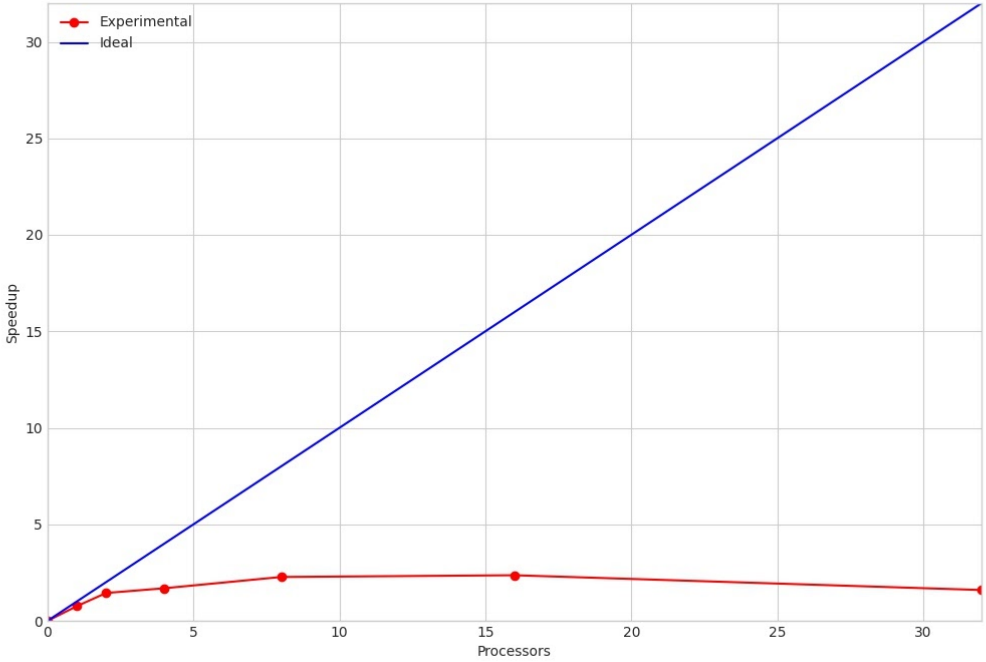
size 1e5, O2, task_size 400

execution	threads	time	stdev	speedup	efficiency
serial	1	0.005900	0.000000	1.000000	1.000000
parallel	1	0.007800	0.000000	0.756400	0.756400
parallel	2	0.004200	0.000000	1.404800	0.702400
parallel	4	0.003600	0.000100	1.638900	0.409700
parallel	8	0.002600	0.000100	2.269200	0.283700
parallel	16	0.002500	0.000000	2.360000	0.147500
parallel	32	0.004000	0.000600	1.475000	0.046100



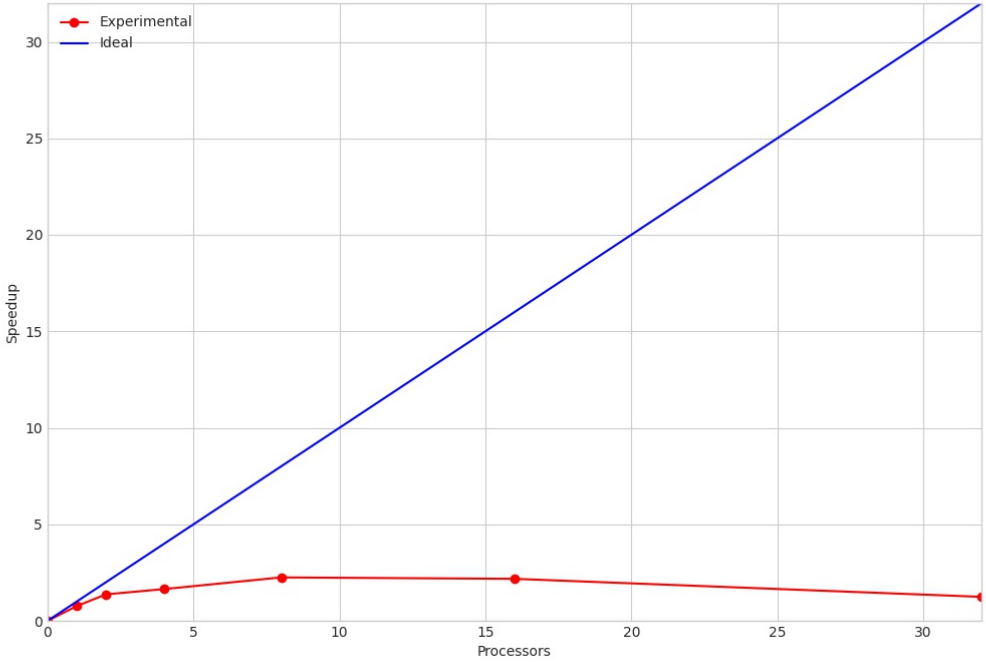
size 1e5, O2, task_size 800

execution	threads	time	stdev	speedup	efficiency
serial	1	0.005900	0.000000	1.000000	1.000000
parallel	1	0.007800	0.000000	0.756400	0.756400
parallel	2	0.004100	0.000000	1.439000	0.719500
parallel	4	0.003500	0.000100	1.685700	0.421400
parallel	8	0.002600	0.000100	2.269200	0.283700
parallel	16	0.002500	0.000100	2.360000	0.147500
parallel	32	0.003700	0.000200	1.594600	0.049800



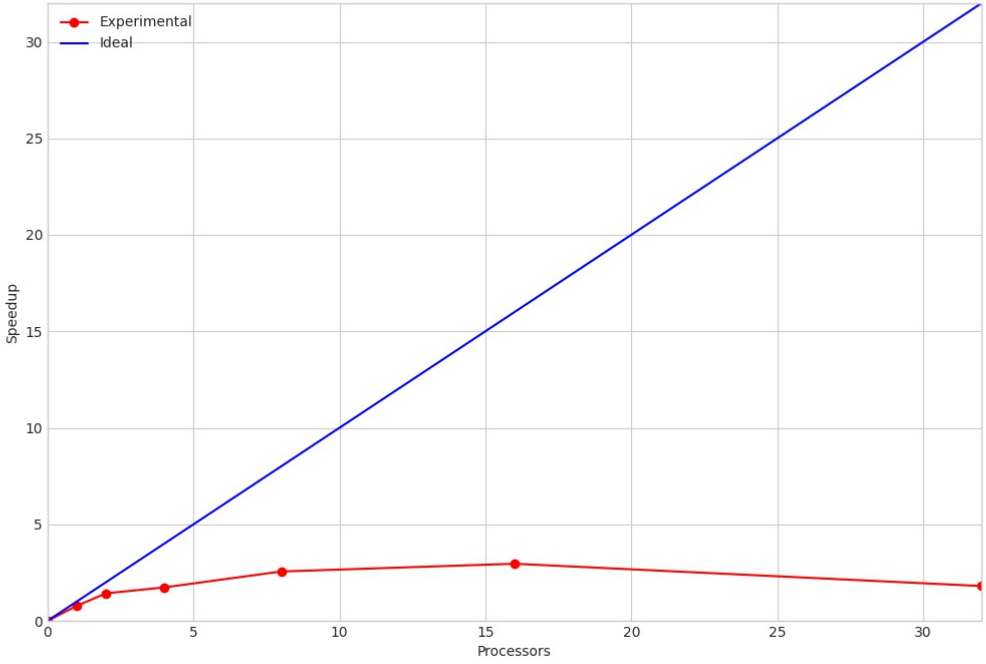
size 4e5, O2, task_size 50

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026300	0.000200	1.000000	1.000000
parallel	1	0.034500	0.000200	0.762300	0.762300
parallel	2	0.019200	0.000100	1.369800	0.684900
parallel	4	0.016000	0.000500	1.643800	0.410900
parallel	8	0.011700	0.000300	2.247900	0.281000
parallel	16	0.012100	0.000600	2.173600	0.135800
parallel	32	0.021200	0.000600	1.240600	0.038800



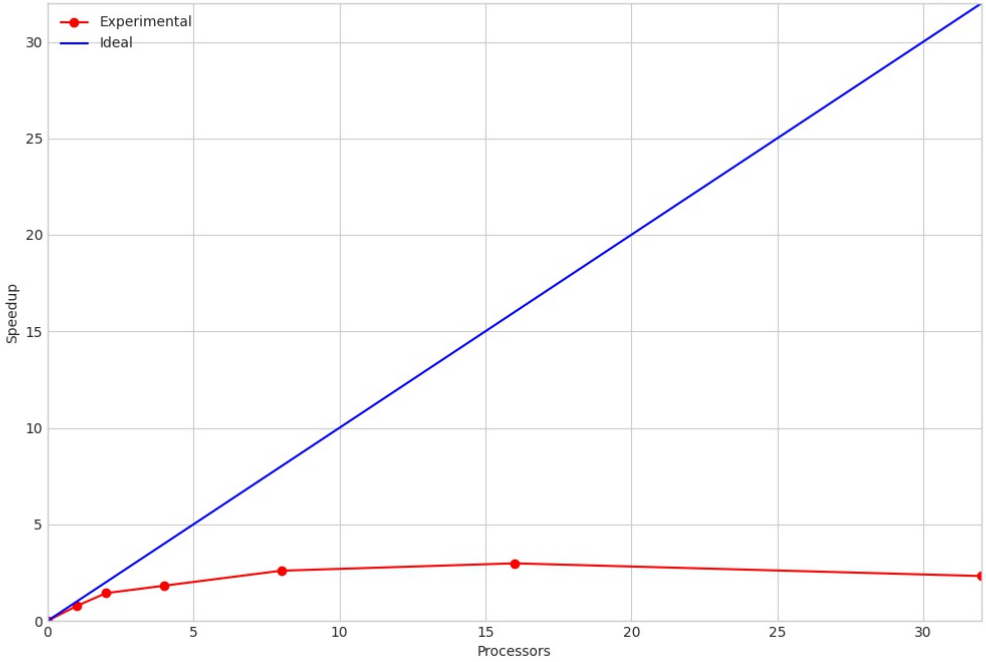
size 4e5, O2, task_size 100

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026300	0.000200	1.000000	1.000000
parallel	1	0.034000	0.000100	0.773500	0.773500
parallel	2	0.018500	0.000100	1.421600	0.710800
parallel	4	0.015200	0.000400	1.730300	0.432600
parallel	8	0.010300	0.000700	2.553400	0.319200
parallel	16	0.008900	0.000300	2.955100	0.184700
parallel	32	0.014600	0.000700	1.801400	0.056300



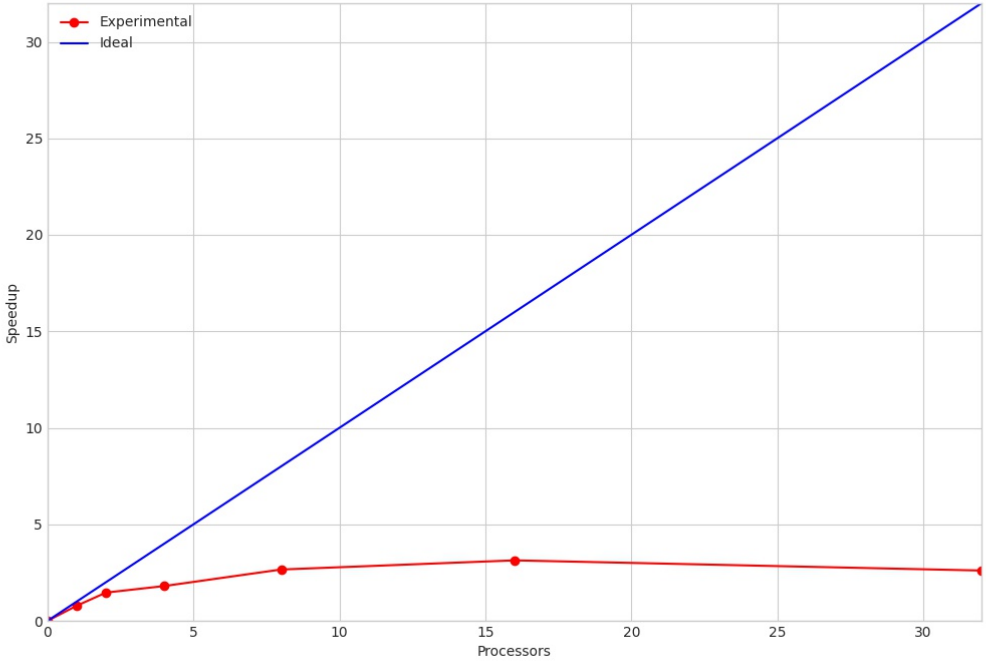
size 4e5, O2, task_size 200

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026200	0.000100	1.000000	1.000000
parallel	1	0.034000	0.000300	0.770600	0.770600
parallel	2	0.018300	0.000800	1.431700	0.715800
parallel	4	0.014400	0.000300	1.819400	0.454900
parallel	8	0.010100	0.000500	2.594100	0.324300
parallel	16	0.008800	0.000400	2.977300	0.186100
parallel	32	0.011300	0.000600	2.318600	0.072500



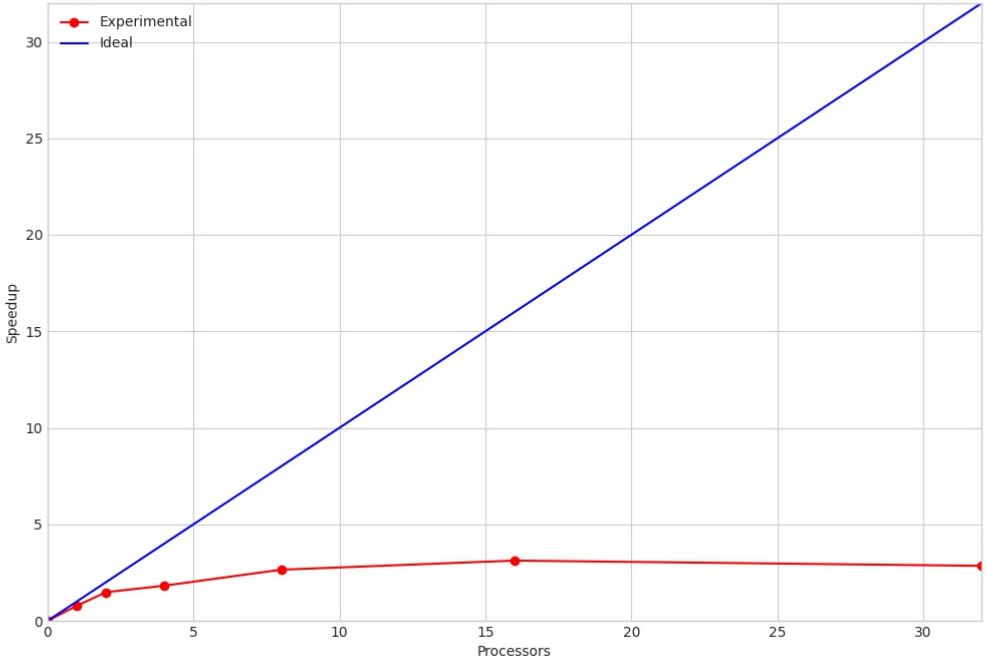
size 4e5, O2, task_size 400

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026300	0.000100	1.000000	1.000000
parallel	1	0.033600	0.000100	0.782700	0.782700
parallel	2	0.018000	0.000200	1.461100	0.730600
parallel	4	0.014600	0.001500	1.801400	0.450300
parallel	8	0.009900	0.000600	2.656600	0.332100
parallel	16	0.008400	0.000200	3.131000	0.195700
parallel	32	0.010100	0.000900	2.604000	0.081400



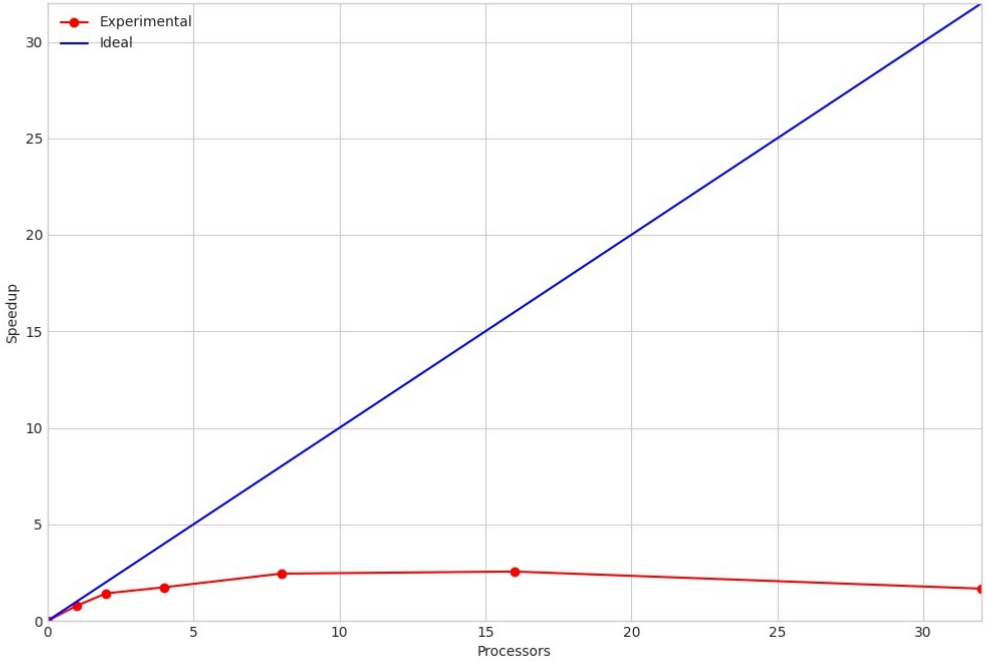
size 4e5, O2, task_size 800

execution	threads	time	stdev	speedup	efficiency
serial	1	0.026200	0.000100	1.000000	1.000000
parallel	1	0.033600	0.000300	0.779800	0.779800
parallel	2	0.017700	0.000200	1.480200	0.740100
parallel	4	0.014400	0.000500	1.819400	0.454900
parallel	8	0.009900	0.000300	2.646500	0.330800
parallel	16	0.008400	0.000400	3.119000	0.194900
parallel	32	0.009200	0.000400	2.847800	0.089000



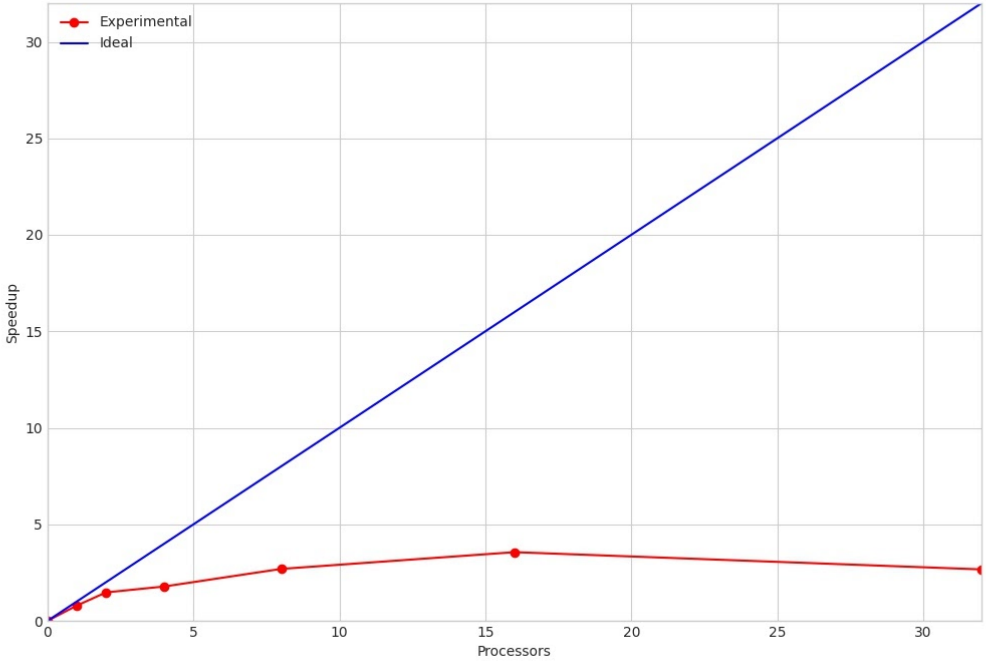
size 16e5, O2, task_size 50

execution	threads	time	stdev	speedup	efficiency
serial	1	0.114300	0.000900	1.000000	1.000000
parallel	1	0.146100	0.001200	0.782300	0.782300
parallel	2	0.080600	0.000800	1.418100	0.709100
parallel	4	0.065800	0.004900	1.737100	0.434300
parallel	8	0.046800	0.001800	2.442300	0.305300
parallel	16	0.044800	0.002700	2.551300	0.159500
parallel	32	0.068600	0.002700	1.666200	0.052100



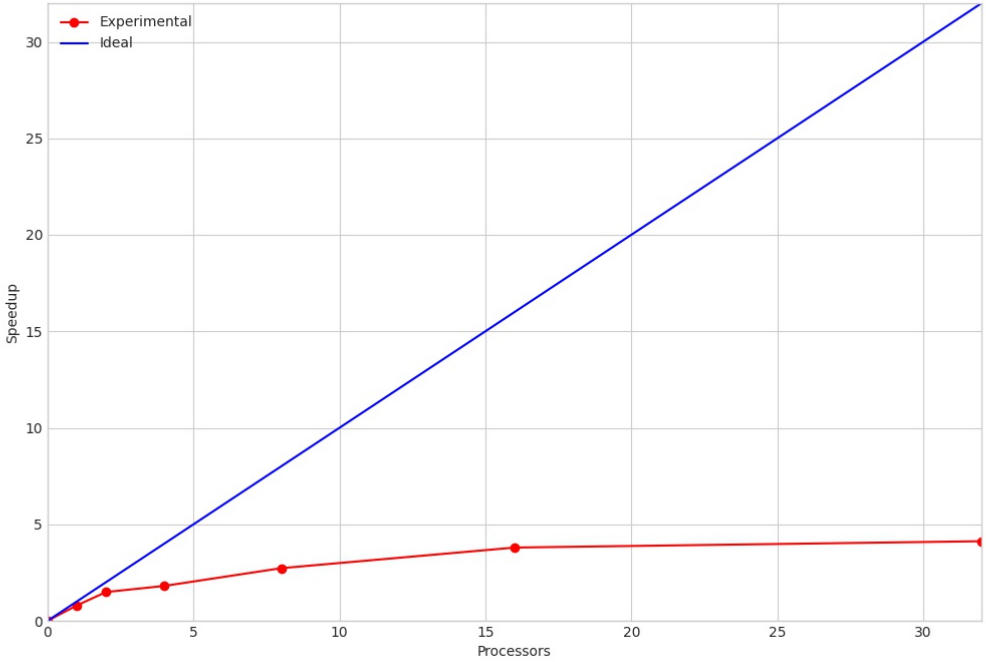
size 16e5, O2, task_size 100

execution	threads	time	stdev	speedup	efficiency
serial	1	0.114500	0.000700	1.000000	1.000000
parallel	1	0.145200	0.002400	0.788600	0.788600
parallel	2	0.077900	0.000500	1.469800	0.734900
parallel	4	0.064400	0.004500	1.778000	0.444500
parallel	8	0.042600	0.002000	2.687800	0.336000
parallel	16	0.032200	0.001400	3.555900	0.222200
parallel	32	0.043000	0.001800	2.662800	0.083200



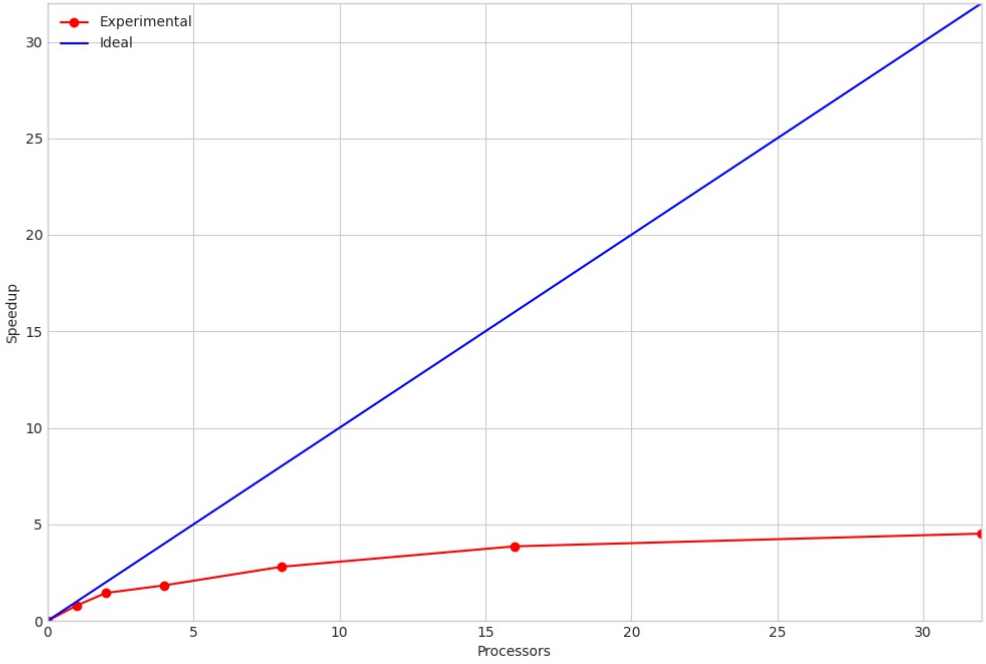
size 16e5, O2, task_size 200

execution	threads	time	stdev	speedup	efficiency
serial	1	0.114200	0.000400	1.000000	1.000000
parallel	1	0.143600	0.000900	0.795300	0.795300
parallel	2	0.076700	0.000800	1.488900	0.744500
parallel	4	0.063200	0.004100	1.807000	0.451700
parallel	8	0.041900	0.002600	2.725500	0.340700
parallel	16	0.030100	0.001300	3.794000	0.237100
parallel	32	0.027700	0.000900	4.122700	0.128800



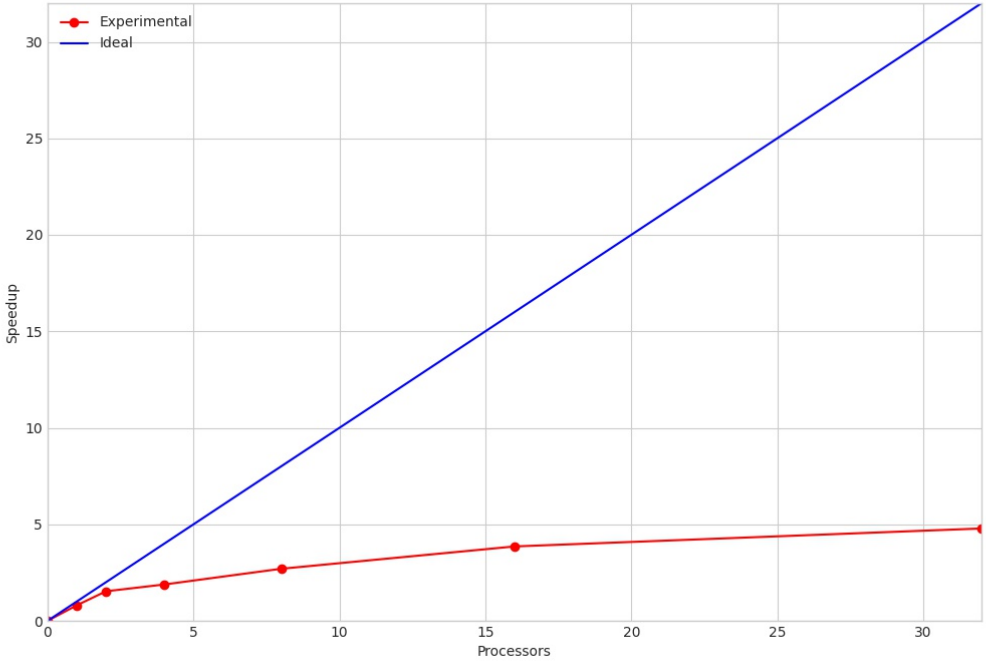
size 16e5, O2, task_size 400

execution	threads	time	stdev	speedup	efficiency
serial	1	0.113800	0.000700	1.000000	1.000000
parallel	1	0.142900	0.001400	0.796400	0.796400
parallel	2	0.078900	0.007700	1.442300	0.721200
parallel	4	0.062000	0.005000	1.835500	0.458900
parallel	8	0.040700	0.002500	2.796100	0.349500
parallel	16	0.029500	0.001100	3.857600	0.241100
parallel	32	0.025200	0.000400	4.515900	0.141100



size 16e5, O2, task_size 800

execution	threads	time	stdev	speedup	efficiency
serial	1	0.114800	0.001700	1.000000	1.000000
parallel	1	0.143000	0.001000	0.802800	0.802800
parallel	2	0.075100	0.001500	1.528600	0.764300
parallel	4	0.061000	0.003800	1.882000	0.470500
parallel	8	0.042600	0.001200	2.694800	0.336900
parallel	16	0.029800	0.001800	3.852300	0.240800
parallel	32	0.024000	0.000700	4.783300	0.149500



Final Considerations

In the end, we can say that, generally, the best setup to improve performance on a merge sort algorithm, given the system specification reported in the “Experimental setup” section, is to use:

- O2 as gcc optimization level
- 400 as task_size
- the maximum number of logical threads available on the machine.

All the measures are available in the tar archive ‘measures.tar.xz’.

Test Cases

In the test folder there is the file merge_sort.c that contains the test cases.

We tested our implementation of the merge sort algorithm many times giving it in input different arrays:

- empty array
- array with only one element
- array whose elements are in reverse order
- array whose elements are all different from each other
- array whose elements are all equal
- array in which some elements are equal
- array whose elements are all positive
- array whose elements are all negative
- array whose elements are already ordered
- array with explicit task_size

We also included a failing test case by passing the wrong size to the algorithm.

API

The documentation has been generated with Doxygen and it is available at the following link https://contestomp.netlify.app.

How To Run

General Instructions

To build the executable both for tests and source code run
```
make
```

To clean the .o files in the build directories run
```
make clean
```

To build and execute test cases run
```
make test
```

To generate measures (TAKES A LOT OF TIME! Our measures are already included so you should skip this step) run
```
make measures
```

To generate a file with random numbers run
```
make generate_file [N=N[:10000]] [MAX=MAX[:2147483647]] [MIN=MIN[:-MAX]]
[FILENAME=path/to/inputfile[:input/in.txt]]
```

To build and execute the source code run
```
[OMP_NUM_THREADS=OMP_NUM_THREADS] make run
[FILENAME=path/to/inputfile[:input/in.txt]] [TASK_SIZE=TASK_SIZE[:100]]
```

To build and execute SILENTLY the source code run
```
[OMP_NUM_THREADS=OMP_NUM_THREADS] make -s run
[FILENAME=path/to/inputfile[:input/in.txt]] [TASK_SIZE=TASK_SIZE[:100]]
```

To generate a file with random numbers and execute a single run of the mergesort run
```
[OMP_NUM_THREADS=OMP_NUM_THREADS] make generate_file run [N=N[:10000]]
[MAX=MAX[:2147483647]] [MIN=MIN[:-MAX]]
[FILENAME=path/to/inputfile[:input/in.txt]] [TASK_SIZE=TASK_SIZE[:100]]
```