# Report Common Assignment 1

Counting sort

Rosa Gerardo

Scovotto Luigi

Tortora Francesco

November, 2021

# Index

# Chapter 1

# Problem description

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

Counting sort is an algorithm for sorting integer numbers. The computational complexity is equal to $O(n)$.

In this report we analize two different implementation of counting sort algorithm.

In the first case study we have chosen to start from the following source:
https://github.com/ianliu/programacao-paralela/blob/master/omp-count-sort/main.c

This solution is fully parallelized but has the disadvantage of having a computational complexity $O(n^2)$

In the second case study we used the original version of the counting cort, so with complexity $O(n)$, but it was not possible to parallel it completely.

## 1.1 Experimental setup

### 1.1.1 Hardware

CPU

```
machdep.cpu.brand_string: Apple M1 Pro
machdep.cpu.core_count: 10
machdep.cpu.cores_per_package: 10
machdep.cpu.logical_per_package: 10
machdep.cpu.thread_count: 10
```

```
This CPU has 10 cores but 8 are for high−performance
and 2 for energetical efficiency
```

RAM

```
kern.ipc.mb_memory_pressure_percentage: 80
kern.dtrace.buffer_memory_inuse: 0
kern.dtrace.buffer_memory_maxsize: 5726623061
kern.memorystatus_apps_idle_delay_time: 10
kern.memorystatus_level: 85
kern.memorystatus_sysprocs_idle_delay_time: 10
kern.memorystatus_purge_on_critical: 8
kern.memorystatus_purge_on_urgent: 5
kern.memorystatus_purge_on_warning: 2
vm.memory_pressure: 0
hw.memsize: 17179869184
hw.optional.ucnormal_mem: 1
audit.session.member_clear_sflags_mask: 16384
audit.session.member_set_sflags_mask: 0
unified memory: 16 GB
type: LPDDR5
```

### 1.1.2 Software

- macOS Monterey Version 12.0.1
- clang version 13.0.0
- The swap is done dynamically, it is 1024MB by default, but as soon as this threshold is reached it is increased to 2048MB and so on.

# Chapter 2

# Performance, Speedup & Efficiency

## 2.1 Case study n°1

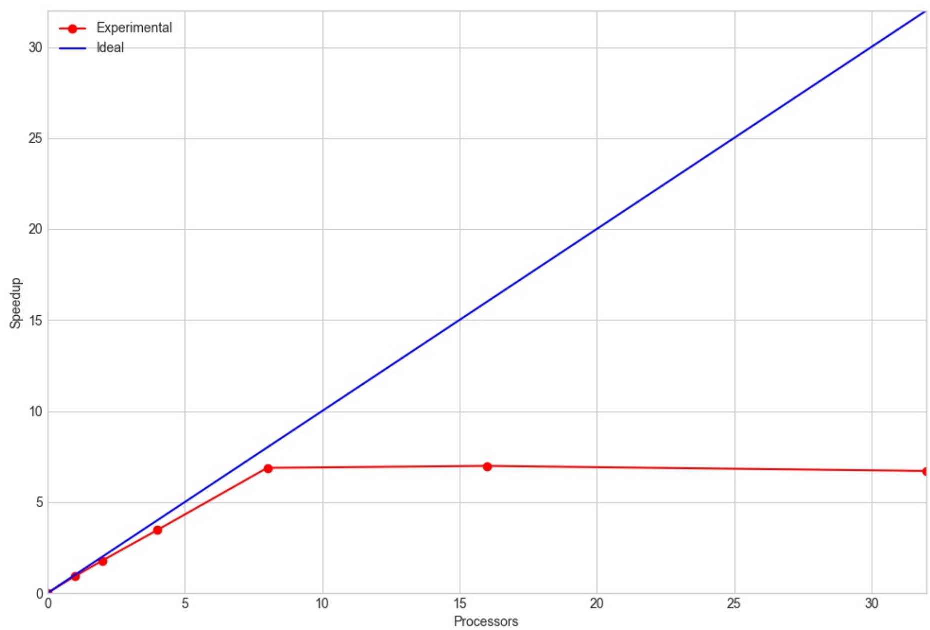
In this case study, the main purpose was to analyze the performance of program with complexity $O(n^2)$ in the following build setup:

- The sequential program are compiled with the gcc optimization -Ox where x = 1,2,3
- The parallel programs are compiled with the gcc optimization -Ox where x = 1,2,3

So here we want to highlight the difference between a sequential program compared to a parallel one, both compiled with the compiler optimizations. Furthermore the case study is done on multiple size that are 50000, 100000, 150000, 200000, with different number of threads (0, 1, 2, 4, 8, 16, 32) on 50 repetitions.

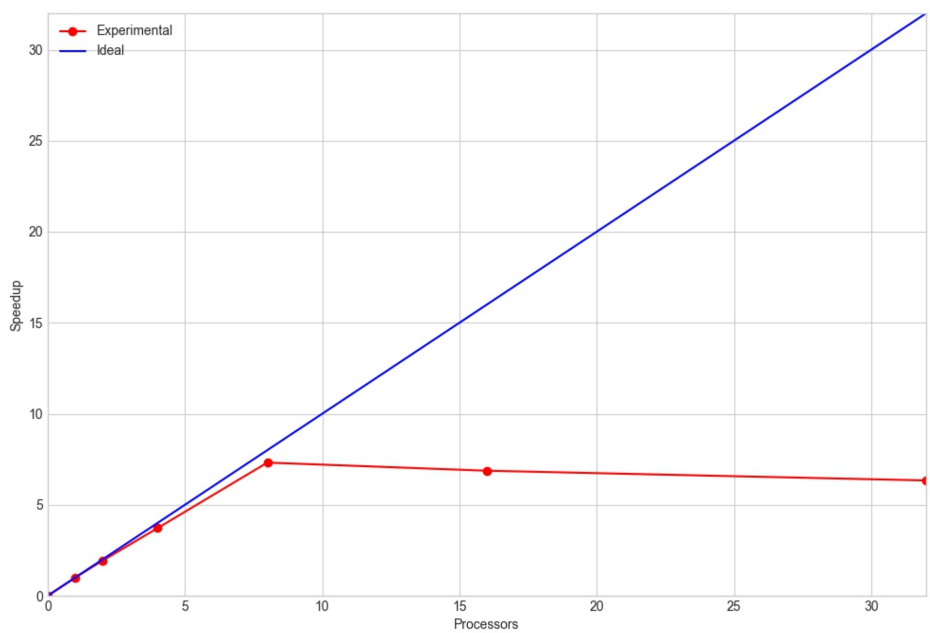
### 2.1.1 Size-50000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,000 43 | 1,579 65 | 1,579 57 | 0,001 00 | 1,581 39 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,000 65 | 1,702 27 | 1,702 52 | 0,002 00 | 1,704 21 | 0,927 93 | 0,927 93 |
| Parallel | 2 | 0,000 70 | 1,764 97 | 1,765 81 | 0,001 00 | 0,884 93 | 1,787 03 | 0,893 51 |
| Parallel | 4 | 0,000 76 | 1,816 04 | 1,816 34 | 0,002 00 | 0,456 59 | 3,463 51 | 0,865 88 |
| Parallel | 8 | 0,001 39 | 1,819 71 | 1,821 18 | 0,002 00 | 0,229 90 | 6,878 66 | 0,859 83 |
| Parallel | 16 | 0,003 68 | 2,171 03 | 2,090 28 | 0,087 15 | 0,226 43 | 6,983 96 | 0,436 50 |
| Parallel | 32 | 0,007 44 | 2,192 56 | 2,123 72 | 0,102 42 | 0,235 77 | 6,707 42 | 0,209 61 |



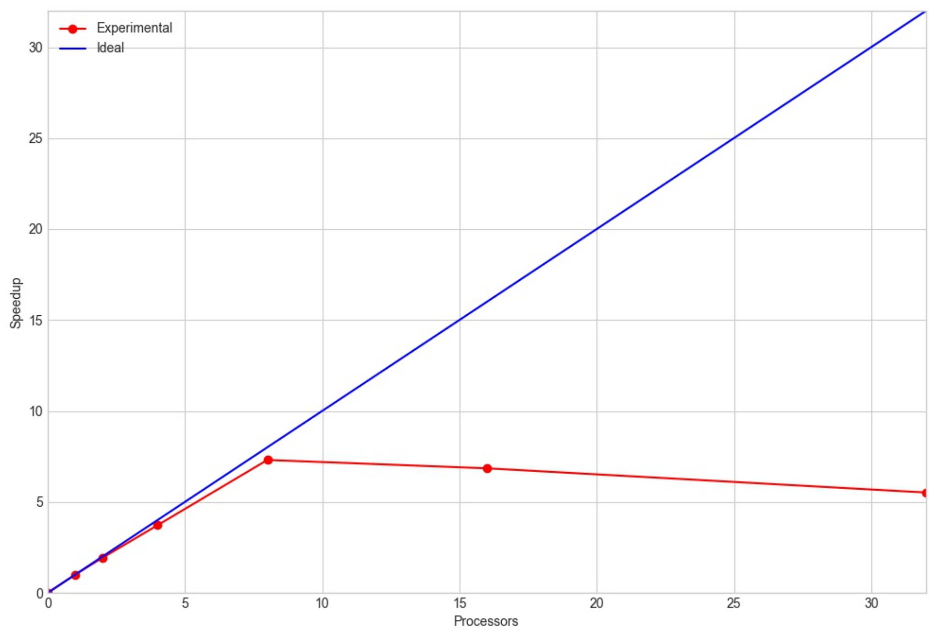
### 2.1.2 Size-50000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,000 43 | 0,533 21 | 0,534 00 | 0,001 00 | 0,535 04 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,000 65 | 0,533 19 | 0,534 00 | 0,001 00 | 0,535 96 | 0,998 29 | 0,998 29 |
| Parallel | 2 | 0,000 70 | 0,549 53 | 0,551 00 | 0,001 00 | 0,277 00 | 1,931 56 | 0,965 78 |
| Parallel | 4 | 0,000 75 | 0,566 93 | 0,568 00 | 0,001 00 | 0,144 00 | 3,715 56 | 0,928 89 |
| Parallel | 8 | 0,001 36 | 0,568 76 | 0,570 55 | 0,001 45 | 0,073 14 | 7,315 66 | 0,914 46 |
| Parallel | 16 | 0,003 65 | 0,705 97 | 0,655 50 | 0,057 81 | 0,077 88 | 6,870 17 | 0,429 39 |
| Parallel | 32 | 0,021 08 | 0,688 75 | 0,679 98 | 0,041 34 | 0,084 45 | 6,335 25 | 0,197 98 |



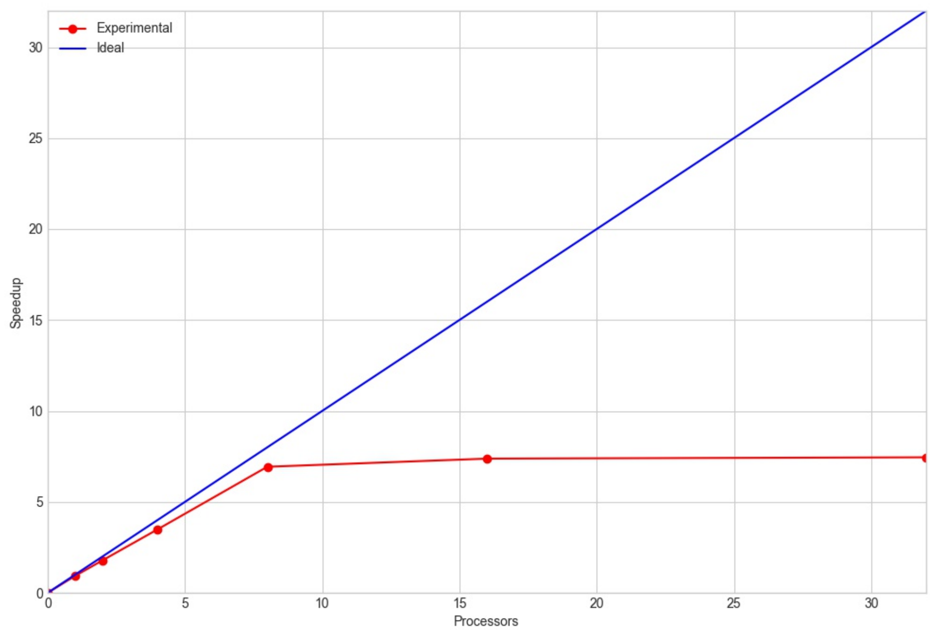
### 2.1.3 Size-50000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,000 43 | 0,533 19 | 0,534 00 | 0,001 00 | 0,535 02 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,000 64 | 0,533 12 | 0,534 00 | 0,001 00 | 0,535 96 | 0,998 25 | 0,998 25 |
| Parallel | 2 | 0,000 70 | 0,549 52 | 0,551 00 | 0,001 00 | 0,277 00 | 1,931 48 | 0,965 74 |
| Parallel | 4 | 0,000 74 | 0,566 97 | 0,568 04 | 0,001 00 | 0,144 00 | 3,715 42 | 0,928 85 |
| Parallel | 8 | 0,001 31 | 0,569 44 | 0,571 30 | 0,001 35 | 0,073 26 | 7,303 50 | 0,912 94 |
| Parallel | 16 | 0,004 18 | 0,707 84 | 0,656 44 | 0,057 21 | 0,078 15 | 6,845 88 | 0,427 87 |
| Parallel | 32 | 0,041 46 | 0,726 13 | 0,729 53 | 0,061 09 | 0,097 03 | 5,514 10 | 0,172 32 |



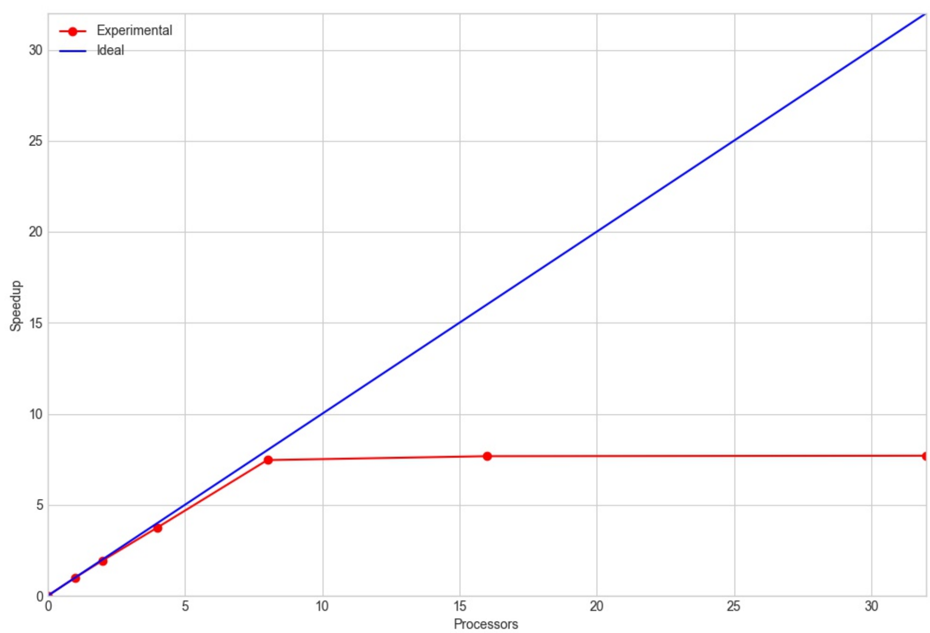
### 2.1.4 Size-100000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,000 86 | 6,360 18 | 6,355 89 | 0,005 50 | 6,362 67 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 09 | 6,832 24 | 6,819 52 | 0,006 54 | 6,835 56 | 0,930 82 | 0,930 82 |
| Parallel | 2 | 0,001 15 | 7,110 07 | 7,108 03 | 0,005 42 | 3,557 64 | 1,788 45 | 0,894 23 |
| Parallel | 4 | 0,001 21 | 7,289 13 | 7,287 86 | 0,003 38 | 1,824 86 | 3,486 66 | 0,871 66 |
| Parallel | 8 | 0,002 13 | 7,320 24 | 7,318 62 | 0,006 23 | 0,918 00 | 6,931 01 | 0,866 38 |
| Parallel | 16 | 0,005 98 | 8,453 96 | 8,321 68 | 0,141 60 | 0,862 53 | 7,376 77 | 0,461 05 |
| Parallel | 32 | 0,009 89 | 8,418 78 | 8,324 73 | 0,110 32 | 0,854 03 | 7,450 20 | 0,232 82 |



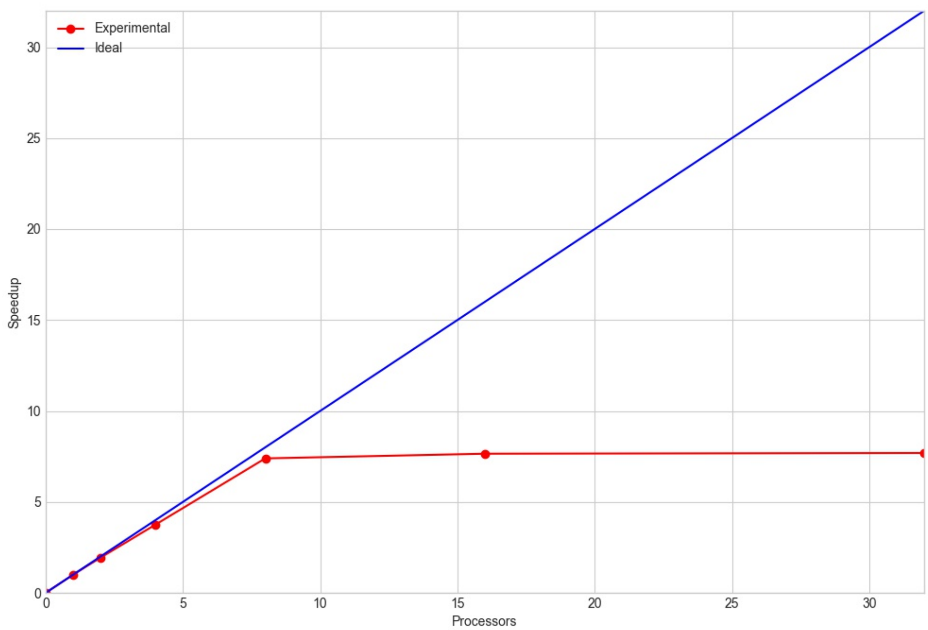
### 2.1.5 Size-100000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,000 86 | 2,134 26 | 2,132 15 | 0,003 24 | 2,136 49 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 09 | 2,132 22 | 2,132 00 | 0,002 46 | 2,135 33 | 1,000 54 | 1,000 54 |
| Parallel | 2 | 0,001 14 | 2,196 87 | 2,198 00 | 0,001 00 | 1,101 00 | 1,940 50 | 0,970 25 |
| Parallel | 4 | 0,001 21 | 2,265 32 | 2,266 32 | 0,002 42 | 0,569 00 | 3,754 81 | 0,938 70 |
| Parallel | 8 | 0,002 08 | 2,271 34 | 2,273 27 | 0,003 00 | 0,286 58 | 7,455 18 | 0,931 90 |
| Parallel | 16 | 0,006 00 | 2,688 98 | 2,615 85 | 0,082 88 | 0,278 59 | 7,668 94 | 0,479 31 |
| Parallel | 32 | 0,010 04 | 2,691 00 | 2,618 57 | 0,088 56 | 0,277 65 | 7,695 03 | 0,240 47 |



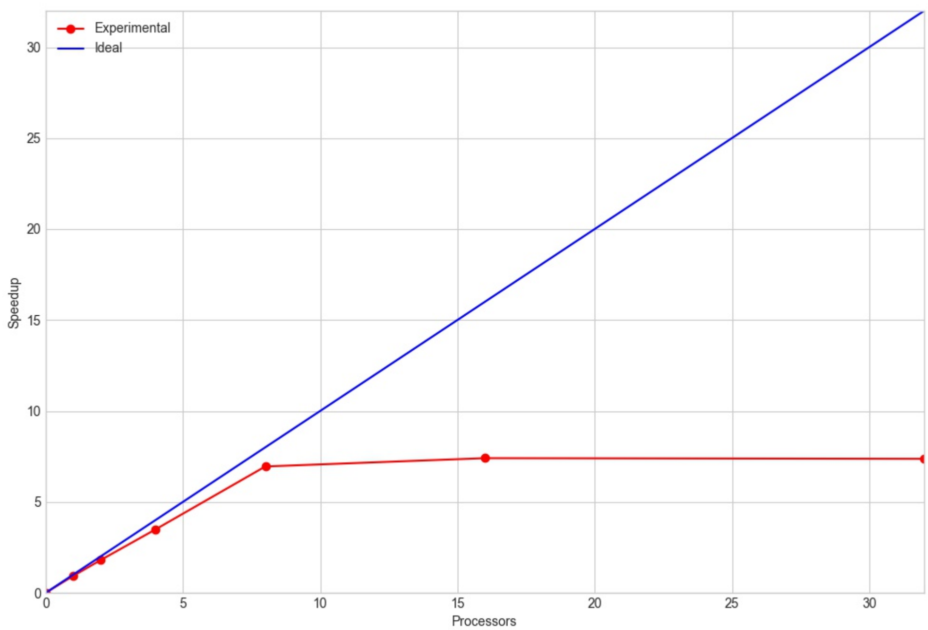
### 2.1.6 Size-100000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,000 86 | 2,132 15 | 2,131 38 | 0,002 00 | 2,134 51 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 08 | 2,132 82 | 2,132 25 | 0,002 53 | 2,135 77 | 0,999 41 | 0,999 41 |
| Parallel | 2 | 0,001 14 | 2,198 38 | 2,198 59 | 0,001 95 | 1,101 98 | 1,936 98 | 0,968 49 |
| Parallel | 4 | 0,001 21 | 2,265 73 | 2,266 51 | 0,002 00 | 0,569 00 | 3,751 34 | 0,937 83 |
| Parallel | 8 | 0,002 19 | 2,281 06 | 2,280 68 | 0,005 08 | 0,288 79 | 7,391 23 | 0,923 90 |
| Parallel | 16 | 0,006 10 | 2,689 87 | 2,614 34 | 0,082 90 | 0,279 06 | 7,648 81 | 0,478 05 |
| Parallel | 32 | 0,010 30 | 2,693 02 | 2,618 57 | 0,085 11 | 0,277 65 | 7,687 77 | 0,240 24 |



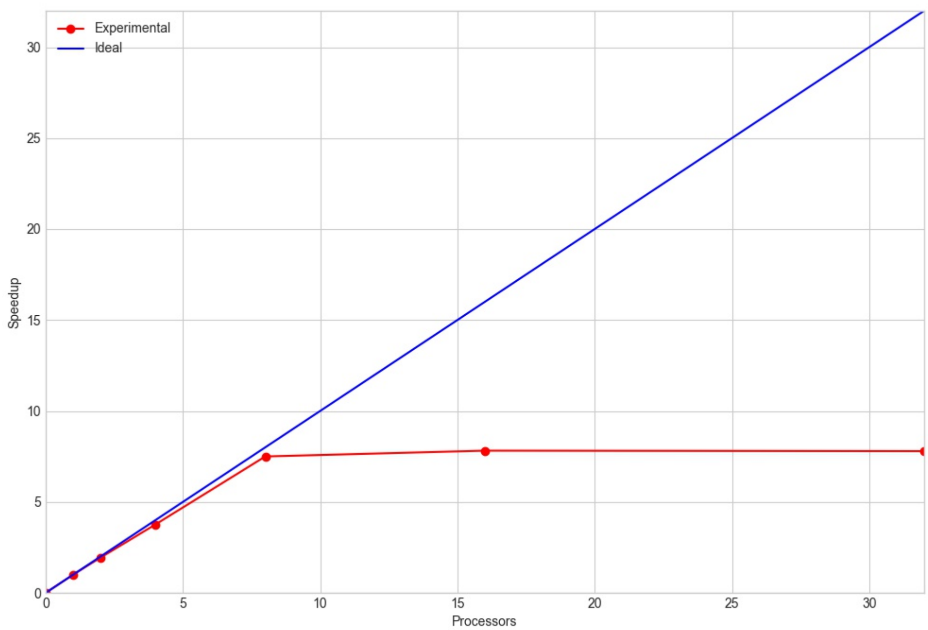
### 2.1.7 Size-150000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,001 28 | 14,271 24 | 14,258 50 | 0,014 11 | 14,274 69 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 51 | 15,357 60 | 15,347 39 | 0,014 43 | 15,361 59 | 0,929 25 | 0,929 25 |
| Parallel | 2 | 0,001 58 | 15,857 89 | 15,850 22 | 0,011 50 | 7,932 11 | 1,799 61 | 0,899 80 |
| Parallel | 4 | 0,001 65 | 16,367 24 | 16,363 32 | 0,008 43 | 4,094 29 | 3,486 48 | 0,871 62 |
| Parallel | 8 | 0,002 78 | 16,413 29 | 16,410 30 | 0,008 70 | 2,054 88 | 6,946 72 | 0,868 34 |
| Parallel | 16 | 0,008 49 | 18,411 42 | 18,301 78 | 0,119 14 | 1,928 32 | 7,402 67 | 0,462 67 |
| Parallel | 32 | 0,014 13 | 18,418 16 | 18,290 33 | 0,146 94 | 1,937 62 | 7,367 14 | 0,230 22 |



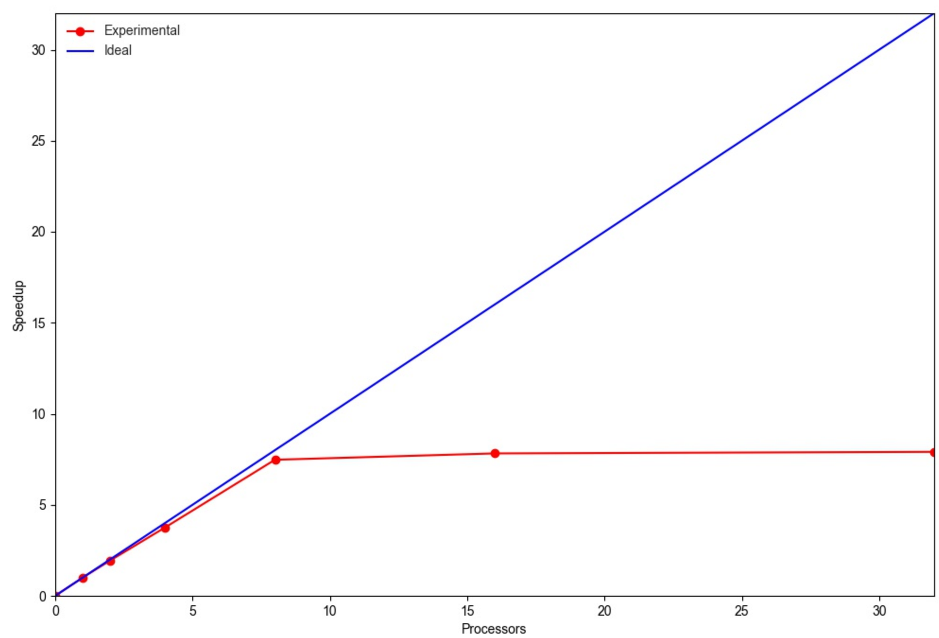
### 2.1.8 Size-150000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,001 27 | 4,799 27 | 4,796 00 | 0,005 62 | 4,802 08 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 51 | 4,799 71 | 4,796 69 | 0,006 10 | 4,803 57 | 0,999 69 | 0,999 69 |
| Parallel | 2 | 0,001 58 | 4,944 90 | 4,943 95 | 0,004 00 | 2,475 44 | 1,939 89 | 0,969 94 |
| Parallel | 4 | 0,001 66 | 5,096 91 | 5,097 72 | 0,003 00 | 1,277 00 | 3,760 44 | 0,940 11 |
| Parallel | 8 | 0,002 81 | 5,102 24 | 5,104 26 | 0,003 00 | 0,640 47 | 7,497 77 | 0,937 22 |
| Parallel | 16 | 0,008 30 | 5,800 88 | 5,740 97 | 0,072 11 | 0,614 39 | 7,816 05 | 0,488 50 |
| Parallel | 32 | 0,013 70 | 5,796 07 | 5,746 79 | 0,070 29 | 0,616 47 | 7,789 67 | 0,243 43 |



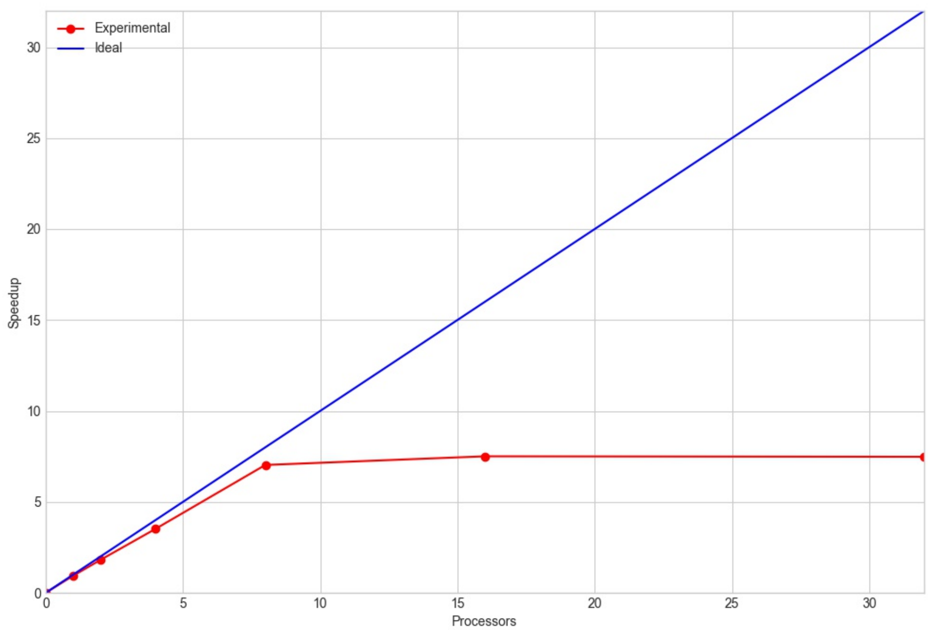
### 2.1.9 Size-150000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,001 29 | 4,798 56 | 4,794 96 | 0,005 47 | 4,801 53 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 52 | 4,800 26 | 4,795 25 | 0,007 67 | 4,803 77 | 0,999 53 | 0,999 53 |
| Parallel | 2 | 0,001 58 | 4,948 97 | 4,946 39 | 0,005 69 | 2,477 49 | 1,938 06 | 0,969 03 |
| Parallel | 4 | 0,001 67 | 5,104 01 | 5,102 06 | 0,005 62 | 1,279 00 | 3,754 13 | 0,938 53 |
| Parallel | 8 | 0,002 88 | 5,114 85 | 5,115 10 | 0,005 35 | 0,642 48 | 7,473 45 | 0,934 18 |
| Parallel | 16 | 0,009 93 | 5,948 47 | 5,867 15 | 0,089 14 | 0,613 94 | 7,820 82 | 0,488 80 |
| Parallel | 32 | 0,012 97 | 5,926 90 | 5,867 86 | 0,076 41 | 0,607 28 | 7,906 61 | 0,247 08 |



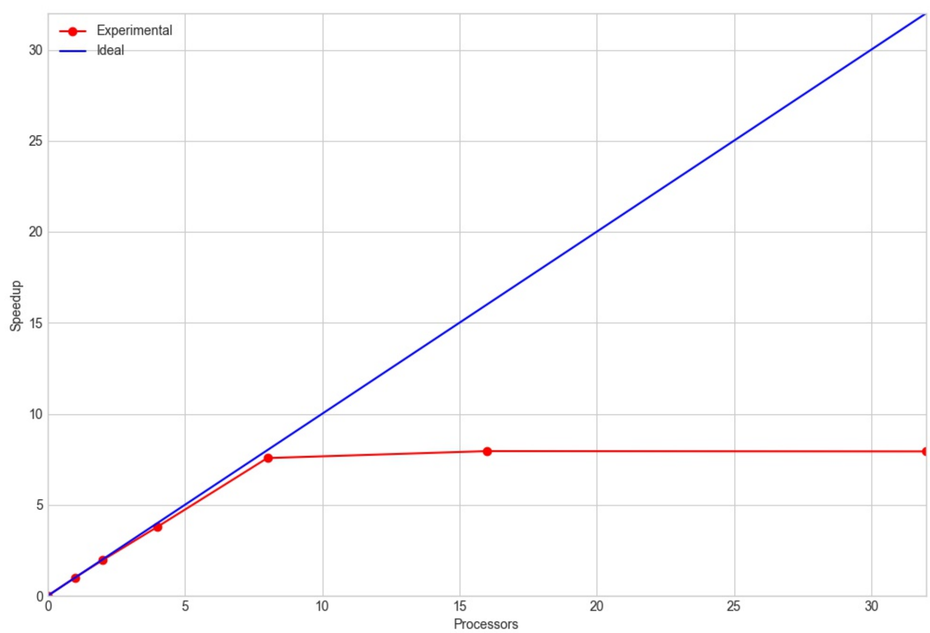
### 2.1.10 Size-200000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,001 71 | 25,671 83 | 25,464 88 | 0,222 51 | 25,671 44 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 95 | 27,339 63 | 27,265 35 | 0,043 89 | 27,344 28 | 0,938 82 | 0,938 82 |
| Parallel | 2 | 0,002 03 | 28,278 44 | 28,257 05 | 0,023 14 | 14,141 13 | 1,815 37 | 0,907 69 |
| Parallel | 4 | 0,002 12 | 29,194 80 | 29,180 64 | 0,017 76 | 7,302 32 | 3,515 52 | 0,878 88 |
| Parallel | 8 | 0,003 52 | 29,190 98 | 29,181 79 | 0,015 13 | 3,652 08 | 7,029 26 | 0,878 66 |
| Parallel | 16 | 0,012 27 | 32,583 31 | 32,442 55 | 0,157 66 | 3,420 63 | 7,504 90 | 0,469 06 |
| Parallel | 32 | 0,018 70 | 32,636 20 | 32,473 72 | 0,174 00 | 3,432 32 | 7,479 32 | 0,233 73 |



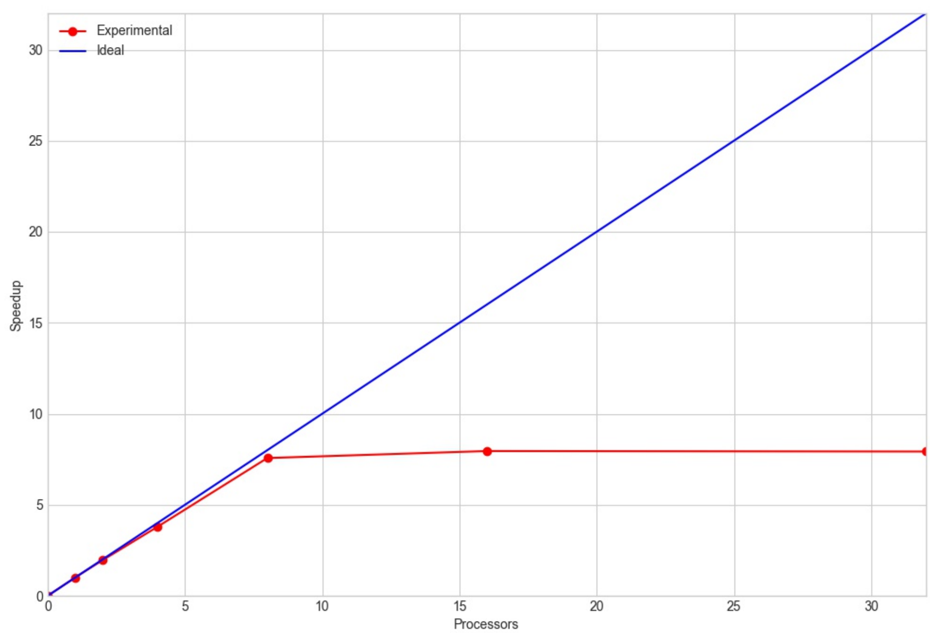
### 2.1.11 Size-200000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,001 72 | 8,602 40 | 8,523 29 | 0,081 50 | 8,605 66 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 94 | 8,532 78 | 8,523 53 | 0,012 66 | 8,536 78 | 1,008 07 | 1,008 07 |
| Parallel | 2 | 0,002 02 | 8,791 80 | 8,787 61 | 0,007 55 | 4,399 42 | 1,956 09 | 0,978 04 |
| Parallel | 4 | 0,002 13 | 9,064 74 | 9,063 48 | 0,005 57 | 2,269 30 | 3,792 21 | 0,948 05 |
| Parallel | 8 | 0,003 51 | 9,074 46 | 9,075 28 | 0,005 29 | 1,137 43 | 7,565 85 | 0,945 73 |
| Parallel | 16 | 0,010 73 | 10,256 95 | 10,173 09 | 0,093 84 | 1,082 87 | 7,947 10 | 0,496 69 |
| Parallel | 32 | 0,017 65 | 10,257 49 | 10,183 50 | 0,093 75 | 1,085 11 | 7,930 65 | 0,247 83 |



### 2.1.12 Size-200000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,001 72 | 8,603 07 | 8,523 00 | 0,082 27 | 8,606 32 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,001 94 | 8,533 59 | 8,524 51 | 0,012 35 | 8,537 73 | 1,008 03 | 1,008 03 |
| Parallel | 2 | 0,002 02 | 8,791 97 | 8,788 00 | 0,008 00 | 4,399 25 | 1,956 31 | 0,978 16 |
| Parallel | 4 | 0,002 12 | 9,065 30 | 9,064 00 | 0,005 50 | 2,269 31 | 3,792 48 | 0,948 12 |
| Parallel | 8 | 0,003 49 | 9,074 93 | 9,075 55 | 0,005 32 | 1,137 36 | 7,566 90 | 0,945 86 |
| Parallel | 16 | 0,010 90 | 10,247 02 | 10,177 00 | 0,085 51 | 1,082 37 | 7,951 33 | 0,496 96 |
| Parallel | 32 | 0,016 34 | 10,262 60 | 10,188 03 | 0,091 54 | 1,086 09 | 7,924 12 | 0,247 63 |



## 2.2 Case study n°2

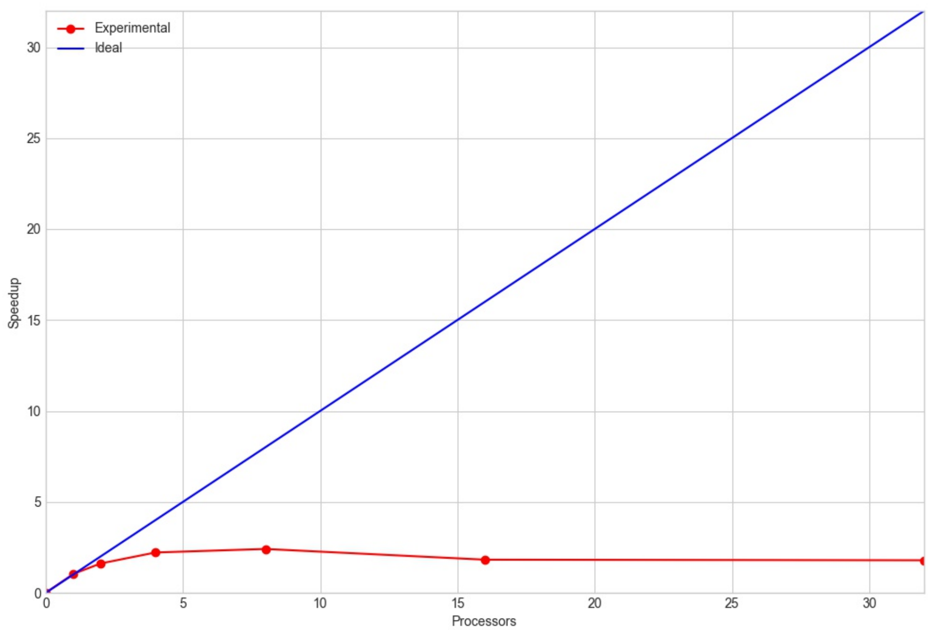
In this case study, the main purpose was to analyze the performance of program with complexity $O(n)$ in the following build setup:

- The sequential program are compiled with the gcc optimization -Ox where x = 1,2,3
- The parallel programs are compiled with the gcc optimization -Ox where x = 1,2,3

So here we want to highlight the difference between a sequential program compared to a parallel one, both compiled with the compiler optimizations. Furthermore the case study is done on multiple size that are 50000000, 100000000, 150000000, 200000000, with different number of threads (0, 1, 2, 4, 8, 16, 32) on 50 repetitions.

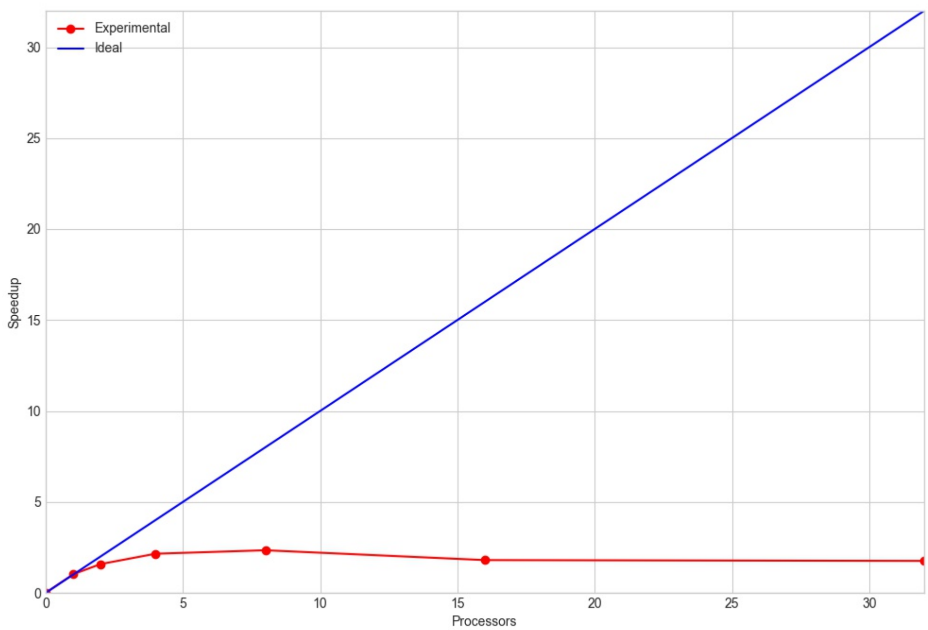
### 2.2.1 Size-50000000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,427 17 | 0,183 99 | 0,581 27 | 0,035 47 | 0,617 12 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,430 31 | 0,160 44 | 0,561 06 | 0,035 55 | 0,596 94 | 1,033 81 | 1,033 81 |
| Parallel | 2 | 0,445 46 | 0,307 50 | 0,724 30 | 0,038 41 | 0,382 40 | 1,613 81 | 0,806 91 |
| Parallel | 4 | 0,475 20 | 0,614 23 | 0,997 23 | 0,101 88 | 0,278 29 | 2,217 54 | 0,554 38 |
| Parallel | 8 | 0,772 65 | 1,016 79 | 1,432 06 | 0,356 97 | 0,255 97 | 2,410 88 | 0,301 36 |
| Parallel | 16 | 1,767 61 | 0,903 17 | 2,019 43 | 0,663 76 | 0,338 73 | 1,821 89 | 0,113 87 |
| Parallel | 32 | 1,785 74 | 0,932 00 | 2,054 27 | 0,679 53 | 0,344 87 | 1,789 43 | 0,055 92 |



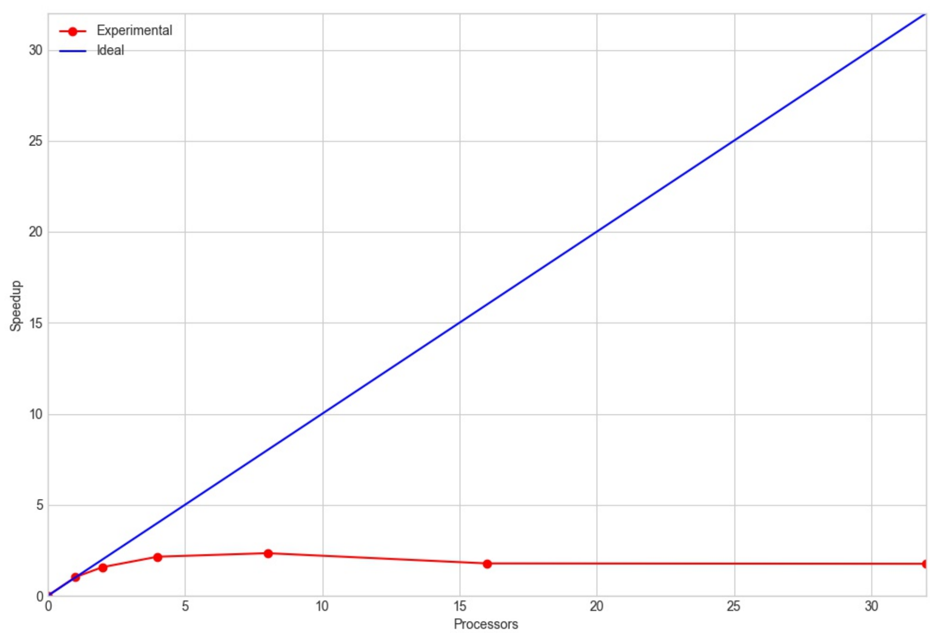
### 2.2.2 Size-50000000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,427 27 | 0,184 26 | 0,581 89 | 0,035 00 | 0,617 18 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,429 60 | 0,168 19 | 0,569 10 | 0,034 49 | 0,604 10 | 1,021 65 | 1,021 65 |
| Parallel | 2 | 0,445 61 | 0,323 48 | 0,740 69 | 0,038 28 | 0,390 48 | 1,580 60 | 0,790 30 |
| Parallel | 4 | 0,475 70 | 0,645 65 | 1,048 44 | 0,097 04 | 0,287 13 | 2,149 50 | 0,537 37 |
| Parallel | 8 | 0,769 59 | 1,072 32 | 1,547 36 | 0,299 26 | 0,263 78 | 2,339 76 | 0,292 47 |
| Parallel | 16 | 1,750 31 | 0,912 33 | 2,016 23 | 0,664 37 | 0,342 90 | 1,799 89 | 0,112 49 |
| Parallel | 32 | 1,779 08 | 0,972 03 | 2,051 88 | 0,694 97 | 0,351 59 | 1,755 38 | 0,054 86 |



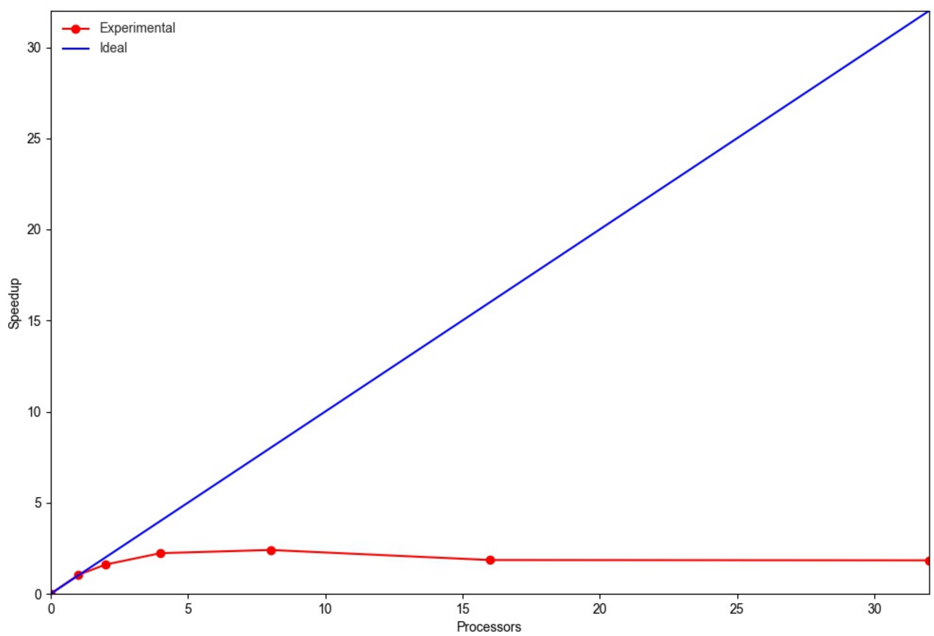
### 2.2.3 Size-50000000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,426 81 | 0,183 73 | 0,581 34 | 0,034 76 | 0,616 35 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,428 91 | 0,167 73 | 0,568 80 | 0,033 94 | 0,603 02 | 1,022 10 | 1,022 10 |
| Parallel | 2 | 0,445 61 | 0,324 32 | 0,740 91 | 0,038 48 | 0,391 04 | 1,576 16 | 0,788 08 |
| Parallel | 4 | 0,475 35 | 0,645 44 | 1,049 70 | 0,088 30 | 0,287 18 | 2,146 19 | 0,536 55 |
| Parallel | 8 | 0,770 02 | 1,038 49 | 1,469 70 | 0,328 42 | 0,263 37 | 2,340 26 | 0,292 53 |
| Parallel | 16 | 1,755 66 | 0,923 41 | 2,015 76 | 0,676 00 | 0,346 38 | 1,779 38 | 0,111 21 |
| Parallel | 32 | 1,776 08 | 0,948 96 | 2,037 03 | 0,692 19 | 0,350 45 | 1,758 72 | 0,054 96 |



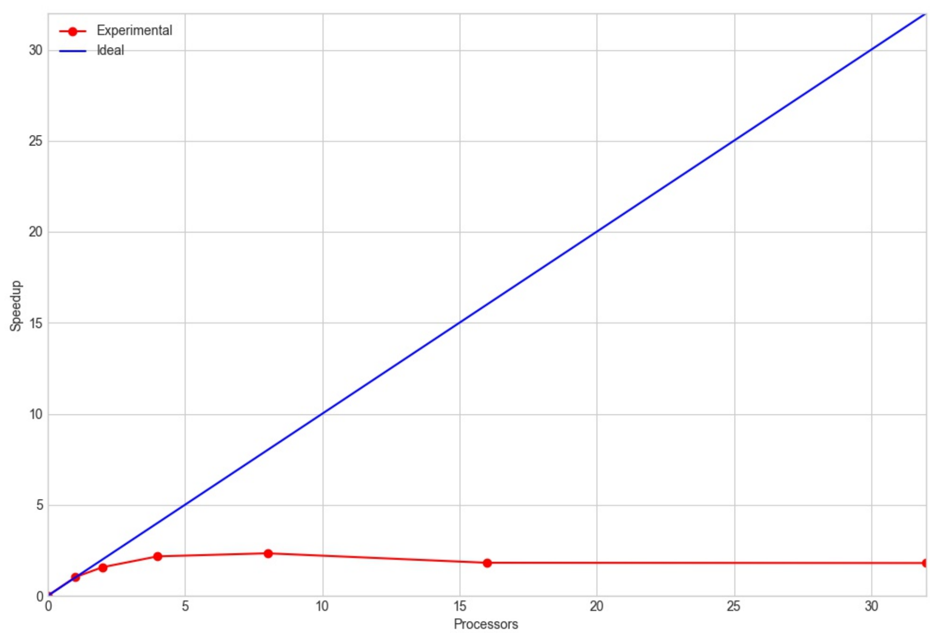
### 2.2.4 Size-100000000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,857 11 | 0,382 91 | 1,176 03 | 0,072 49 | 1,249 30 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,860 72 | 0,338 07 | 1,137 37 | 0,070 30 | 1,208 61 | 1,033 67 | 1,033 67 |
| Parallel | 2 | 0,889 71 | 0,593 62 | 1,415 78 | 0,076 62 | 0,779 51 | 1,602 67 | 0,801 34 |
| Parallel | 4 | 0,935 86 | 0,993 93 | 1,809 51 | 0,145 48 | 0,560 89 | 2,227 38 | 0,556 84 |
| Parallel | 8 | 1,540 83 | 2,017 43 | 3,476 81 | 0,176 74 | 0,520 00 | 2,402 51 | 0,300 31 |
| Parallel | 16 | 3,547 65 | 1,443 27 | 3,996 74 | 0,994 31 | 0,674 81 | 1,851 35 | 0,115 71 |
| Parallel | 32 | 3,560 90 | 1,442 25 | 4,006 81 | 1,002 24 | 0,680 62 | 1,835 54 | 0,057 36 |



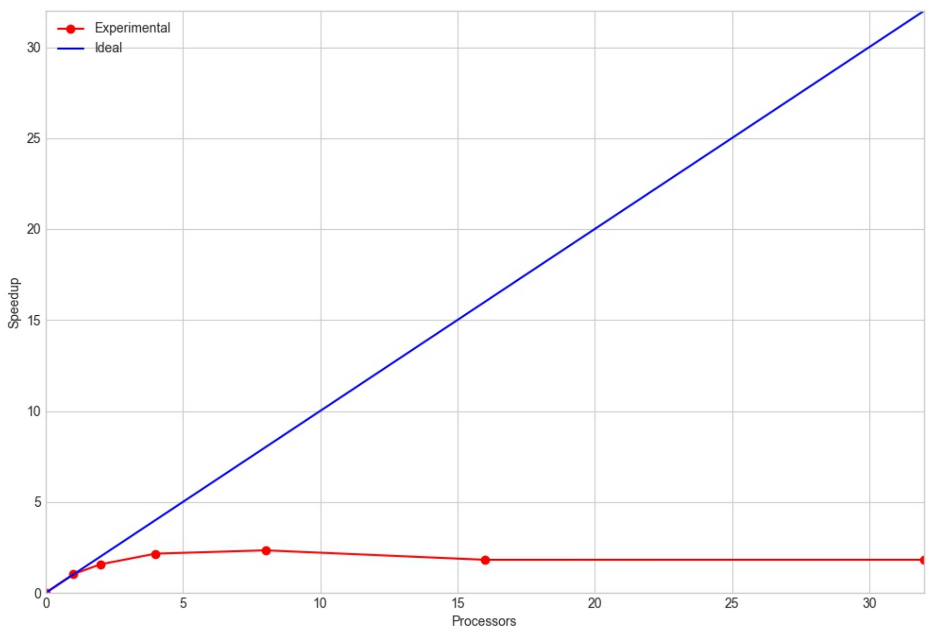
### 2.2.5 Size-100000000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,856 82 | 0,383 56 | 1,176 07 | 0,071 92 | 1,249 46 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,859 20 | 0,352 90 | 1,151 69 | 0,069 06 | 1,221 50 | 1,022 89 | 1,022 89 |
| Parallel | 2 | 0,889 60 | 0,624 12 | 1,446 82 | 0,075 88 | 0,794 65 | 1,572 34 | 0,786 17 |
| Parallel | 4 | 0,935 60 | 1,152 84 | 1,919 06 | 0,177 05 | 0,577 11 | 2,165 02 | 0,541 26 |
| Parallel | 8 | 1,539 32 | 2,229 84 | 3,636 23 | 0,143 42 | 0,535 37 | 2,333 83 | 0,291 73 |
| Parallel | 16 | 3,537 91 | 1,501 78 | 4,003 82 | 1,029 69 | 0,688 61 | 1,814 48 | 0,113 40 |
| Parallel | 32 | 3,559 86 | 1,448 83 | 4,006 50 | 0,997 79 | 0,694 46 | 1,799 18 | 0,056 22 |



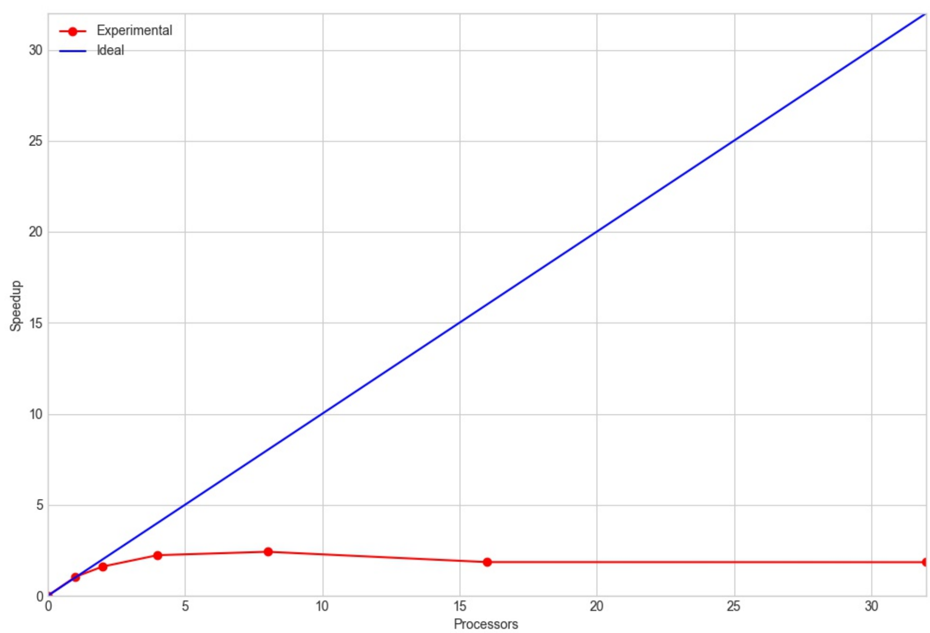
### 2.2.6 Size-100000000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 0,857 02 | 0,382 78 | 1,176 14 | 0,072 44 | 1,249 27 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 0,858 47 | 0,353 32 | 1,152 47 | 0,068 73 | 1,221 76 | 1,022 52 | 1,022 52 |
| Parallel | 2 | 0,889 04 | 0,623 30 | 1,445 77 | 0,075 09 | 0,795 85 | 1,569 73 | 0,784 87 |
| Parallel | 4 | 0,942 48 | 1,160 49 | 1,939 71 | 0,178 52 | 0,580 24 | 2,153 02 | 0,538 25 |
| Parallel | 8 | 1,537 05 | 2,233 98 | 3,642 23 | 0,136 81 | 0,535 13 | 2,334 50 | 0,291 81 |
| Parallel | 16 | 3,540 07 | 1,497 93 | 4,004 17 | 1,031 90 | 0,687 13 | 1,818 09 | 0,113 63 |
| Parallel | 32 | 3,441 00 | 1,524 01 | 3,944 42 | 1,044 50 | 0,686 76 | 1,819 09 | 0,056 85 |



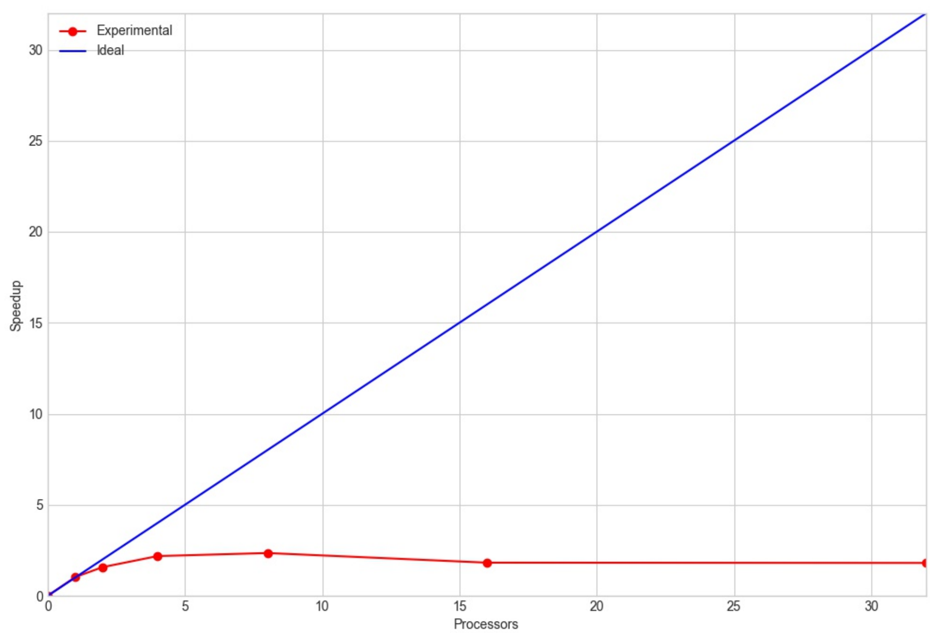
### 2.2.7 Size-150000000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 1,285 62 | 0,577 23 | 1,765 43 | 0,109 30 | 1,875 81 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 1,287 06 | 0,511 59 | 1,708 90 | 0,104 88 | 1,813 30 | 1,034 47 | 1,034 47 |
| Parallel | 2 | 1,330 51 | 0,792 13 | 2,027 52 | 0,109 04 | 1,168 06 | 1,605 91 | 0,802 95 |
| Parallel | 4 | 1,394 88 | 1,205 28 | 2,428 61 | 0,198 46 | 0,841 20 | 2,229 92 | 0,557 48 |
| Parallel | 8 | 2,303 64 | 2,071 36 | 4,082 34 | 0,315 78 | 0,776 06 | 2,417 09 | 0,302 14 |
| Parallel | 16 | 5,339 29 | 1,862 34 | 5,994 95 | 1,220 48 | 1,012 90 | 1,851 92 | 0,115 75 |
| Parallel | 32 | 5,358 91 | 1,877 17 | 6,013 97 | 1,238 18 | 1,016 35 | 1,845 63 | 0,057 68 |



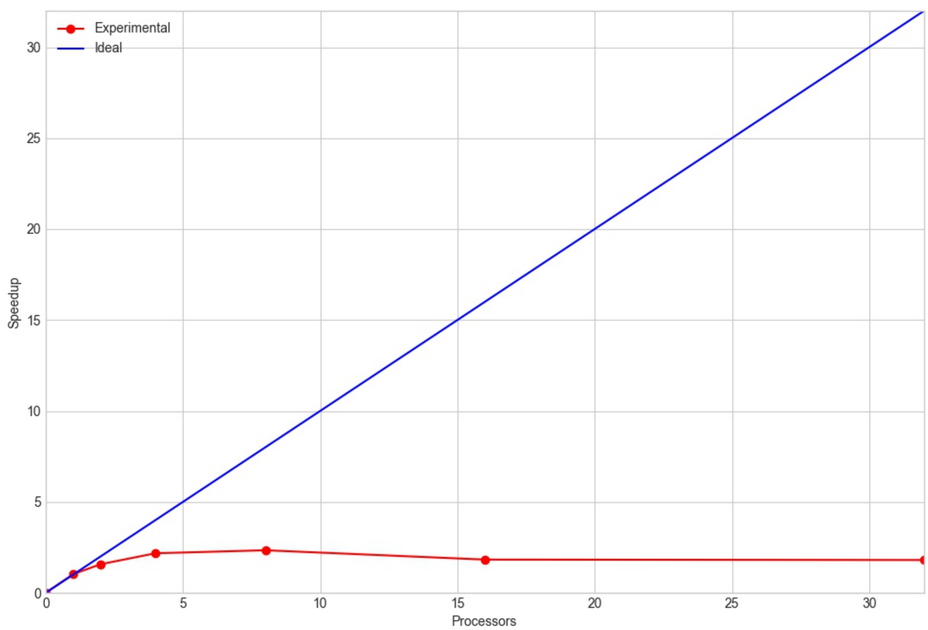
### 2.2.8 Size-150000000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 1,285 69 | 0,577 43 | 1,765 83 | 0,109 91 | 1,876 70 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 1,281 63 | 0,532 69 | 1,730 52 | 0,097 23 | 1,827 27 | 1,027 05 | 1,027 05 |
| Parallel | 2 | 1,331 89 | 0,834 56 | 2,069 14 | 0,108 79 | 1,192 39 | 1,573 90 | 0,786 95 |
| Parallel | 4 | 1,389 52 | 1,320 56 | 2,520 70 | 0,203 88 | 0,860 56 | 2,180 79 | 0,545 20 |
| Parallel | 8 | 2,301 39 | 2,189 67 | 4,155 46 | 0,332 92 | 0,799 08 | 2,348 57 | 0,293 57 |
| Parallel | 16 | 5,317 89 | 1,951 80 | 6,001 71 | 1,283 55 | 1,030 97 | 1,820 33 | 0,113 77 |
| Parallel | 32 | 5,356 59 | 2,030 38 | 6,073 43 | 1,343 05 | 1,040 05 | 1,804 43 | 0,056 39 |



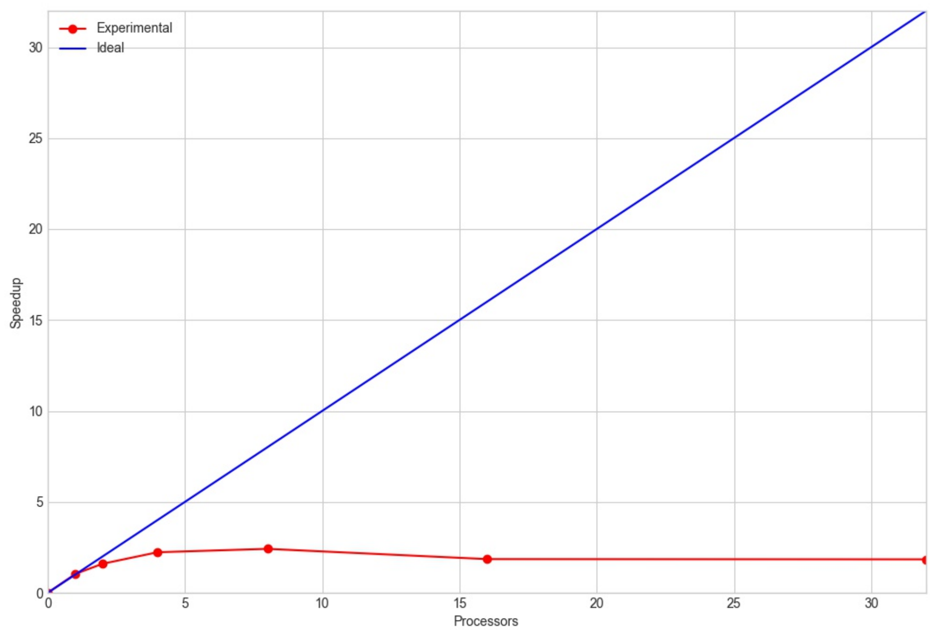
### 2.2.9 Size-150000000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 1,280 87 | 0,580 06 | 1,767 80 | 0,105 32 | 1,874 44 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 1,280 26 | 0,533 09 | 1,729 22 | 0,096 97 | 1,827 30 | 1,025 80 | 1,025 80 |
| Parallel | 2 | 1,332 21 | 0,832 30 | 2,068 87 | 0,108 62 | 1,192 93 | 1,571 29 | 0,785 65 |
| Parallel | 4 | 1,391 04 | 1,330 67 | 2,526 08 | 0,210 36 | 0,862 19 | 2,174 05 | 0,543 51 |
| Parallel | 8 | 2,301 92 | 2,266 59 | 4,285 49 | 0,339 95 | 0,801 12 | 2,339 78 | 0,292 47 |
| Parallel | 16 | 5,202 85 | 1,877 65 | 5,887 20 | 1,224 68 | 1,026 13 | 1,826 71 | 0,114 17 |
| Parallel | 32 | 5,346 13 | 1,869 49 | 6,092 83 | 1,187 21 | 1,039 91 | 1,802 50 | 0,056 33 |



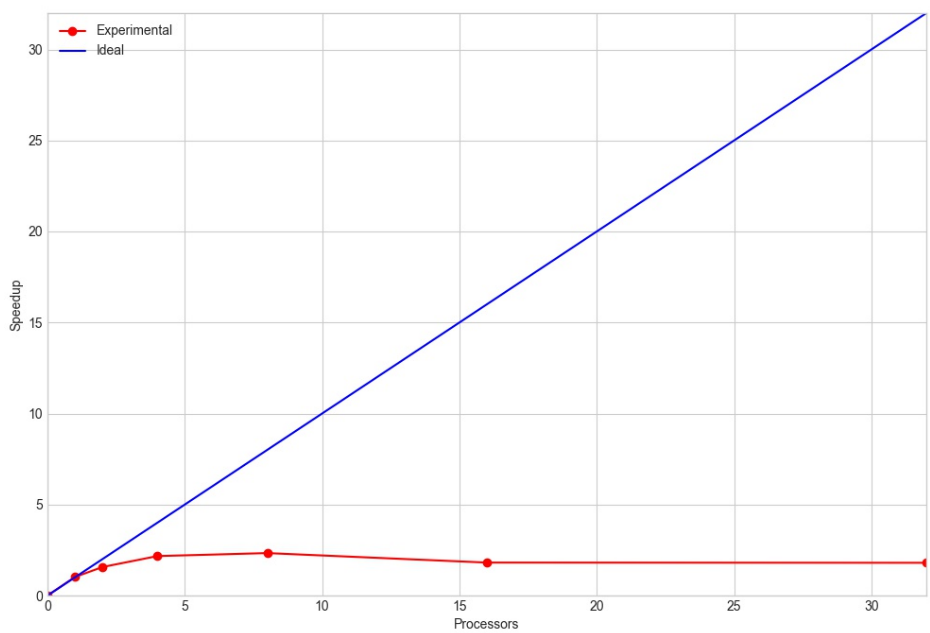
### 2.2.10 Size-200000000-O1

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 1,707 76 | 0,782 18 | 2,363 91 | 0,140 86 | 2,503 73 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 1,714 39 | 0,688 66 | 2,280 15 | 0,137 07 | 2,420 03 | 1,034 59 | 1,034 59 |
| Parallel | 2 | 1,785 72 | 0,994 84 | 2,639 90 | 0,155 84 | 1,566 82 | 1,597 97 | 0,798 99 |
| Parallel | 4 | 1,852 57 | 1,467 28 | 3,076 14 | 0,265 59 | 1,122 68 | 2,230 14 | 0,557 54 |
| Parallel | 8 | 3,069 91 | 2,008 75 | 4,663 74 | 0,460 72 | 1,036 03 | 2,416 66 | 0,302 08 |
| Parallel | 16 | 7,076 23 | 1,993 94 | 8,030 79 | 1,053 00 | 1,350 53 | 1,853 88 | 0,115 87 |
| Parallel | 32 | 7,144 62 | 1,973 63 | 8,054 39 | 1,072 84 | 1,360 51 | 1,840 28 | 0,057 51 |



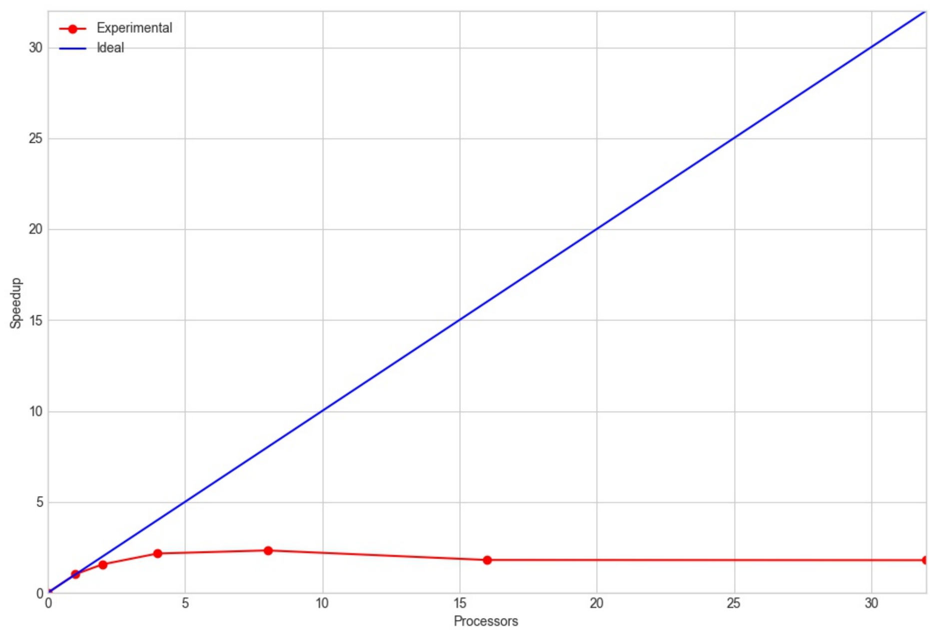
### 2.2.11 Size-200000000-O2

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 1,703 81 | 0,782 93 | 2,366 62 | 0,137 28 | 2,502 53 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 1,715 74 | 0,722 45 | 2,314 21 | 0,140 00 | 2,456 44 | 1,018 76 | 1,018 76 |
| Parallel | 2 | 1,786 11 | 1,059 47 | 2,703 00 | 0,155 78 | 1,599 15 | 1,564 91 | 0,782 46 |
| Parallel | 4 | 1,846 07 | 1,625 38 | 3,155 23 | 0,315 82 | 1,154 50 | 2,167 63 | 0,541 91 |
| Parallel | 8 | 3,067 68 | 2,204 39 | 4,833 97 | 0,498 27 | 1,073 21 | 2,331 83 | 0,291 48 |
| Parallel | 16 | 7,077 20 | 2,172 52 | 8,074 00 | 1,167 20 | 1,382 31 | 1,810 40 | 0,113 15 |
| Parallel | 32 | 7,132 41 | 2,173 56 | 8,098 08 | 1,174 82 | 1,391 71 | 1,798 17 | 0,056 19 |



### 2.2.12 Size-200000000-O3

| Version | Threads | Init | Counting Sort | User | Sys | Elapsed | Speedup | Efficiency |
|---|---|---|---|---|---|---|---|---|
| Serial | 1 | 1,702 70 | 0,780 00 | 2,361 10 | 0,135 76 | 2,499 78 | 1,000 00 | 1,000 00 |
| Parallel | 1 | 1,724 23 | 0,720 42 | 2,311 80 | 0,148 90 | 2,459 00 | 1,016 58 | 1,016 58 |
| Parallel | 2 | 1,783 28 | 1,055 13 | 2,700 82 | 0,154 64 | 1,596 00 | 1,566 28 | 0,783 14 |
| Parallel | 4 | 1,849 39 | 1,642 82 | 3,235 96 | 0,280 80 | 1,157 03 | 2,160 51 | 0,540 13 |
| Parallel | 8 | 3,066 02 | 2,223 61 | 4,819 97 | 0,499 40 | 1,072 76 | 2,330 22 | 0,291 28 |
| Parallel | 16 | 7,082 85 | 2,198 15 | 8,095 70 | 1,176 00 | 1,384 07 | 1,806 11 | 0,112 88 |
| Parallel | 32 | 7,132 99 | 2,160 20 | 8,138 43 | 1,168 11 | 1,392 62 | 1,795 02 | 0,056 09 |



# Chapter 3

# Considerations

## 3.1 Case study n°1

In this case both programs, serial and parallel, were compiled with the same optimization options, the speedup was calculated respect to the serial program with gcc optimizations. The maximum computed speedup is 7. The elapsed time increases steadily with the size of the problem, but the elapsed time of the parallel program doesn't increase at the same rate; so the speedup increases a bit with the problem size: the serial program is getting much slower as the size grows. The best configuration is with the parallel version of 8 threads on an array of 200,000 elements with O3 optimization.

## 3.2 Case study n°2

Serial and parallel programs are also compiled with gcc optimisations. The maximum computed speedup is 2. Although the arrays have a dimension 1000 times greater than in the previous case it can be noticed that the elapsed time is always smaller. The best configuration is with the parallel version of 8 threads on an array of 200,000,000 elements with O1 optimization.

## 3.3 Other considerations

The choice of scheduling is left to OMP, and it is the best when compared to other types of scheduling. Another type of optimisation in the code could be to exploit data locality and caches or loop unrolling techniques to improve performance, but in all cases the gcc optimisers (especially -O3, the most aggressive) adopt all the improvement techniques.

## 3.4 Final considerations

The array is initialised by the generateArray function, which uses rand_r to generate random numbers to insert into it.

Very high values for the array size are used, as using smaller arrays the init time was too low (assuming values of about $10^{-7}$), therefore not evaluable.

One thing worth noting is that the best speedup is around 8 with the first case study, as the processor has 8 cores (high performance)

In the second case study we achieve a speedup of about 2 because the counting sort function cannot be executed in parallel due to the data dependency. The maximum theoretic speedup depends on the fraction of sequential part on the whole program according to Ahmdal's Law.

# Chapter 4

# API

## 4.1 Public Functions

| Type | Name |
|---|---|
| void | **generateArray**(ELEMENT_TYPE a[], int n, int threads)<br>This function generate an array with randomic numbers |
| void | **countSort**(ELEMENT_TYPE a[], int n, int threads)<br>This function sorts an array according to the counting sort algorithm using optimized loops, the complexity is $O(n^2)$ |
| void | **countSortOn**(ELEMENT_TYPE A[], ELEMENT_TYPE C[], int length, int threads)<br>This function sorts an array according to the counting sort algorithm using optimized loops, the complexity is $O(n)$ |

## 4.2 Public Functions Documentation

**function generateArray**

```
void generateArray(
    ELEMENT_TYPE a[],
    int n,
    int threads
)
```

Parameters:

- *a* pointer to an empty array which must be populated with random numbers
- *n* size of A

- *threads* number of threads

**function countSort**

```
void countSort(
    ELEMENT_TYPE a[],
    int n,
    int threads
)
```

Parameters:

- *a* a pointer to an array which must be sorted
- *n* size of A
- *threads* number of threads

**function countSortOn**

```
void countSortOn(
    ELEMENT_TYPE A[],
    ELEMENT_TYPE C[],
    int length,
    int threads
)
```

Parameters:

- *A* a pointer to an array which must be sorted
- *C* a pointer to a result array
- *n* size of A
- *threads* number of threads

# Chapter 5

# How to run

1. Create a build directory and launch cmake

```
mkdir build
cd build
cmake ..
```

2. Generate executables with

```
make
```

3. To generate measures for algorithm with complexity $O(n^2)$ (case study n.1) run

```
make generate_measures
```

   or if you want to generate measures for algorithm with complexity $O(n)$ (case study n.2) run

```
make generate_measuresOn
```

4. To extract mean times and speedup curves from measures of algorithm with complexity $O(n^2)$ run

```
make extract_measures
```

   or run

```
make extract_measuresOn
```

   for measures of algorithm with complexity $O(n)$

Results of measures of algorithm with complexity $O(n^2)$ can be found in the ‘measures/measure‘ directory and results of measures of algorithm with complexity $O(n)$ can be found in the ‘measures/measureOn‘, divided by problem size and the gcc optimization option used.

You can find the complete project on GitHub: https://github.com/scov8/CommonAssignment-Team02

The previous year's group 02 files proposed by the professor during the course were used for file generation and extraction.

The counting sort function for test case n. 1 was taken here:
https://github.com/ianliu/programacao-paralela/blob/master/omp-count-sort/main.c