

# Operating Systems Project: Kernel Optimization

Authors:

Juan Manuel Romero Guardado  
Jorge Dominic Márquez Muñoz

October 16, 2016

## 1 Abstract

Every day of our lives, we are aware of the crescent changes of the technology in the daily applications. Nowadays, the increment of new programs, methods, tasks to perform and works to do is clearly exponential compared to how often new technologies and updates to existing ones came into our world a long time ago. All these new collections of nicely looking programs and tools may take us to an era in which the demanding market (in a more industrial approach) will be faster than our apparent performances. This paper shows the impact and importance of an operating system optimization attempt, all in the mood of having way better work in different circumstances.

## 2 Introduction

In recent years, computer science as a general overview has been living a major scaling in terms of demanding a better performance of the system and systems that uses to achieve specific goals in the industrial applications. In this schemes, the various operating systems we can find in the ordinary lives and in more advanced applications gain a special importance in all of these problems. As we know, an operating system is the principal, well, system (better call it program) that manages distinct resources for the computers' hardware mainly. In other words, an operating system is an essential part of the daily computers usage that, in a sense, is in charge of the correct and optimized use of the overall system through which specific objectives will be fulfilled.

Well, we said that the operating system may have optimization included in itself, but the truth is that, in some cases, optimization of an operating system resides in changing some things that are just useless for certain conditions that may constitute a more specific task or approach of an application in the real world. We can think of a very basic example: when somebody buys a plain computer for his or her work, the booting time (this is the time that the operating system takes to "load" itself) can grow slower from the beginning or after a period of time after using it. These problems may arise from having processes in the operating systems that are really not needed by the user. An

example of this can be an antivirus process. Some antivirus programs make the operating system start them for a complete analysis each time. But an average user, that works calmly on its things, doesn't download things that much and has a relatively short amount of files really doesn't need to scan his or her computer every time. And this kind of process really its the RAM memory like crazy.

As we see, the key is to fine the correct changes that can make our OS optimized as we really need it. The purpose of this paper is to expose some changes in the operating system kernel and to compare them in different tests to find out the better scenarios of our current operating system performance.

### 3 Theoretical Framework

As we said before, an operating system is a program that is needed by almost every computer in the present time. It is the component of a computer that makes all the running possible. How can it do it? By many possible operations. The operating systems manage hardware and computer architecture, they serve as process organizers, they act as intermediates between those process and provide special measures and functions to guarantee safe usage and protection to the numerous data that is used internally. So with all this said combined with the information on the other sections we can now confirm that the operating systems are essential parts of today's computers, since they can either deliver really good or bad performances.

On the same thread, if we talk about operating systems we must also talk about one of the most important components of it: the kernel. Many say that the kernel is the main unit of functionality, and the only program that is always running. All this may be true in some sense, but a major definition is needed. The kernel (or nucleus) is the core of the operating system. It is the main responsible of providing services that applications and programs need to work normally. It handles all requests, memory and disk management and scheduling processes, which means that decides which processes are more important than others. It loads when the system starts and stays running in memory all the time, so it must be relatively small to not waste memory other programs might need in the future. The kernel is allocated in the kernel space, a special memory piece that protects it from suffering wrong operation by the user.

Hopefully this information will be useful for the examination of this paper until this point and for the following points. There are some few terms that need to be defined, but we will keep this on section 6 "Development" to encourage the further reading of this paper.

### 4 Objective

As we said some times before, the optimization of the kernel in an operating system is a very important topic to be covered, since the many applications that we can find are as varied as the number of systems we can find, and none of them works the same for all the cases that there might be. This is what we may

call as a part of kernel optimization, the cases in which we find that certain data and values in the inside of our operating system might make it to work better for a certain circumstance, for each of the application we can find can be radically different than the others. The goal of this work is to find values that can make a better performance for our kernel, and compare its behavior to others with different values.

As we may see, there can be a lot of things that we can modify in a kernel to make it work better, but most of these things are not so easy to understand. Every component of the kernel has a specific function, and there are a lot of components in this very kernel. The bad news is that modifying components without having a clue of what to do can end in the complete opposite of the optimization we want to achieve. A bad kernel configuration may end in failures or in leaving the systems completely useless.

## 5 Justification

Once again, the importance of finding the correct kernel optimization for a certain situation is inherent to the work that is done. We can think of all the industrial applications that can use systems like these ones, especially on a massive production company with a high demand. It is obvious for us and for them that the faster the work is done the better business performance we might get, and getting an advantage in speed over the other companies will eventually give you and your company a great boost among your competition. We now talk on some advantages and disadvantages:

Advantages:

1. **Advantage 1:** Optimizing our operating system will take us on to take the most advantage of the available time to make the work that is needed to be done.
2. **Advantage 2:** Optimizing the kernel will make the most of the functionality of our computer systems, and will take them to a better state and to efficiently take advantage of all the components of our computers.

Disadvantages:

1. **Disadvantage 1:** Changing the kernel incorrectly might lead into the complete opposite of the optimization we want to get: system malfunctions, freezing, blue screens, etc.
2. **Disadvantage 2:** If somebody changes the kernel without knowing what he or she is changing, starts a state of malfunction and then tries to return all the adjustments to their original states will be very difficult to settle everything that changes before.

## 6 Development

For the development of this work we decided to change a kernel component called the Runtime. This Runtime value is the one that tells us how much of

a time period can a process on the system take. We executed an AIO Stress using the Phoronix Test Suite. An AIO Stress is a test that generates and loads a system with random work, and the Phoronix Test Suite is a free benchmark for numerous tests in a machine, that can go from simply testing performance or for real quality purposes.

For this experiment, we made 11 different tests with 11 different values for the Runtime value, represented in the kernel as *kernel.sched\_rt\_runtime\_us*. This is the value we must change in our kernel, and for the integers we chose for it we selected a range from 100,000 nanoseconds (which means that all processes may take 0.1 seconds) to 990,000 nanoseconds, going in an interval of 100,000 nanoseconds and passing through the default value of 950,000 nanoseconds.

We ran the test in an Alienware 15 laptop in the terminal of Ubuntu version 16.04.1 LTS. The command we used to change the value of the Runtime variables is `sysctl -w kernel.sched_rt_runtime_us=40`. On the following section we start discussing the results and possible meanings for them.

## 7 Results

The results of our experiments lead to results of average MB/s of data, which are described on the table below:

Value	Average
100000 ns	162.11 MB/s
200000 ns	163.8 MB/s
300000 ns	172.21 MB/s
400000 ns	159.53 MB/s
500000 ns	157.55 MB/s
600000 ns	164.46 MB/s
700000 ns	150.20 MB/s
800000 ns	155.56 MB/s
900000 ns	145.45 MB/s
950000 ns (default)	243.70 MB/s
990000 ns	276.39 MB/s

Table 1: Results of AIO Stress with different Runtime values.

For a quick starting analysis, we can see that values get to change in some sort of exponential style in some values, but then go down and up in what seems to be a random pattern. We attach screenshots of the last two tests in our Appendix (Part 10 of this very document).

We can appreciate better our results by graphing them, which we did with the rudimentary help of Excel, and ended up showing the graph depicted in the following page:

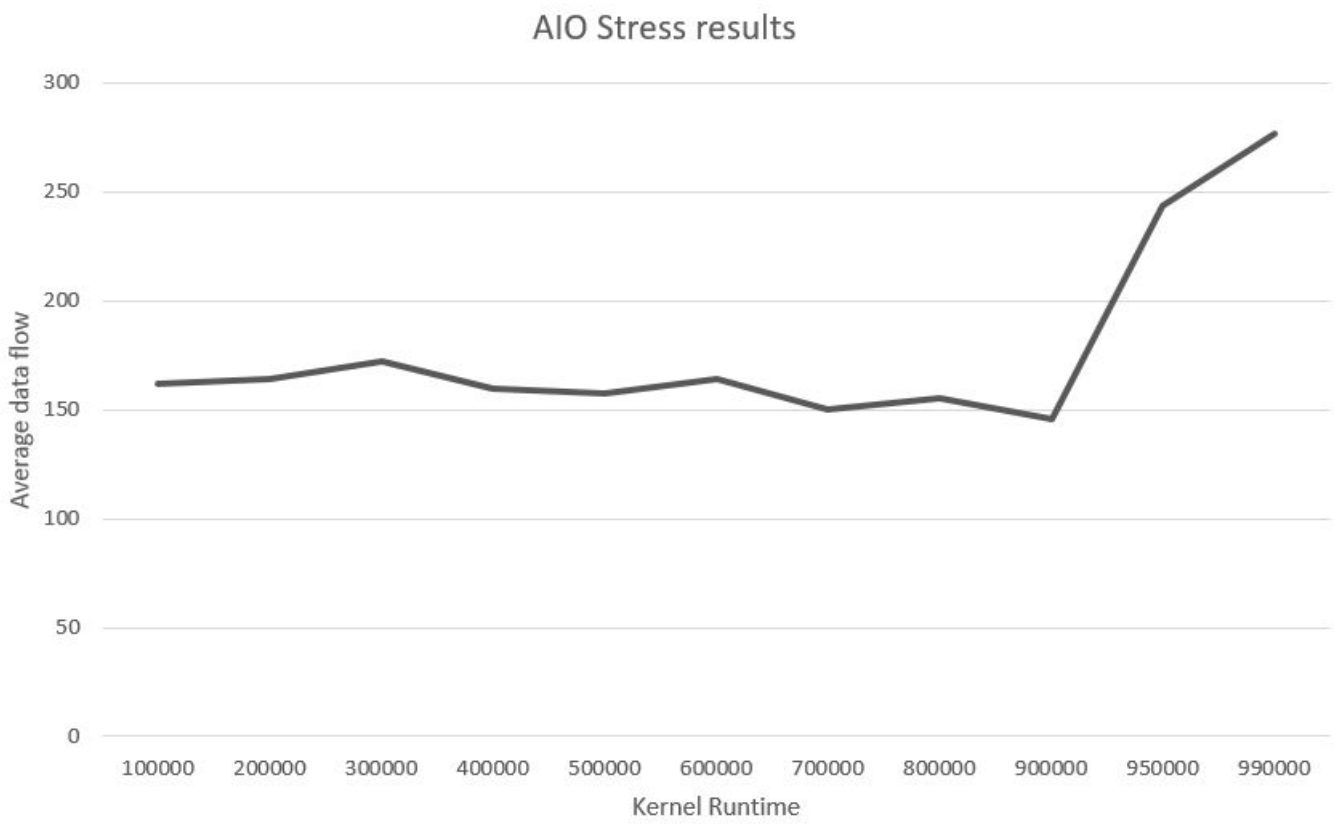


Figure 1: Line graph of the results of the AIO Stress test with different values.

We also get results for a standard error and a standard deviation of the AIO Stress results, but we will leave them aside for further investigation. Analyzing deeper this results, we can come to know that the general better value exists and happens at the bigger value we introduced ad Runtime in the Phoronix tests. We investigated earlier that 990000 nanoseconds is the maximum value for the Runtime variable, and so we saw that it is the one with higher performance. From this part of the topic we can think that finding the optimal value for the parameter and setting it forever in the system would be quick, easy and practical, giving a better system performance no matter what. Well, all of this might be a trap. There are several thing that we should think about before doing that. Principally the hardware components.

Most computer systems nowadays are prepared for the system to run better, but we are watching these systems just as lines of code that are to be executed, and not giving credit to all the physical components that build up the architecture for the system to work. In general, what stops most systems from becoming more powerful are the hardware devices, and in the engineering realm it is well known that the today's hardware devices are not advanced enough to optimize all the uses. Hardware devices are not as modern enough to sustain both the heat generated by the operations of the hardware and the quantity of instructions per clock cycle of a microprocessor. As a previous conclusion, and as we have been saying through this report, there are many things to consider in the end to achieve an optimal performance.

## 8 Conclusion

After all of these analysis we can conclude that all our changing on the values had their own results in making the usage of the operating system better or worse, for we could see both values in our tests. Although we didn't come to a proper real life application of our work (because all of this was done with a previously constructed kernel benchmark test) the tests that were made in the project served for our own understanding of these situations, and could really see that the results are completely real, and that is what must be extrapolated to the real applications of the computers and systems out there.

In all, we know that we only changed a limited series of things in here, and this also works for us to understand that there are a lot of things to be considered for a kernel optimization. Also there may be a previous study of all these factors and also of the outer world factors that can affect the course of the company or industry we could be acting on. Therefore we think that an investigation to achieve a very well optimization of the kernel and the operating system is needed.

Finally, we can see that this optimizations are possible, and should be a very important part of the everyday in a certain industry, whose situation might be a special need of getting a better performance to survive the competition, or just a company that wishes to give a new value to their products, wants to modernize their procedures or wants to earn better profits. Something definitely worth to be checked out.


## 9 References


### References

- [1] Beal, Vangie. (2002). Kernel. october 17, 2016, from Webopedia, web site: <http://www.webopedia.com/TERM/K/kernel.html>
- [2] Holland Kern, Elizabeth. (2006). What is kernel?. october 17, 2016, from TechTarget, web site: <http://searchenterpriselinux.techtarget.com/definition/kernel>
- [3] Kerrisk, Michael. (2016). Sched. October 20, 2016, from The Linux Programming Interface, web site: <http://man7.org/linux/man-pages/man7/sched.7.html>
- [4] McMichael, Jack. (2016). OS optimization tool. october 17, 2016, from VMWare, web site: <https://labs.vmware.com/flings/vmware-os-optimization-tool>
- [5] Rouse, Margaret. (2016). Operating System. october 17, 2016, from WhatIs, web site: <http://whatis.techtarget.com/definition/operating-system-OS>
- [6] Sliberschatz, Avi. &Bale Galvin, Peter. &Gagne, Greg. (2009). Operating System Concepts. USA: John Wiley & Sons, Inc..

## 10 Appendix

In this section we just include the two results of the AIO Stress test. Please recall this is the test we made to test the effectivity of changing the values to the Runtime parameter through this work. Here we can find the results of the two final tests we made: test 13 and 14, with the Runtime values being 950000 nanoseconds and 990000 nanoseconds respectively.

op systems test 13	
	Runtime = 950000
	AIO-Stress 243.70
	Standard Error 0.69
	Standard Deviation 0.49%
PHORONIX-TEST-SUITE.COM	

op systems test 14	
	Runtime = 990000
	AIO-Stress 276.39
	Standard Error 0.41
	Standard Deviation 0.26%
PHORONIX-TEST-SUITE.COM	