



UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO
INE5425 – MODELAGEM E SIMULAÇÃO

Practical Project of Systems Simulation

Project Group 09

Junior Semerzier - 13106344

Marcelo José Dias – 15205398

Thomas Fernandes Feijoo - 12200662

Prof. Rafael Cancian

FLORIANÓPOLIS

2019/1

CONTENTS

1 PROBLEM DESCRIPTION

2 GOALS

3 DATA ANALYSIS AND TREATMENT FOR SIMULATION

4 SYSTEM MODELING AND SIMULATION

5 VALIDATION AND VERIFICATION

6 EXPERIMENTAL PROJECT

7 ANALYSIS OF RESULTS

8 RESULTS PRESENTATION

1. PROBLEM DESCRIPTION

The developer of an operating system wants to evaluate the impact of:

- (a) of the preemptive scheduling algorithm;
- (b) the workload;
- (c) the scheduler's time quantum on the performance of a computer, which is measured as the utilization rate of the processors (the fraction of the time in which processors are running processes).

The system corresponds to a multicore with 4 processors, submitted to a workload composed of CPU-bound and IO-bound processes. In workload, the percentage of CPU-bound processes can range from 50% to 80% (minimum and maximum levels).

The scheduling algorithms to be evaluated are round-robin and priority.

The quantum of time can vary from 100us to 10ms (minimum and maximum levels), after which a process can be preempted.

The timer interrupt frequency is 100us. The operating system can check whether a process needs to be preempted or not at each timer interrupt.

Process priority may vary from -10 to +10 (lower value implies higher priority), with CPU-bound processes having priority normally distributed in this range (average 0 and deviation 10) and IO-bound processes tend to be of a higher priority, according to a triangular with parameters -10, -5, 5.

Both types of processes are characterized by performing cycles involving execution in the CPU and access to devices. The amount of these cycles varies uniformly between 5 and 25. The difference is that in each cycle, CPU-bound processes run in the CPU for a time that has been sampled and that is in the file "tempo_execucao_CPU_bound.txt" and IO-bound processes by a time that was sampled and is in the file "tempo_execucao_IO_bound.txt".

When a process is created it is in the READY state to execute and is placed in a queue to eventually be staggered, according to the scheduling algorithm used. When any processor gets idle, the next process in the ready queue is staggered, its context is loaded (context overhead is 250ns), its state changes to RUN and it runs on the idle processor.

The time the process executes in the processor during that cycle depends on the cycle duration, the time quantum, and the process priority.

In the round-robin algorithm, the processes execute until the end of their cycle if the remaining time is less than or equal to the quantum of time, or they execute until consuming their quantum of time. Upon completion of its quantum of time, if there are processes in the ready queue, the running process is preempted; otherwise, it will gain another quantum of time to execute.

In the priority algorithm, if there are processes with the same priority waiting in the queue, then it is worth the round-robin algorithm. If the running process is more priority than all in the queue, then it executes until it finishes the execution time of its cycle.

After finishing the execution part of its cycle, a process goes to the part of the cycle in which it accesses the peripheral device. The peripheral device is shared so that more than one process can access it simultaneously. The access time to the device varies according to a triangular distribution with parameters 10ms, 50ms, 100ms.

After finishing accessing the device, the process has terminated a cycle (CPU + device). If there are still more cycles to run, the process returns to the ready queue for the next cycle; otherwise, it is finished.

In addition to evaluating the impact of the factors cited on performance metrics, the operating system developer wants to obtain statistics on:

- the overhead of context exchange,
- the lifetime for CPU-bound and IO-bound processes;
- processes in the ready queue.

He would also like to know if he can say with 90% certainty that the waiting time of the processes in the ready queue is different for each scheduling algorithm, and if processes with priority between -10 and 0 (inclusive) have time to wait in the smaller queue when they are staggered by the priority algorithm.

Simulate this system for 1 hour, with a 10% warm-up time.

2. GOALS

Regarding the impact of the variables, we have three that should be considered:

- The scheduling algorithm itself: Priority or Round Robin;
- CPU Bound Chance: The percentage of CPU-bound processes can range from 50% to 80% (minimum and maximum levels)
- Quantum: The quantum of time can vary from 1ms to 10ms (minimum and maximum levels). Note: We changed the minimum level so that we can execute the simulations in a viable time

The combination of these variables, at their maximum and minimum levels, served to analyze:

- Scheduling algorithms;
- Processor utilization rate;
- Context exchange overhead;
- CPU-bound and IO-bound process lifespan;
- Waiting time in the ready queue.

Regarding statistical analysis, we consider the most costly scenario to determine:

- the waiting time of the processes in the ready queue is different for each scheduling algorithm
- if processes with priority between -10 and 0 (inclusive) have queue timeout when they are scaled by the priority algorithm

3. DATA ANALYSIS AND TREATMENT FOR SIMULATION

3.1 Process of sampling and data collection

In this work, we were provided the data of the probability distributions to randomize the execution time of each process. These data were in a text file, one for CPU-bound processes and another for IO-bound processes, so the data collection process was not done because we consider the quality of the data provided to be reliable.

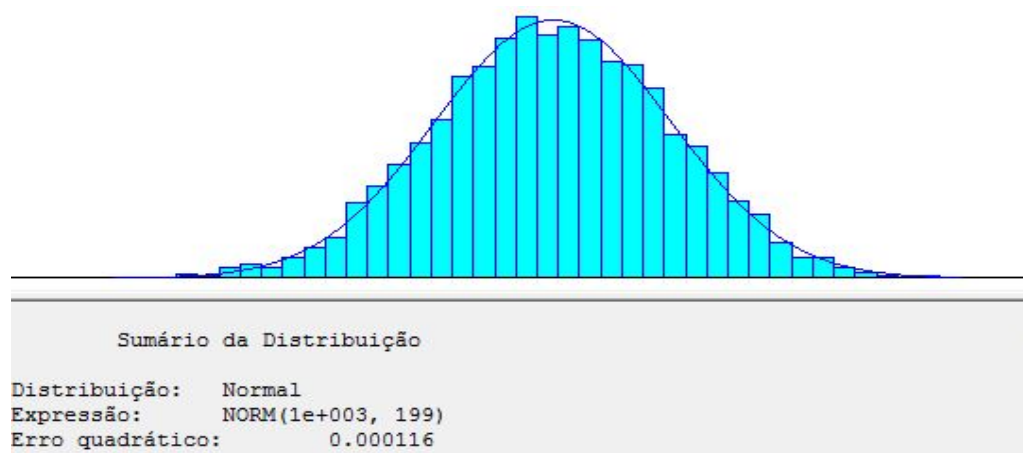
3.2 Data treatment

With the probability distributions given in the problem description, we consider that the treatment of the data to generate the cycle number and execution time of the processes are reliable. Therefore we did not treat these sample data to generate the number of process cycles.

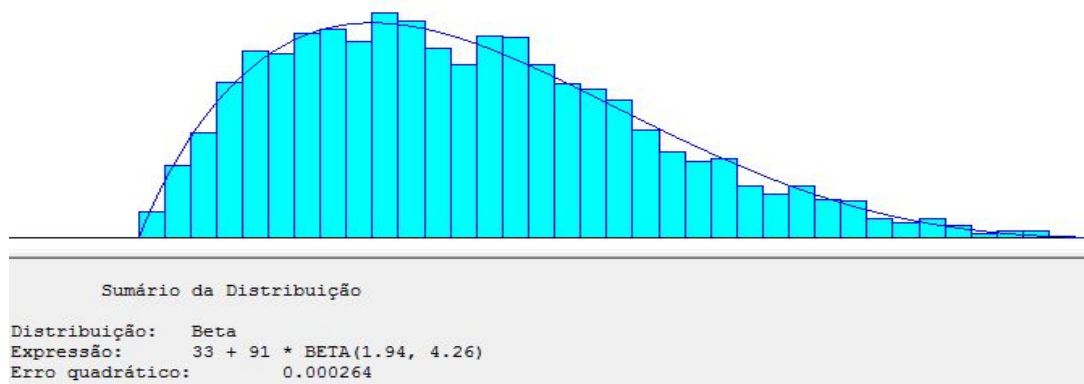
3.3 Identification of probability distribution

To identify the distribution of the file data, meaning, the sample data that are execution time for each cycle of the Cpu bound and Io bound processes, the Arena Input Analyzer tool was used to determine the distribution of these sample data.

For the sampled data collected for the CPU bound processes, this Normal probability distribution was obtained:



And for the IO bound processes, a Beta distribution was found:



These probability distributions were used to generate the execution time for the processes of the two algorithms. But some adjustments were necessary because of the number of an entity within the system that can not exceed 150. Such adjustments will be presented later in this report.

4. SYSTEM MODELING AND SIMULATION

For the modeling of the system was used the model of simulation oriented to events, because when events occur, a change of state in the system is provoked. However, the system was developed in relation to the problem description.

4.1 Arena modeler

To model the step-by-step of each scheduling algorithm, we use the Arena software in its free version. But, because it is a free version, some limitations have been imposed and will be explained later.

The Arena software is an integrated graphical simulation environment that contains all the features for process modeling, design, and animation, statistical analysis and results from the analysis. The software was developed by Rockwell Automation.

4.2 System modeling

Having the problem at hand, we started our solution first by remembering the behavior of the two scheduling algorithms that need to be compared, in this case: Priority and Round-Robin.

The priority algorithm is, as the name itself says, an algorithm where each process in the ready state receives a priority, processes with higher priorities are executed first, priorities that can be assigned dynamically or statically. It is a preemptive algorithm. A small change proposed by the problem is when two or more processes have the same priority and it is necessary to choose which of the two will be scheduled, in this situation is used Round-Robin.

About the Round Robin algorithm, it is one of the simplest and most robust of the current techniques used for load distribution problems, in this scheduling the operating system has a timer, called a quantum, where all processes gain the same quantum value to rotate in the CPU, after the quantum ends and the process has not finished, a preemption occurs and the process is inserted at the end of the queue. If the process terminates before a quantum, the CPU is released for the execution of new processes. In both cases, after the CPU release, a new process is chosen in the ready queue. New processes are inserted at the end of the ready queue. When a process is taken from the ready queue to the CPU, a context switch occurs,

which results in additional processing time. Unlike the priority algorithm, it does not suffer from the starvation problem. It is preemptive.

As we have the algorithms in hand and we know how they work step by step, it becomes easier to do their modeling.

A study period on the Arena was necessary, where we analyzed which components would be interesting for our solution. For example, we need to know which component would be responsible for creating a process, which component would be responsible for setting the attributes of a process, where to store ready processes. Most component choices are intuitive: Create a block for process creation, Decide for flow checks, Assign for set attribute. But there are parts that can be considered non-intuitive, and we will explain its implementation later.

Another important part of modeling is the possibility of extracting statistics, which resulted in additional blocks with the sole function of keeping records of behaviors to be analyzed.

Execution parameters were initially determined according to the information contained in the problem description. The execution time of the CPU Bound and IO Bound processes have been extracted from their provided runtime files. Such extraction was done through the Arena Input Analyzer, which provides which distribution is being used, and which values. For CPU Bound processes, a *NORM distribution (1000, 199) / 2000* was used, and for IO Bound processes, a distribution was used $(33 + (91 * BETA(1.94, 4.26))) / 2000$. The division present in both distributions is to perform the conversion from seconds to milliseconds.

After our final execution of the modeling, some values had to be changed due to limitations with respect to the time of simulation, for example we had to wait for 24h to simulate 1h, and limitations of the free version of the Arena, like the limitation of 150 units in the system. Such changes will be presented after explaining some points of the modeling.

4.3 Modeling

One decision we made was to divide it into two files, one for each algorithm as our solution grew, and it was difficult to transact between them. So we will detail them next.

4.3.1 Priority

An entity is created using a uniform distribution between 0,5 and 1, then its number of cycles is set, and then it is decided which type of process it is the entity, having 50% or 80% chance of being CPU Bound and the remaining processes will be Io Bound.

The priority of a process must be in the range -10 and 10. To determine the priority of a CPU Bound process a normal distribution of mean 0 and deviation 10 must be made, but there is a chance that a process will be created with a priority outside the range -10 and 10, making it necessary to verify its priority, and redetermine it if necessary. To determine the priority of an IO Bound process, a triangle with values -10, -5, and 5 was followed.

Next, the execution time of each process is set following its respective distributions, previously determined in the Input Analyzer, as explained in an earlier chapter.

The process is then placed in the Ready queue and then checked to see if it was previously preempted, if it was, its context is loaded. For statistical purposes, we store the time the process entered the Ready queue. When leaving the Ready queue, we save the process priority information so we can analyze it later. We set the process as Running and recorded the moment he started using the core.

Since we are working with a 4-core, we can have 4 processes running at the same time, and to model this behavior, we create a queue that contains the cores being used.

A process will exit the CPU when it receives a signal, such a signal is created at the end of the stream it initializes with `Create _TIMER_`, which starts a verification every 0.001 seconds that triggers a Search by a process that has already finished its execution time. If it is found, the core is deallocated, the process execution time is zeroed and then access to the device is made. The access time to the device varies according to a triangular distribution with parameters 10, 50, 100, and a division per 1000 was required to convert the unit of time to milliseconds. After access to the device, a cycle of its total is reduced, and if it does not have more cycles, it leaves the system, if not, new execution time is assigned to it and a new cycle is started.

If a process, that has already finished its execution time, is not found, we check if the Ready queue is empty, if it is not, we check if all cores are allocated, if they are not, there is no need to preempt. If they are, we store the best and worst priority which are currently running. If the first process in the Ready queue has a priority better than the worst running

priority, there is a need for preemption, so a signal is sent to preempt the worst priority process running. Another check is made if the priorities are equal, and if so, it is checked if the running process has already run longer than a quantum, and if so, it is preempted.

When a process is preempted, the core is deallocated, the time spent inside the cpu is subtracted from its execution time, then its context is saved and the process is put back into the ready queue.

As you can see, the only time a process left the system was when its cycles were over.

4.3.2 Round Robin

Parts of the modeling of the priority algorithm were reused. The definition of the number of cycles and type of process is the same, as well as the insertion of it in the Ready queue.

When the process exits the Ready queue, it checks whether it has been preempted before and loads its context, if any. If the remaining process execution time is less than the quantum, there will be a CPU time spent, so we chose to zero the execution time.

A delay with quantum time is used to simulate the CPU time, and in the end, the quantum is reduced from the process execution time. If the ready queue is empty, the process can run for another quantum without having to free the CPU, otherwise, the core is deallocated.

If its execution time is not yet finished, its context is saved. If it is finished, it accesses the device, with a time equal to the one set in the priority algorithm. A cycle is removed at the end of the access, and it is checked for more cycles. If there is, a new runtime is assigned and a new cycle starts, otherwise, the process exits the system.

4.4 Changes

As said in the course of this report, some changes were necessary so that we could achieve a sufficient number of replications, in an acceptable time, so that we could analyze the results.

Using the values initially proposed, the limitation of 150 units in the system was quickly overcome, and when we were able to find values that made the simulation run for a

few minutes, we realized that 1h of simulated time would take 1 day, so we decided to decrease the simulation time, so we could run all the necessary replications in 1 night.

The changes made are:

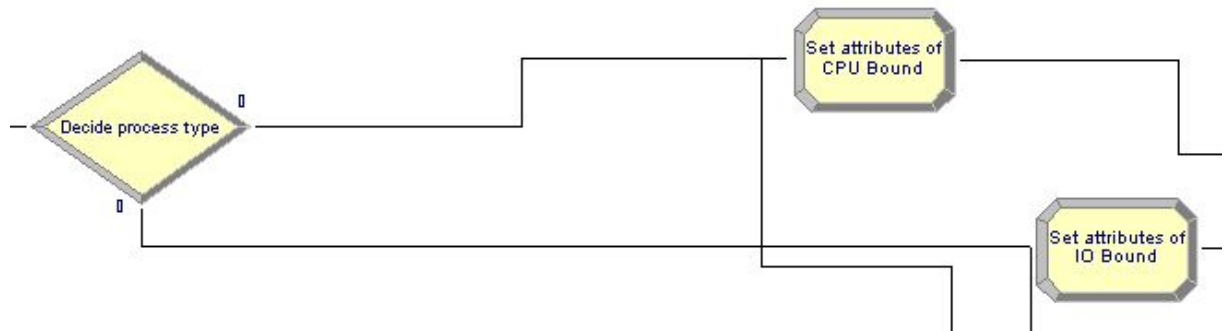
- Quantum: originally a quantum of 100us to 10ms was proposed, but we changed to be of 1ms to 10ms (minimum and maximum levels);
- Execution Times: We reduce both the execution time of CPU-bound processes and IO-bound processes by half;
- Cycles: originally varied uniformly between 5 and 25, but we changed to a uniform distribution 5 and 10;
- Simulation time: changed from 1h to 10minutes

5. VALIDATION AND VERIFICATION

Important sections of the model, analyzed in this step.

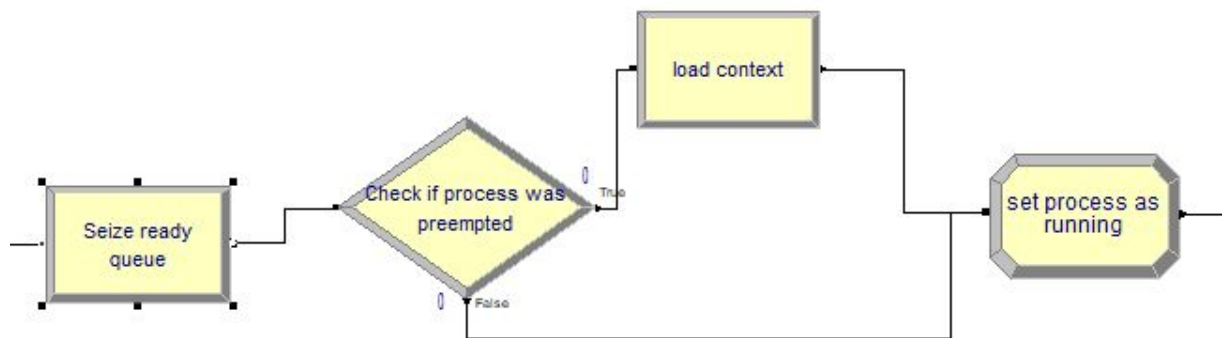
5.1 Round Robin Algorithm Model

I. Workload

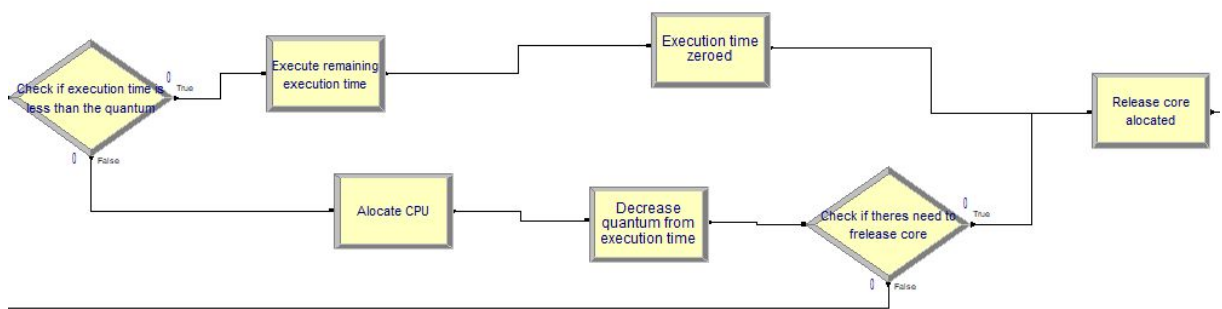


A decide component uses the global variable, CPU-bound chance, choosing the process type, being CPU bound or IO bound, which led to their respective assigns. These set Process Type 1 to CPU and 0 to IO bounds, the entities also have an attribute called Process State with 0 to Ready and 1 to Running state.

ii. Cores

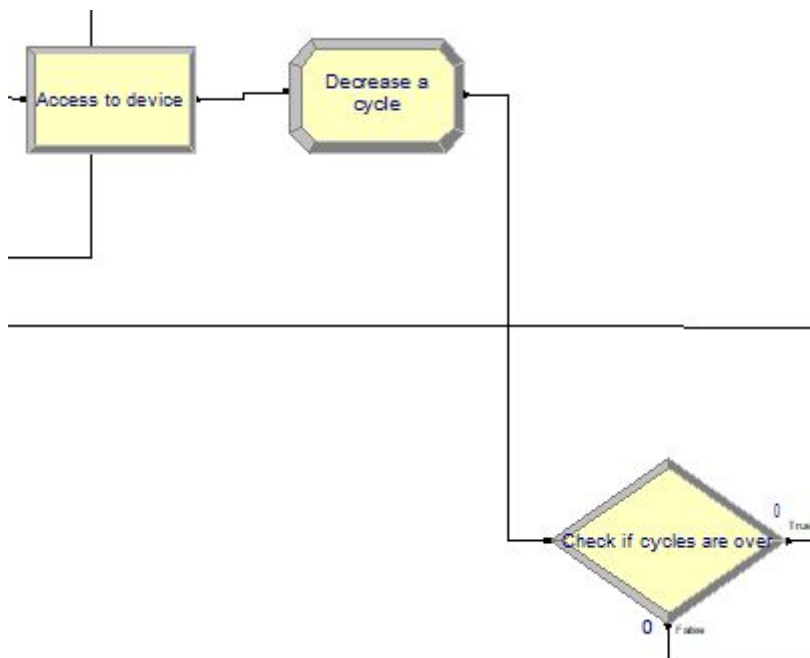


A sub-model represents the CPU utilization, at start a Seize block receive the entities and allocates up to 4 resources, which represents the 4 cores in a Set, the exceeding entities stay at the Ready.Queue. The first 4 are set to the Running State.



Two Delays are used to hold the process with the meaning of a process being executed by the processor. The upper part is used when the remaining execution time is less than the quantum, otherwise spends the quantum time.

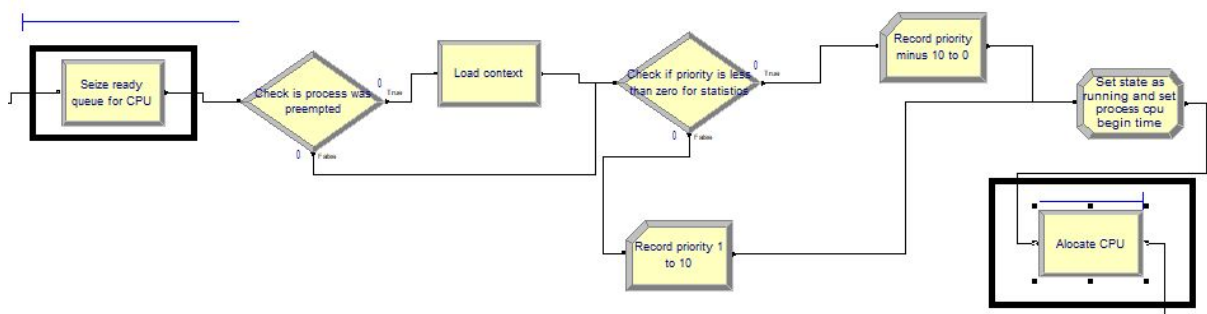
iii. Cycles



In case the execution time is over, the process accesses the device, an Assign decreases a cycle, if the cycles are over, the entity goes to the end, else it will return to the model receiving new values to the new cycle.

5.2 Priority Algorithm Model

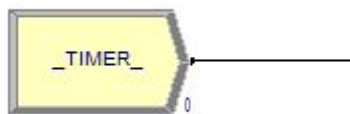
1. Cpu



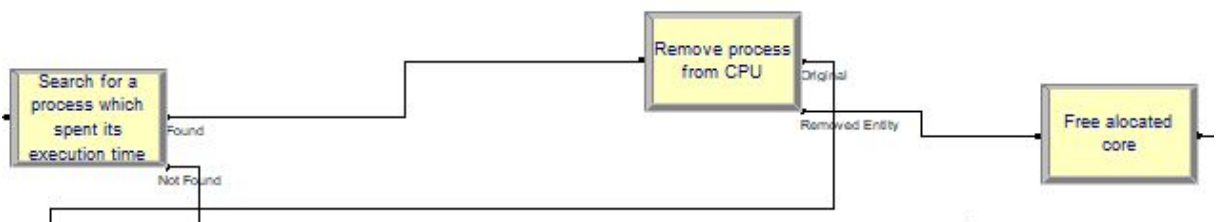
In this section, the first component is a Seize component to represent the queue of processes in Ready state, the second marked in the image is a *Hold* with processes allocating the Cores the others in between are there for registry purposes.

This Hold utilizes a queue ordered by the priority of the processes, this way we can send a signal to remove the first entity if we know another process needs the Cpu.

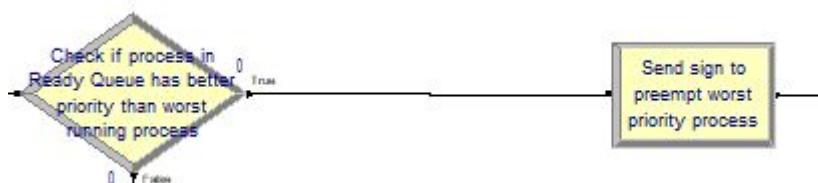
ii. Interruptions



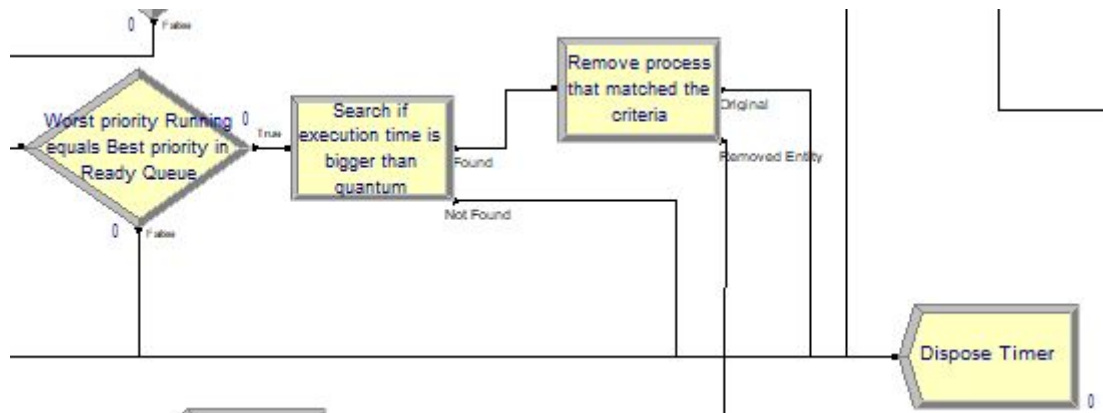
Creates an entity that will be used to check if an interruption it will be needed by priority or quantum time, it generates at a frequency of 100 microseconds.



Search and Remove, to find in the CPU if a process already finished its time, the entity is removed from the Hold component and led to a Release.



The same timer checks if a process with more priority is waiting, then the timer entity sends a signal to the Hold, releasing the entity who represents a process with the worst priority.



In case of false, it checks if the priorities are the same, then uses the Round Robin method.

5.3 Validation

With the experiment results below we can empiric validate the system.

	Scenario Properties				Controls			Responses							
	S	Name	Program File	Reps	CPU Bound chance	quantum	Num Reps	Core 1.Utilization	Core 2.Utilization	Core 3.Utilization	Core 4.Utilization	Ready.Queue .WaitingTime	Ready.Queue NumberInQue	CPU Bound Process.Total	IO Bound Process.Total
1		Scenario 1	65 : PRIO.p	90	50	0.010	90	0.8575	0.7822	0.6201	0.4050	0.0096	0.1901	4.4357	0.6423
2		Scenario 2	65 : PRIO.p	90	50	0.001	90	0.8580	0.7851	0.6260	0.4066	0.0068	0.1922	4.4380	0.6409
3		Scenario 3	65 : PRIO.p	90	80	0.010	90	0.9962	0.9953	0.9932	0.9881	0.0613	12.9316	10.6034	0.6853
4		Scenario 4	65 : PRIO.p	90	80	0.001	90	0.9970	0.9963	0.9947	0.9911	0.0366	13.9505	10.7680	0.6822
5		Scenario 5	65 : RR.p	90	50	0.010	90	0.8577	0.7814	0.6182	0.3931	0.0031	0.1802	4.3982	0.6456
6		Scenario 6	65 : RR.p	90	50	0.001	90	0.8571	0.7787	0.6159	0.3971	0.0004	0.1913	4.4197	0.6448
7		Scenario 7	65 : RR.p	90	80	0.010	90	0.9961	0.9953	0.9933	0.9881	0.0444	17.4712	20.0199	1.8094
8		Scenario 8	65 : RR.p	90	80	0.001	90	0.9972	0.9965	0.9951	0.9913	0.0051	19.8354	22.2100	1.8136

When the model has more CPU bound processes the Cpu utilization goes near to full capacity, and the waiting queue has more processes and average time in general. This makes sense since the Cpu bound process has more execution time than the IO bound process.

6.1 Experimental Project

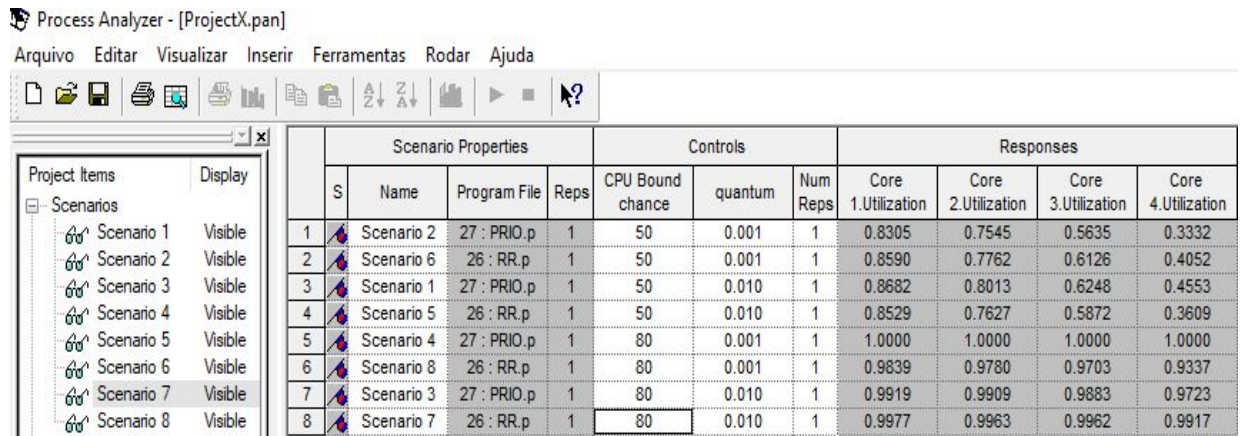
A 2^k factorial design was used to calculate the impact of the variables.

Foi realizado um projeto fatorial 2^k para calcular o impacto das variáveis.

The next picture shows the process for determining the CPU rate using the Arena Process Analyzer tools, a *X.pan* was generated. These values were used in Design Expert software to calculate the factor impact (*ImpactoVariaveis.dx7* file).

Process Analyzer - [ProjectX.pan]

Arquivo Editar Visualizar Inserir Ferramentas Rodar Ajuda



Scenario Properties				Controls			Responses			
S	Name	Program File	Reps	CPU Bound chance	quantum	Num Reps	Core 1.Utilization	Core 2.Utilization	Core 3.Utilization	Core 4.Utilization
1	Scenario 2	27 : PRIO.p	1	50	0.001	1	0.8305	0.7545	0.5635	0.3332
2	Scenario 6	26 : RR.p	1	50	0.001	1	0.8590	0.7762	0.6126	0.4052
3	Scenario 1	27 : PRIO.p	1	50	0.010	1	0.8682	0.8013	0.6248	0.4553
4	Scenario 5	26 : RR.p	1	50	0.010	1	0.8529	0.7627	0.5872	0.3609
5	Scenario 4	27 : PRIO.p	1	80	0.001	1	1.0000	1.0000	1.0000	1.0000
6	Scenario 8	26 : RR.p	1	80	0.001	1	0.9839	0.9780	0.9703	0.9337
7	Scenario 3	27 : PRIO.p	1	80	0.010	1	0.9919	0.9909	0.9883	0.9723
8	Scenario 7	26 : RR.p	1	80	0.010	1	0.9977	0.9963	0.9962	0.9917

It is observed the great impact of the probability of the CPU_Bound process with 98,62%. Individually is followed by Quantum with 0,19% and the algorithm with 0,041% however, the second parameter of the greatest impact is the combination of the three factors with 0,96%.

Term	Stdized Effects	Sum of Squares	% Contribution
Intercept			
A-Algorithmo	-6.788E-003	9.214E-005	0.041
B-Quantum	-0.015	4.366E-004	0.19
C-CPU-bound Chance	0.33	0.22	98.62
AB	0.011	2.616E-004	0.12
AC	-5.162E-003	5.330E-005	0.024
BC	7.350E-003	1.080E-004	0.048
ABC	-0.033	2.180E-003	0.96
Lenth's ME	0.053		
Lenth's SME	0.13		

6.2 Response 1 CPU Rate

ANOVA for selected factorial model

Analysis of variance table [Partial sum of squares - Type III]

Source	Sum of Squares	df	Mean Square	F Value	p-value Prob > F	
Model	0,223708	3	0,074569	114,6058	0.0002	significant
A-Algoritmo	9,21E-05	1	9,21E-05	0,141611	0.7258	
B-Quantum	0,000437	1	0,000437	0,671015	0.4587	
C-CPU-bound Chance	0,223179	1	0,223179	343,0049	< 0.0001	
Residual	0,002603	4	0,000651			
Cor Total	0,22631	7				

The Model F-value of 114.61 implies the model is significant. There is only a 0.02% chance that a "Model F-Value" this large could occur due to noise.

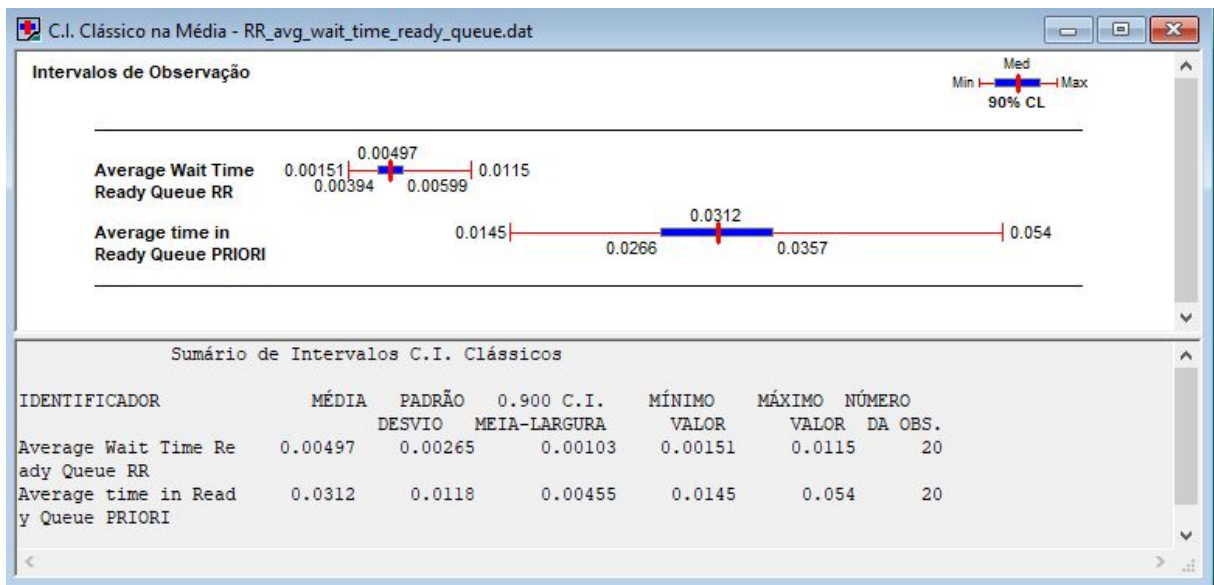
7. ANALYSIS OF RESULTS

One of the questions we want to know is: is it possible to say with 90% certainty if the average waiting time in the Ready queue is different for the two algorithms?

For this experiment we used variables that leave the system with a bigger workload:

- Quantum = 0.001s
- CPU Bound chance = 80%

The results are imported into the Output Analyzer to calculate the Confidence Interval. The first experiment was done with 20 replications, and these were the results:



- As we can see, the Priority Algorithm generated an interval [0.0266; 0.0357] with an average of 0.0312s. The range limit is not within the target of 10% of the mean [0.02808; 0.03432];
- The Round Robin algorithm generated an interval [0.00394; 0.00599] with an average of 0.00497s. The limits also are not in the range of 10% of the mean [0.00473; 0.00547]

So we will calculate the number of replications that will be required for this confidence interval to stay at 10% of the mean.

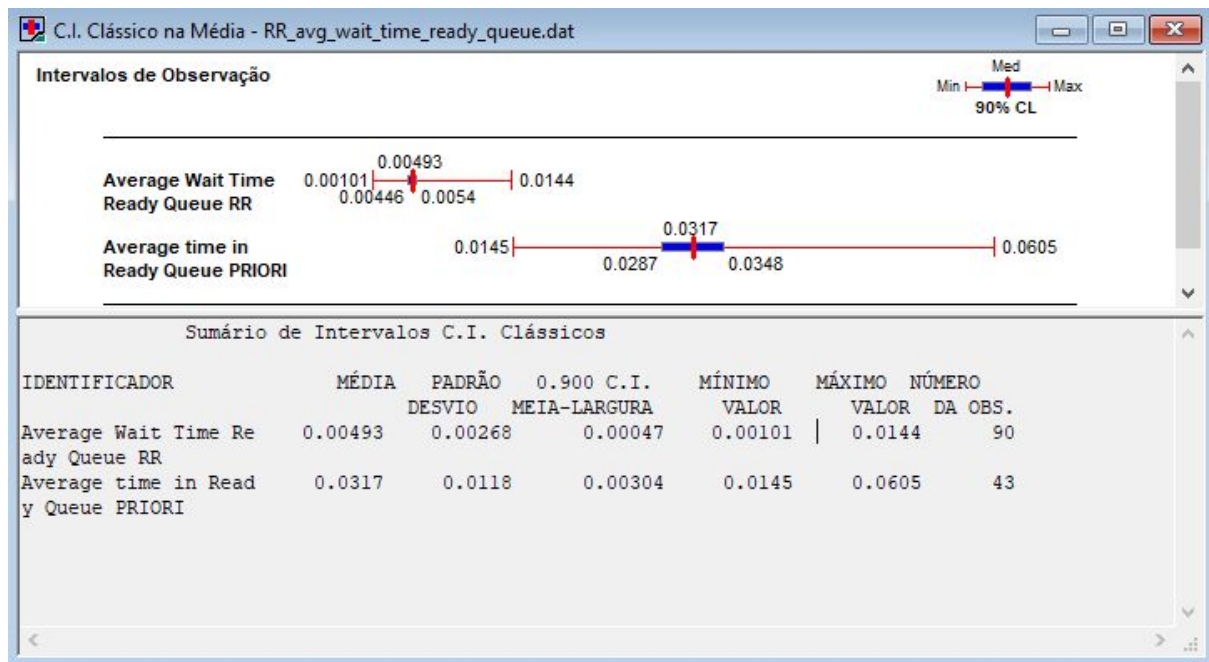
Calculating Sample Size required for Algorithm Priority:

- T-value for 90% confidence and degree of freedom 19: 1,729
- Error $E = 10\% * \text{mean} = 0,0312$
- $s = 0,0118$
- Sample size: $n = t^2 * s^2 / E^2$
- $n = 42,767$, so we need 43 replications.

Calculating Sample Size required for Algorithm Round Robin:

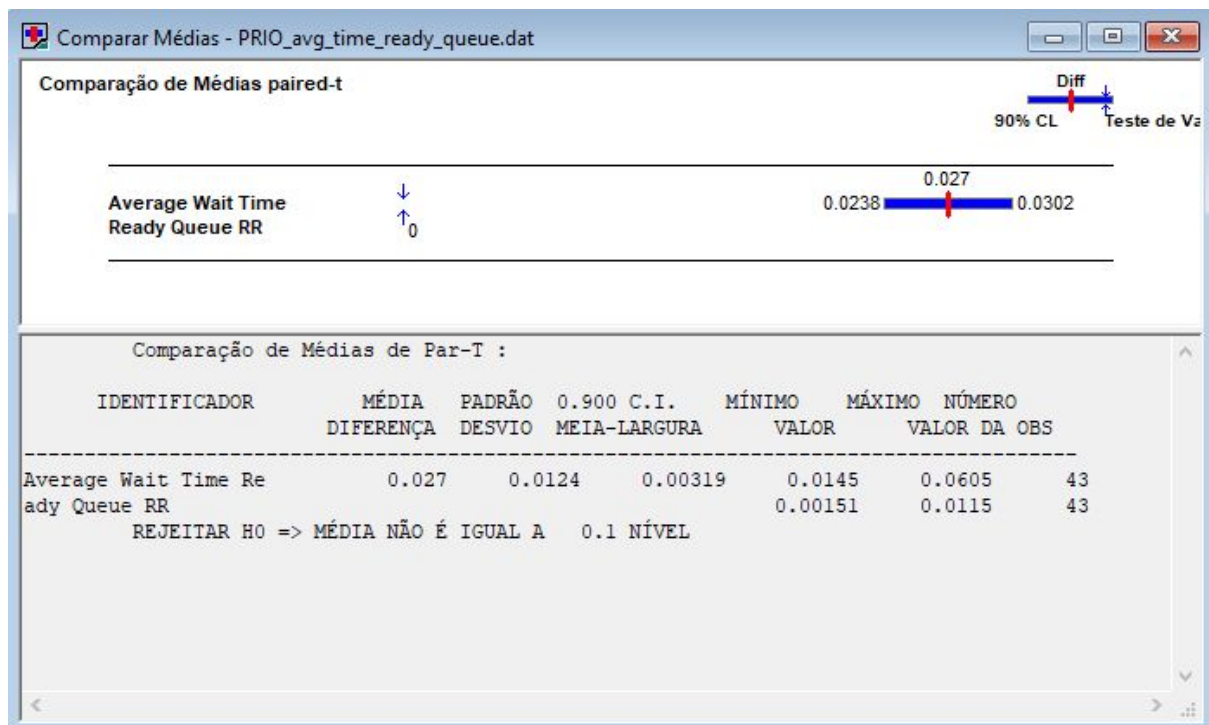
- T-value for 90% confidence and degree of freedom 19: 1,729
- Error $E = 10\% * \text{mean} = 0,000497$
- $s = 0,00265$
- Sample size: $n = t^2 * s^2 / E^2$
- $n = 84,9902$ so we need 85 replications.

With the number of replications required, we re-tested the 43 and 90 replicate tests, and this is the result:



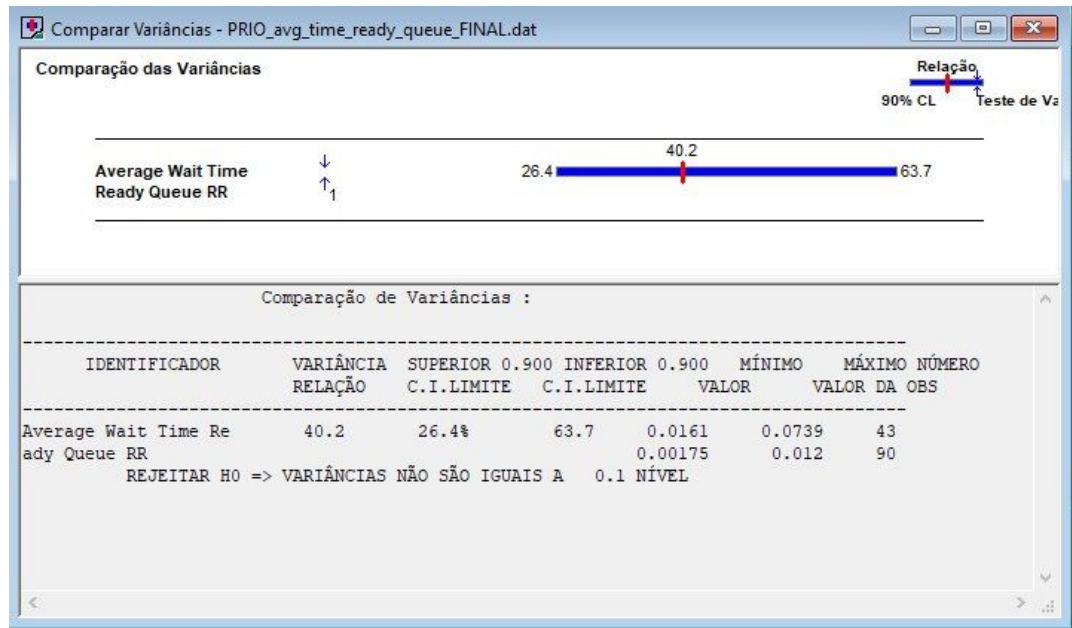
Now, both are within the range of 10% of the mean. So now we can compare the means:

- Null hypothesis: The means of the two algorithms are the same.
- Alternative hypothesis: The means are different.



We can observe that the Null Hypothesis was rejected, that is, it can not be said with 90% confidence that the average time in the queues are the same.

We can check also that the Variances are different.



Another question to be answered is: **do processes with a priority of -10 to 0 stay less time in the Ready queue?**

First experiment with 20 replications, variables:

- quantum = 0.001s
- Cpu-bound chance = 80%

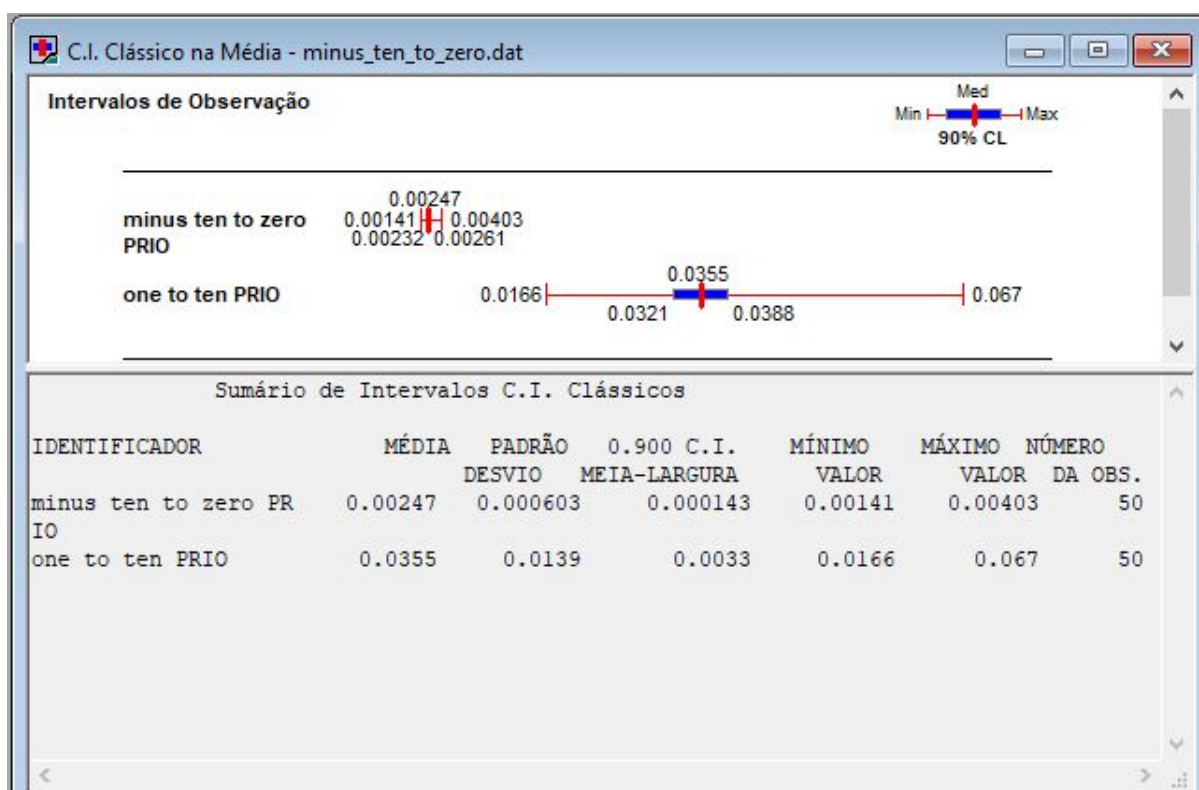
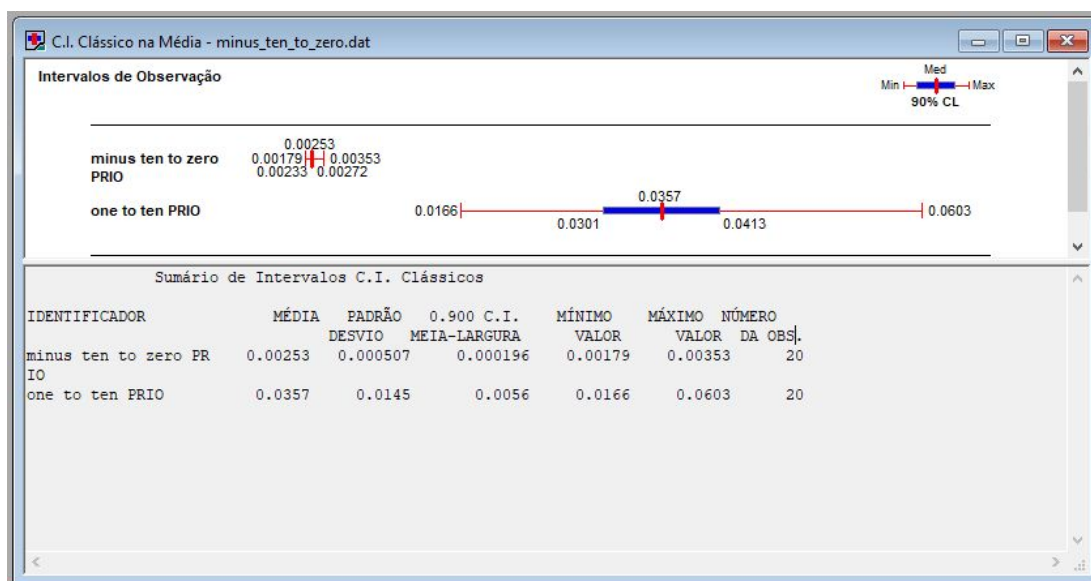
The confidence range of the priority processes -10 to 0, are within the 10% margin of the average. But those 1 to 10 are not.

The upper limit for the mean 0.0357 is 0.0392, and the experiment resulted in an upper limit of 0.0413.

So we calculate the required size sample:

- T-value for 90% confidence and degree of freedom 19: 1,729
- Error E = 10% * mean = 0,00357
- s = 0,0145
- Sample size: $n = t^2 * s^2 / E^2$
- n = 49,31619 so we need 50 replications.

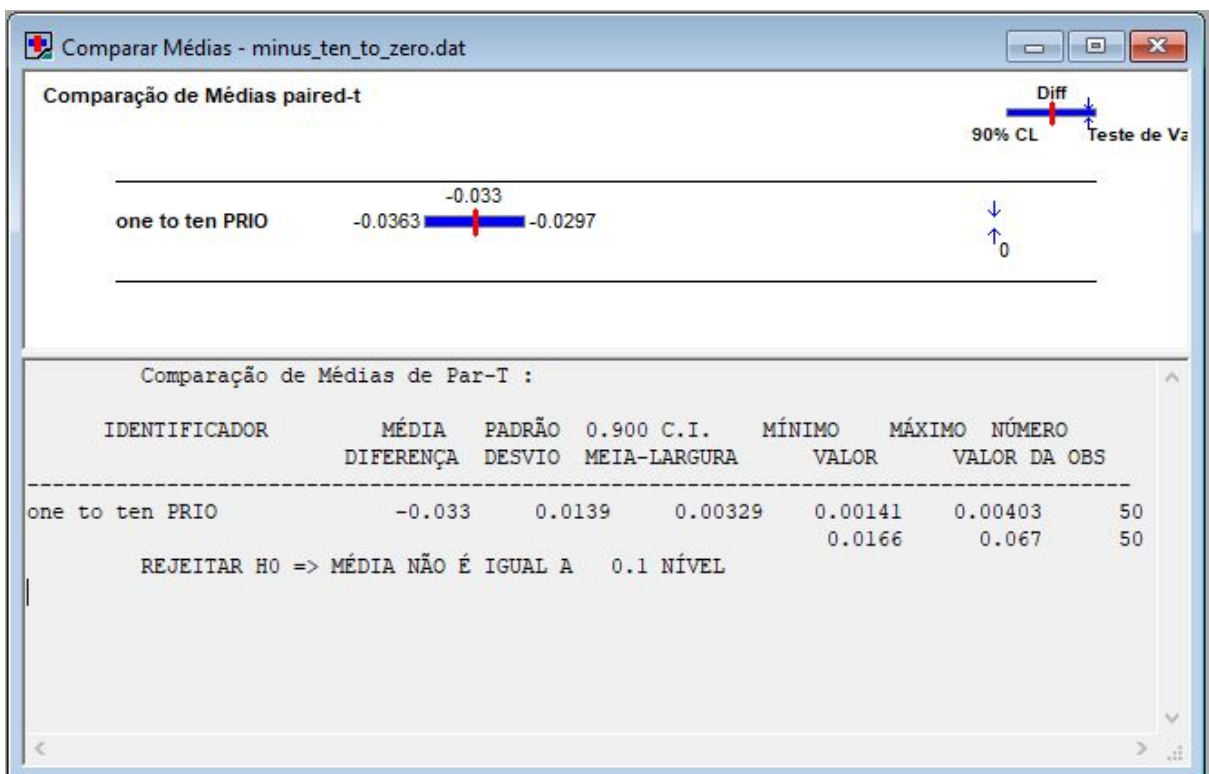
New values for 50 replications:



	Priorities -10 a 0	Priorities 1 a 10
Range of 10%	[0,00222; 0,00272]	[0,03195; 0,03905]
Confidence Interval	[0,00232; 0,00261]	[0,0321; 0,0388]

As we can see, both are within the range of 10% of the mean. So now we can compare the means:

- Null hypothesis: The means of the two algorithms are the same.
- Alternative hypothesis: The means are different.



With 90% confidence, we can not say that the means are the same.

8. Results Presentation

As a final step, we replicate 90x each scenario of our simulation. The choice of such amount was determined earlier by calculating the sample size for a 90% confidence interval of the mean.

Since there are 8 scenarios, with 90 replications each, a processing night was necessary, which resulted in the following image:

Scenario Properties			Controls			Responses									
Name	Program File	Reps	CPU Bound chance	quantum	Num Reps	Core 1.Utilization	Core 2.Utilization	Core 3.Utilization	Core 4.Utilization	Ready.Queue.WaitingTime	Ready.Queue.NumberInQue	CPU Bound Process.Total	IO Bound Process.Total	CPU Bound Process.Tran	IO Bound Process.Tran
Scenario 1	65 : PRIO.p	90	50	0.010	90	0.8575	0.7822	0.6201	0.4050	0.0096	0.1901	4.4357	0.6423	0.0000	0.0000
Scenario 2	65 : PRIO.p	90	50	0.001	90	0.8580	0.7851	0.6260	0.4066	0.0068	0.1922	4.4380	0.6409	0.0000	0.0000
Scenario 3	65 : PRIO.p	90	80	0.010	90	0.9962	0.9953	0.9932	0.9881	0.0613	12.9316	10.6034	0.6853	0.0000	0.0000
Scenario 4	65 : PRIO.p	90	80	0.001	90	0.9970	0.9963	0.9947	0.9911	0.0366	13.9505	10.7680	0.6822	0.0001	0.0000
Scenario 5	65 : RR.p	90	50	0.010	90	0.8577	0.7814	0.6182	0.3931	0.0031	0.1802	4.3982	0.6456	0.0000	0.0000
Scenario 6	65 : RR.p	90	50	0.001	90	0.8571	0.7787	0.6159	0.3971	0.0004	0.1913	4.4197	0.6448	0.0000	0.0000
Scenario 7	65 : RR.p	90	80	0.010	90	0.9961	0.9953	0.9933	0.9881	0.0444	17.4712	20.0199	1.8094	0.0000	0.0000
Scenario 8	65 : RR.p	90	80	0.001	90	0.9972	0.9965	0.9951	0.9913	0.0051	19.8354	22.2100	1.8136	0.0000	0.0000

In it we can see which algorithm was used, the CPU Bound chance and the quantum value. It is possible to realize that when the CPU Bound chance is 80%, the 4 cores are used more than 98% of the time, regardless of the algorithm. But by reducing the CPU Bound chance to 50%, we see a drop in CPU utilization, reaching the extreme where Core 1 is used 2x more than Core 4. Such a phenomenon is probably explained because Core 1 is being released before the next cores can be allocated, and because we use a queue to store the cores, we always try to allocate the lowest number first. Such data can be checked in columns 1 to 4 in Responses

Both the waiting time and the number of processes in the ready queue is significantly lower when the Bound Chance CPU is 50%, reinforcing an idea in the previous paragraph that the Cores are managing to finish executing the processes before new ones arrive. Such data can be checked in columns 5 and 6 of Responses.

In columns 7 and 8 of Responses, we can see the average time a process of each type stays in the system. IO Bound processes stay, on average, less time on the system in any comparison made with CPU Bound processes.

With respect to the context switch overhead, illustrated in the remaining columns, the only moment it was slightly relevant was in the Priority algorithm scenario, with CPU Bound

chance 80% and quantum of 0.001s. Even though it was superior to other scenarios, its impact was minimal.

Now comparing the two algorithms in equivalent scenarios, we can see that the use of cores was almost the same, but the Round Robin algorithm takes advantage in process waiting time in the Ready queue. Now when CPU Bound chance is 50%, both algorithms are equivalent to the time a process stays in the system, but when we change to 80%, the processing time in the system doubles in the Round Robin algorithm compared to Priority