Sistemas Operacionais
EP1 - setembro/2016

...

Carlos Augusto Motta de Lima - 7991228
Danilo Aleixo Gomes de Souza - 7972370

Arquitetura do Shell

- O Shell implementado possui dois comandos embutidos: **chmod** e **id** (que deve ser chamado com o parâmetro -u).

- Em ambos casos, a implementação foi feita usando chamadas de sistema, o comportamento apresentado é similar aos comandos de mesmo nome no **bash**

Arquitetura do Shell

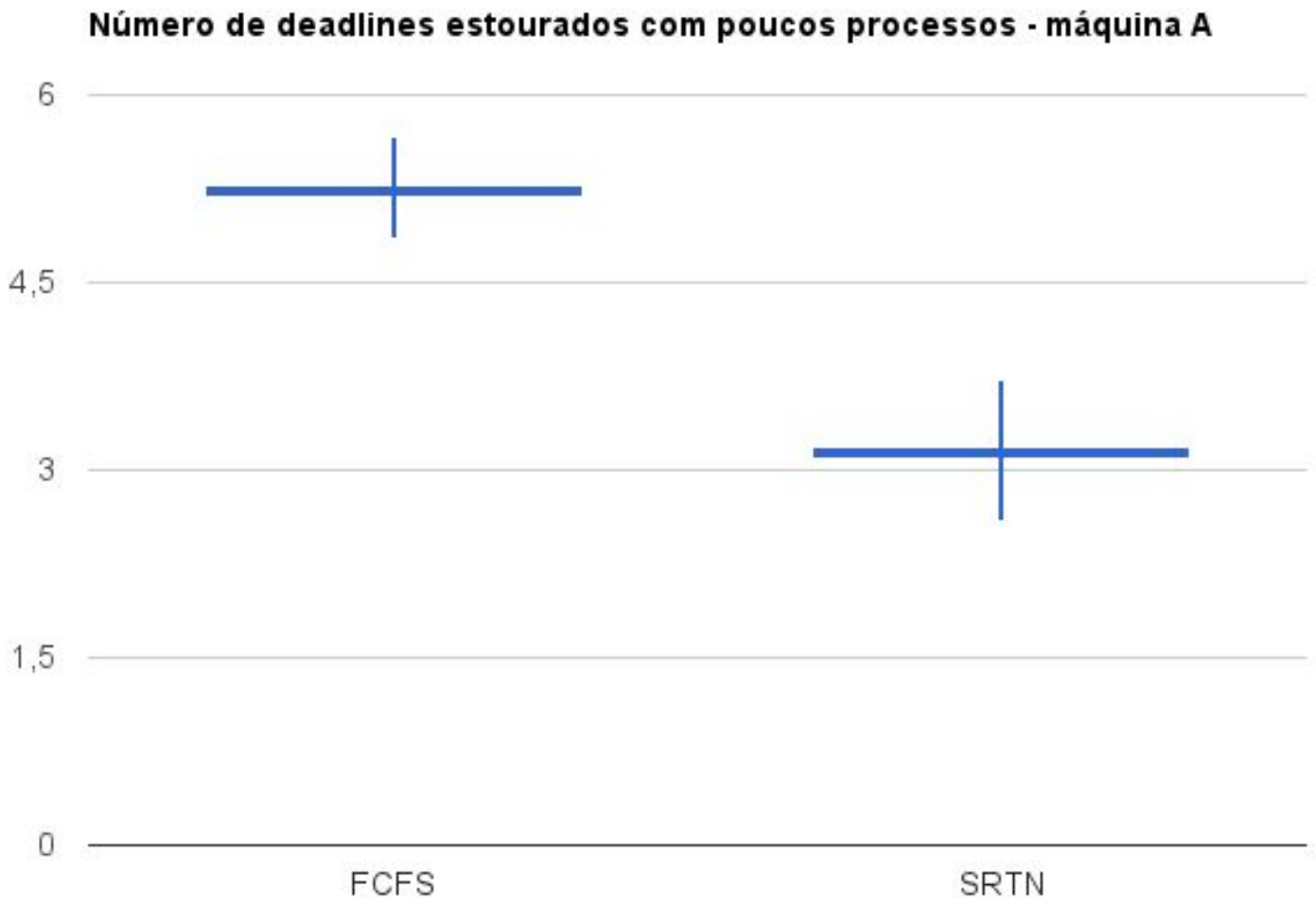
- O Shell também é capaz de invocar arquivos executáveis (em especial **/bin/ls -l**, **/bin/date** e **./ep1**, conforme especificado no enunciado. No entanto, ele é capaz de chamar também outros executáveis)

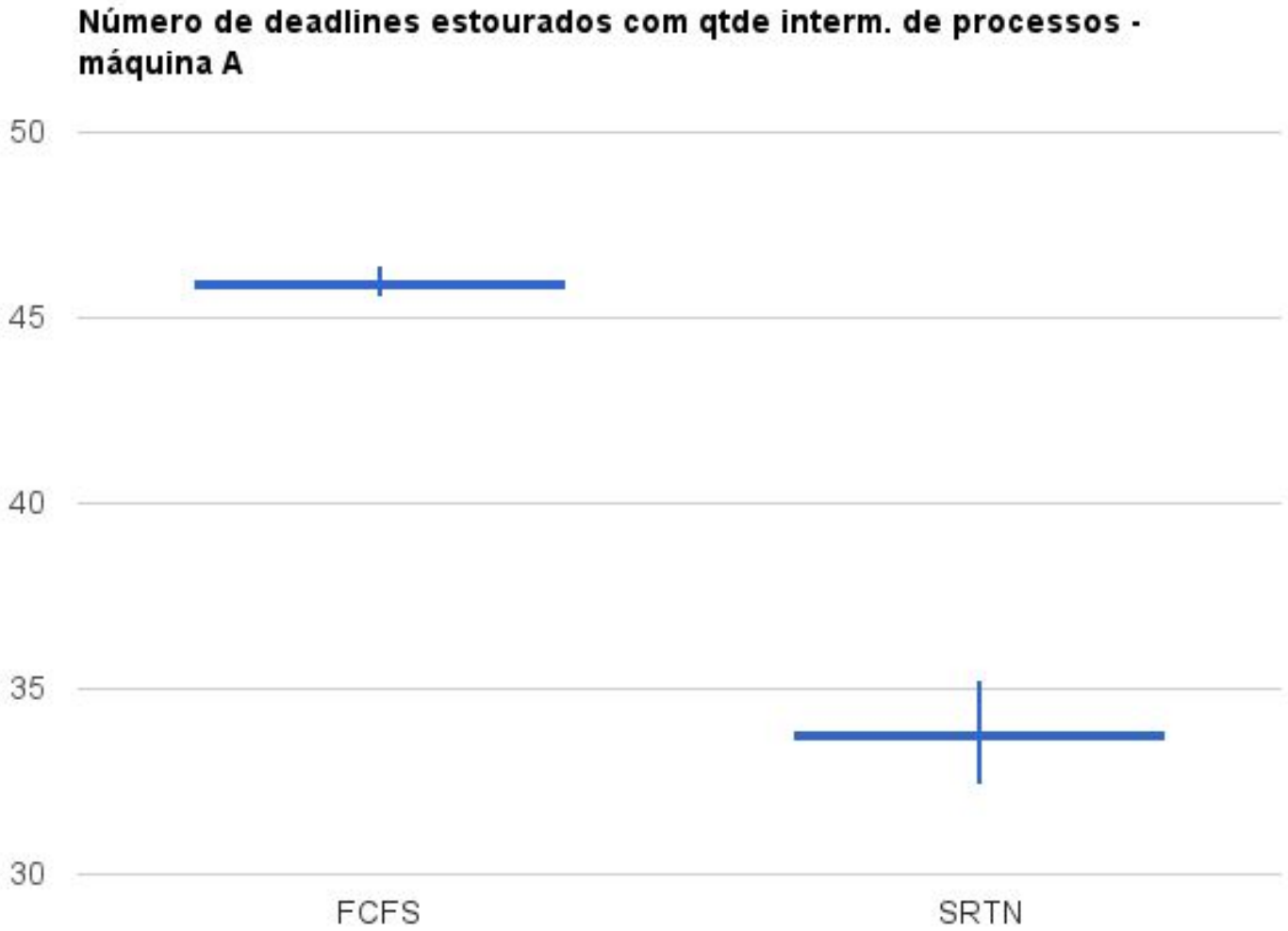
- Quando invoca um executável, o Shell cria um processo filho para rodá-lo e espera que o filho termine a execução

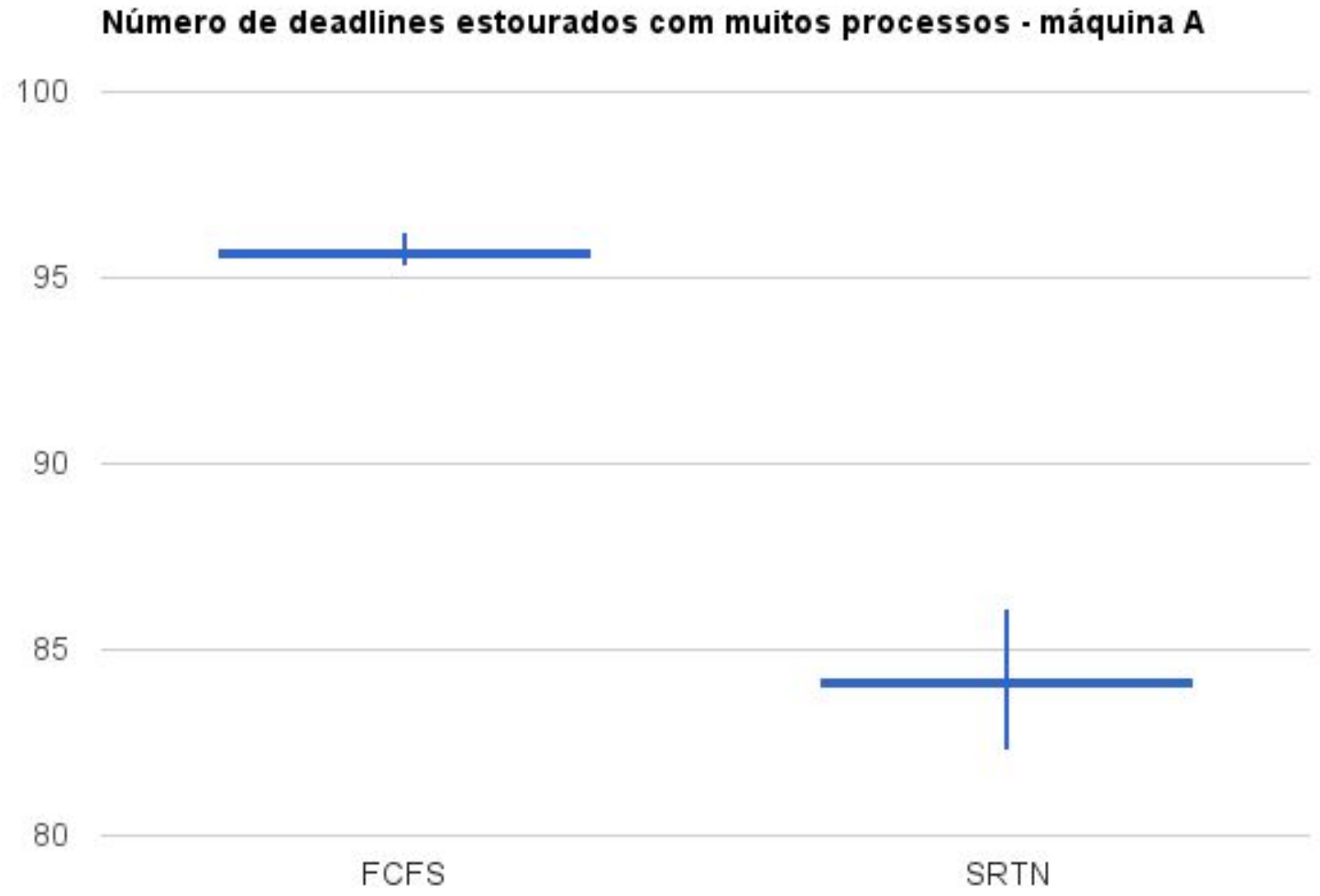
Experimentos

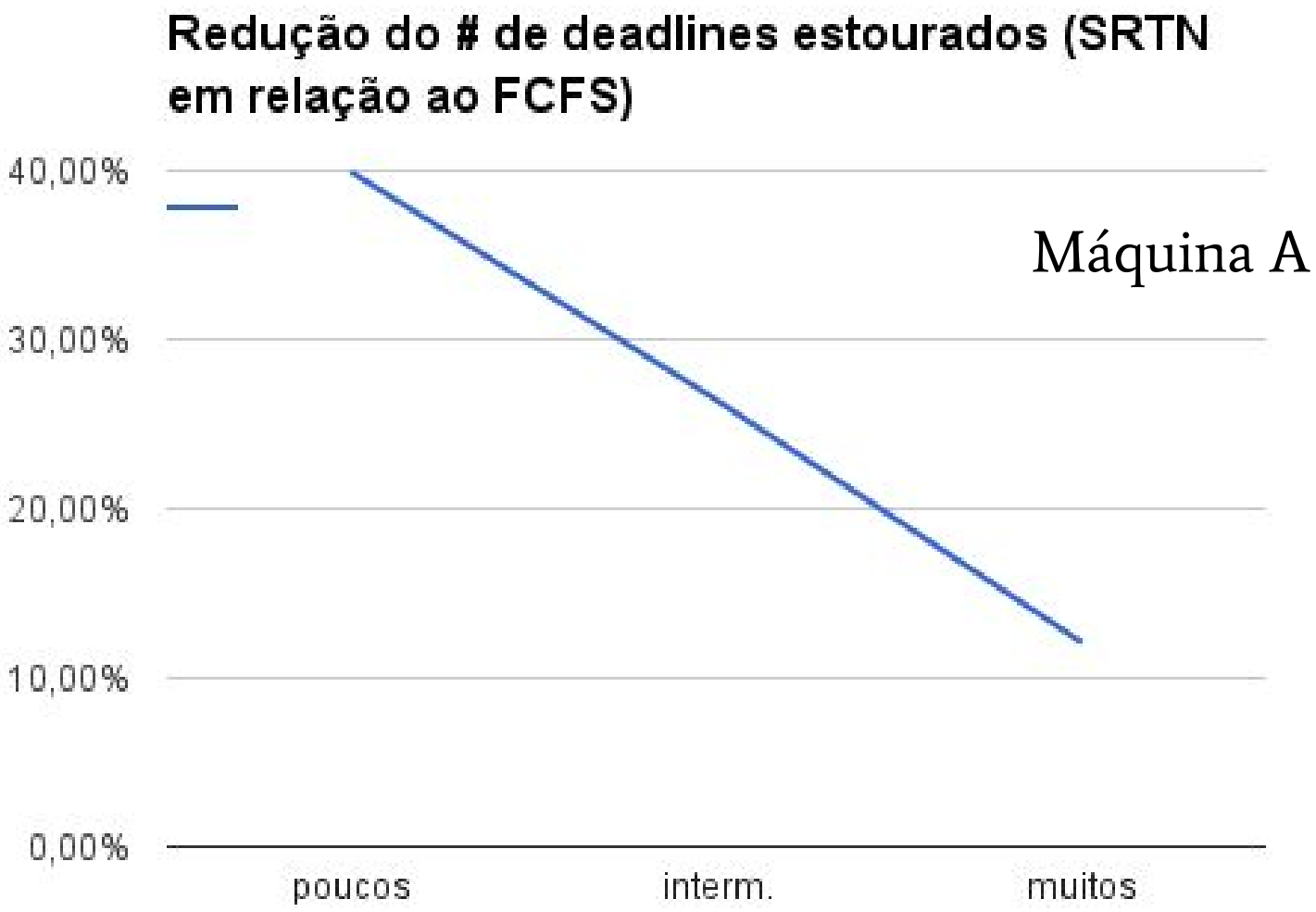
- Para a realização dos experimentos, se utilizou duas máquinas: uma com 4 processadores (máquina A) e outra com apenas 1 (máquina B).

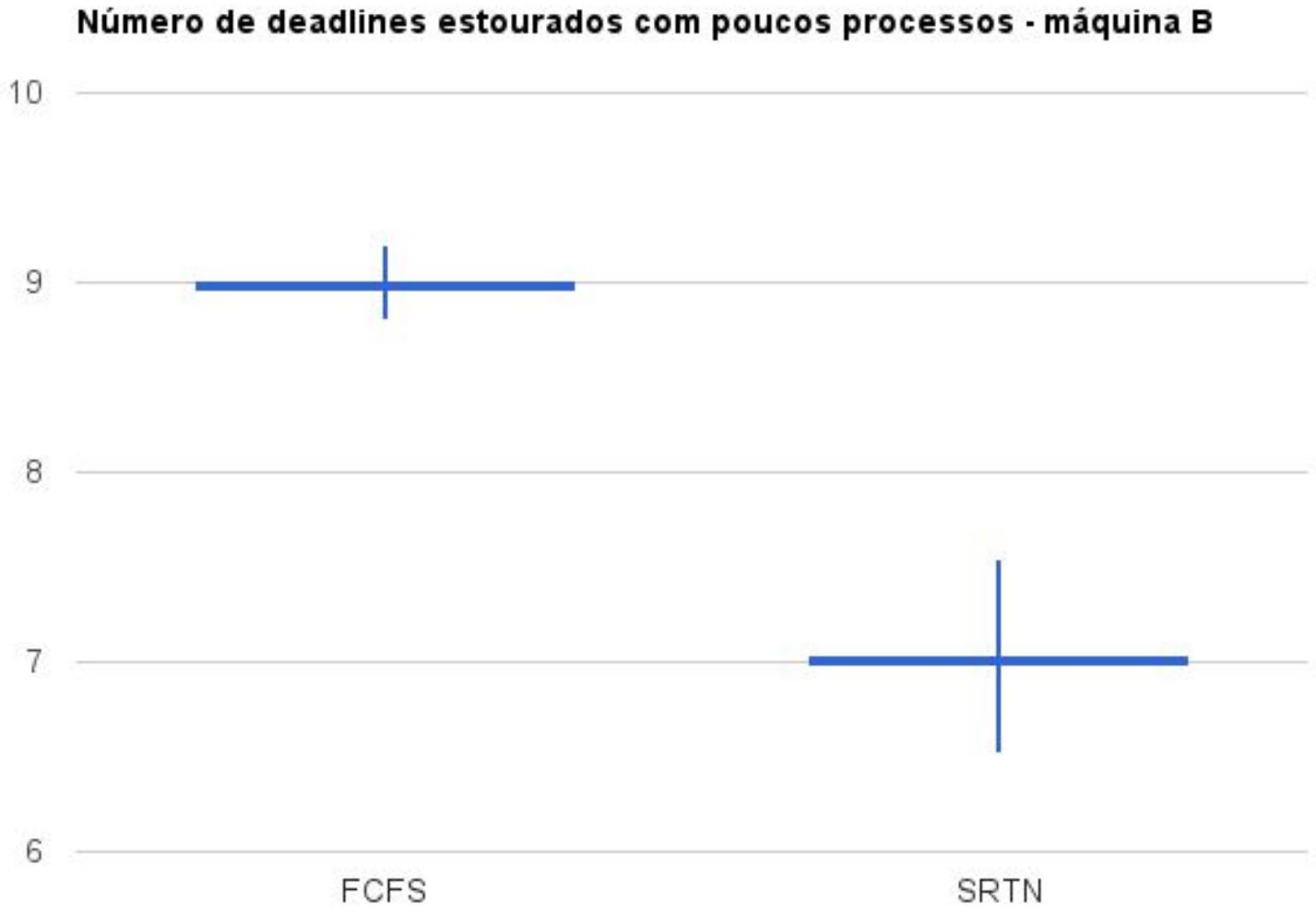
- Quanto ao número de processos em cada teste, considerou-se 10 como sendo “poucos processos”, 50 como sendo “número intermediário de processos” e 100 como “muitos processos”

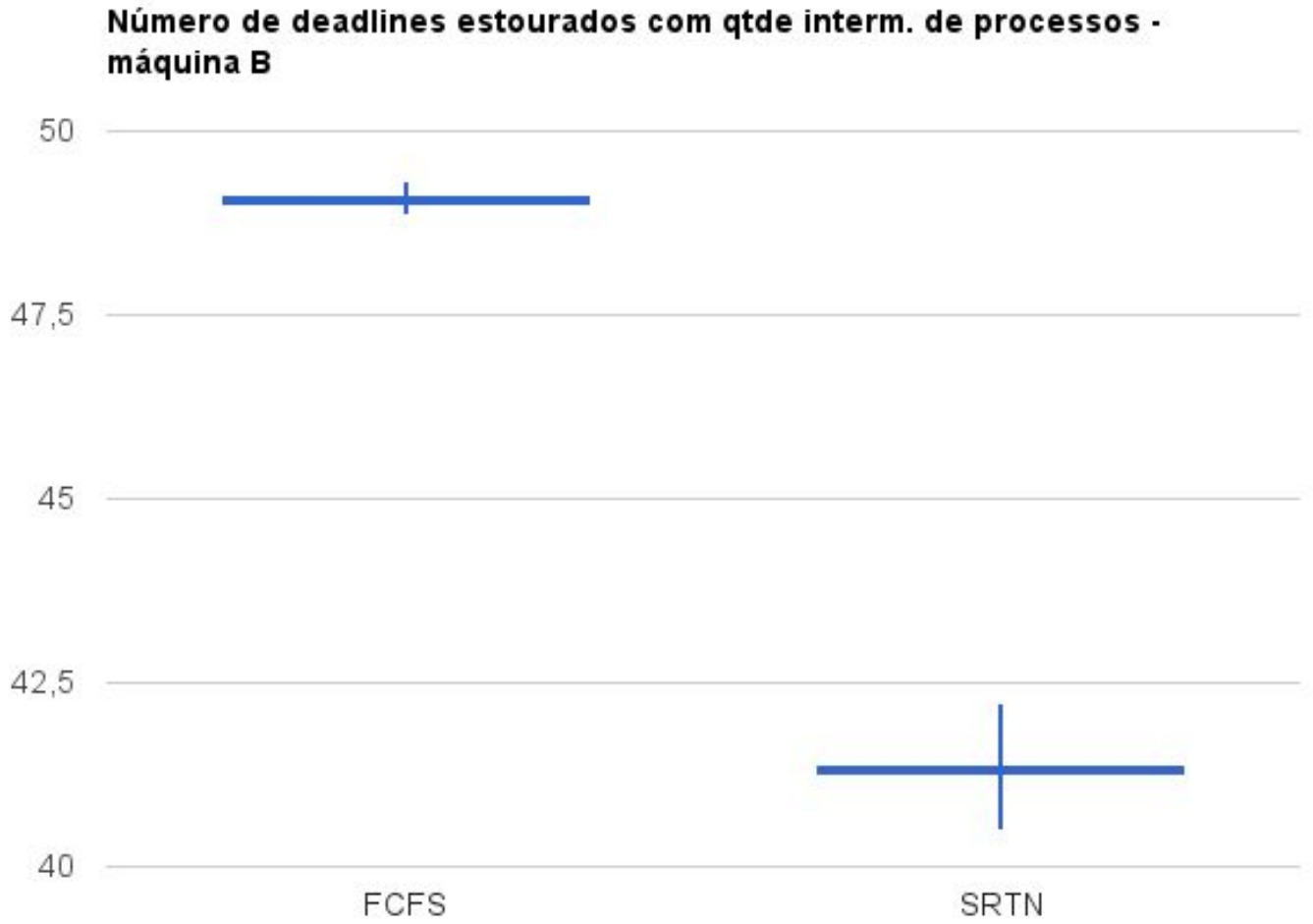

Número de deadlines estourados com poucos processos - máquina A
6
4,5
3
1,5
0
FCFS
SRTN

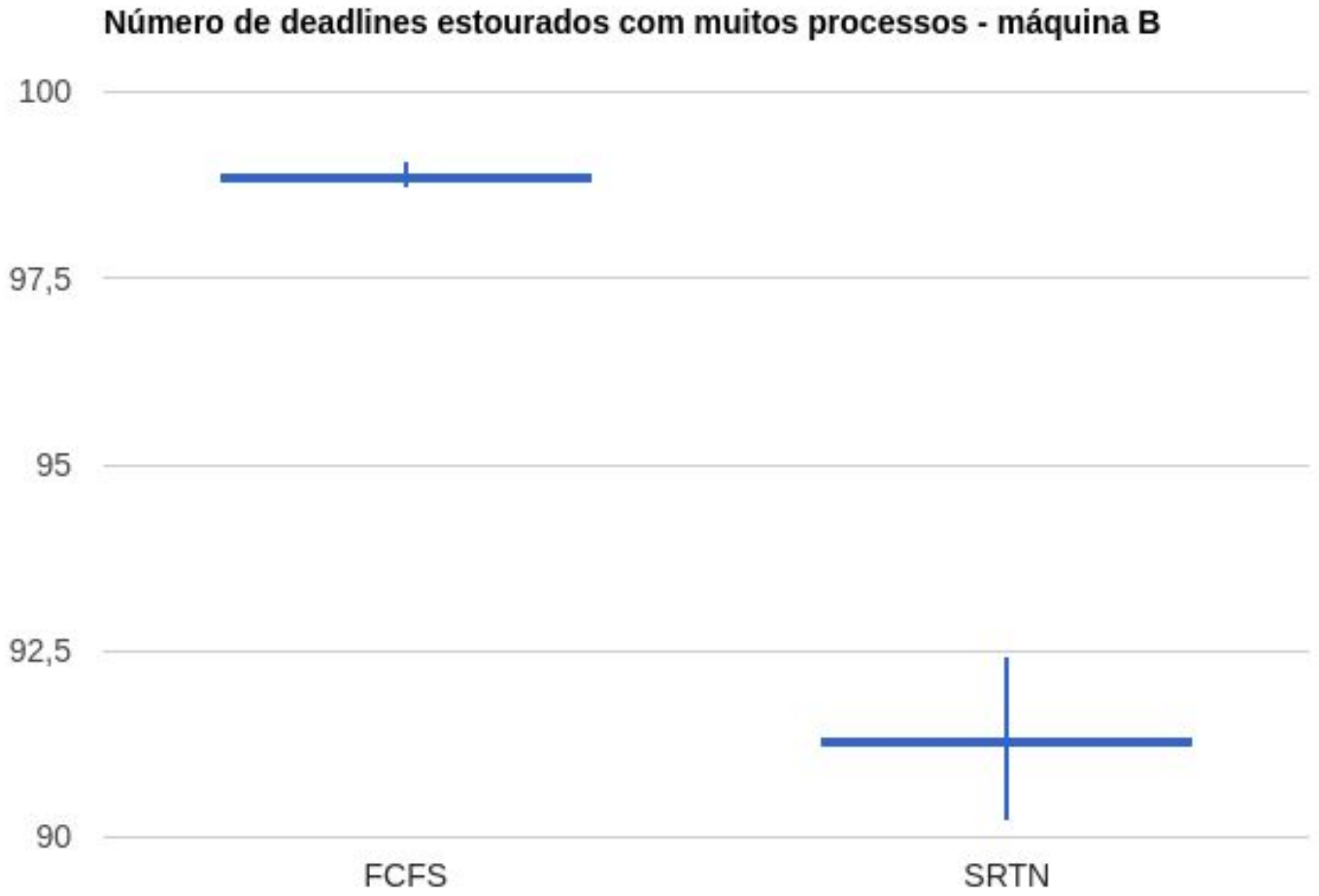

Número de deadlines estourados com qtde interm. de processos - máquina A
50
45
40
35
30
FCFS
SRTN


Número de deadlines estourados com muitos processos - máquina A
100
95
90
85
80
FCFS
SRTN

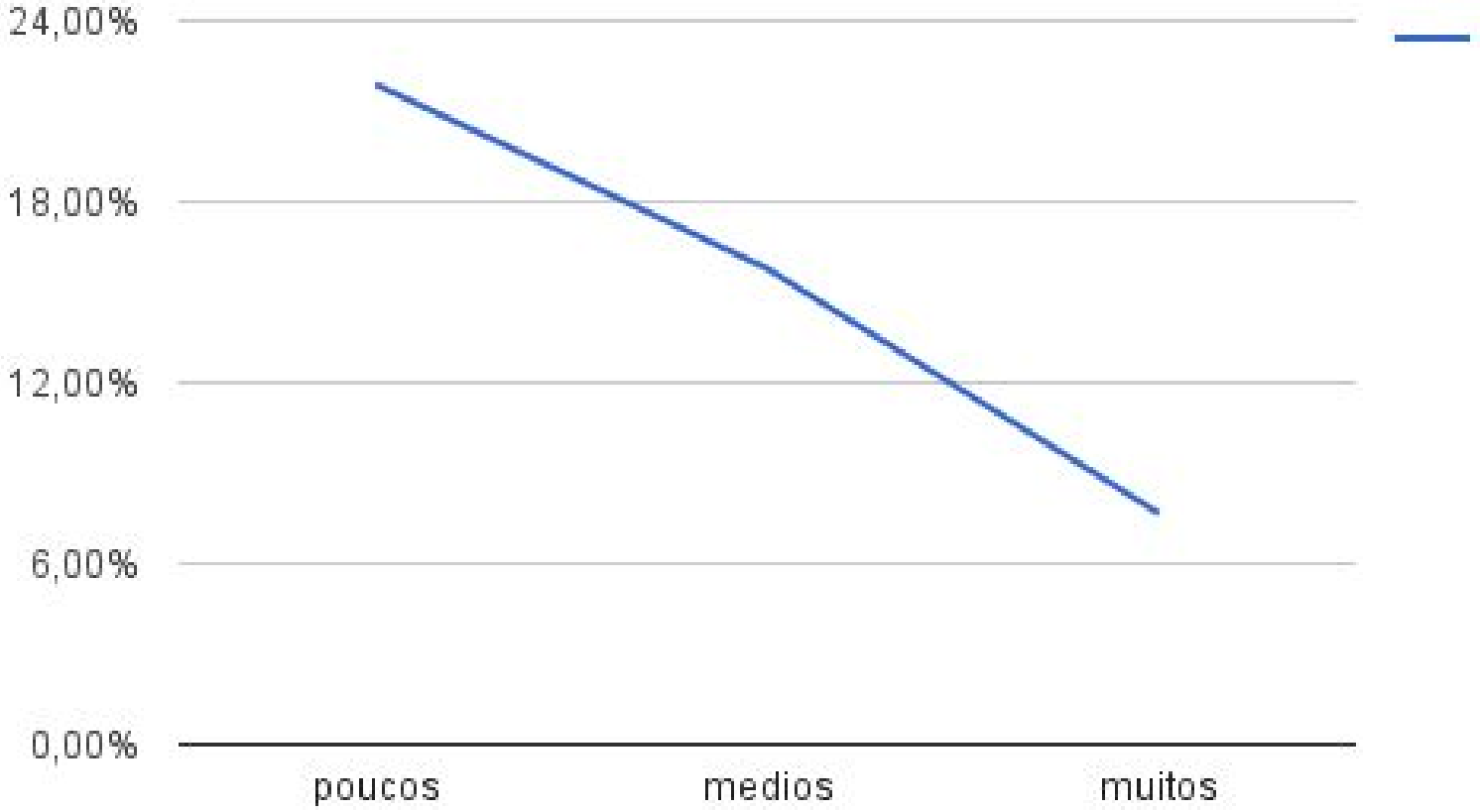

Redução do # de deadlines estourados (SRTN em relação ao FCFS)
40,00%
30,00%
20,00%
10,00%
0,00%
poucos
interm.
muitos
Máquina A

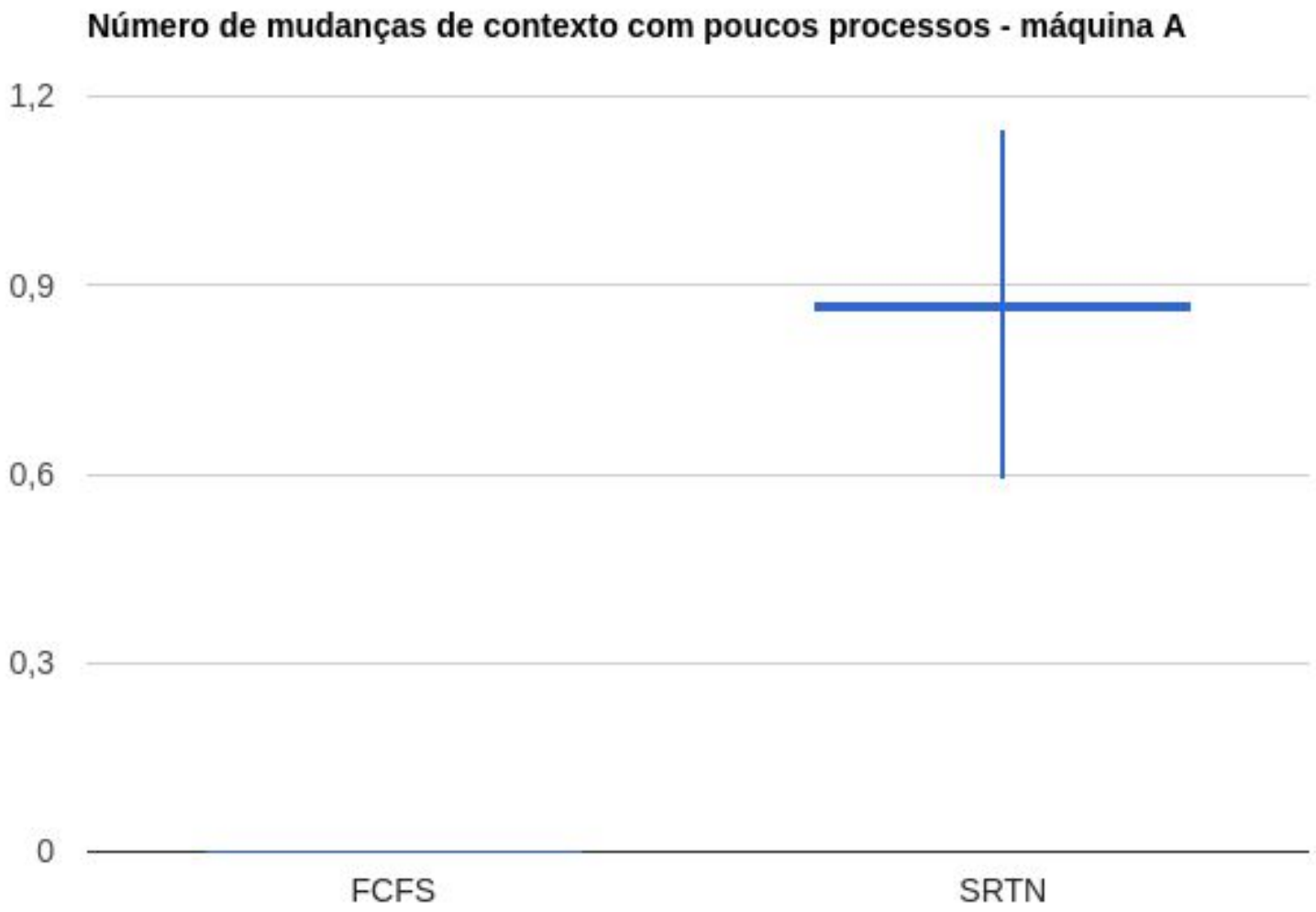

Número de deadlines estourados com poucos processos - máquina B
10
9
8
7
6
FCFS
SRTN

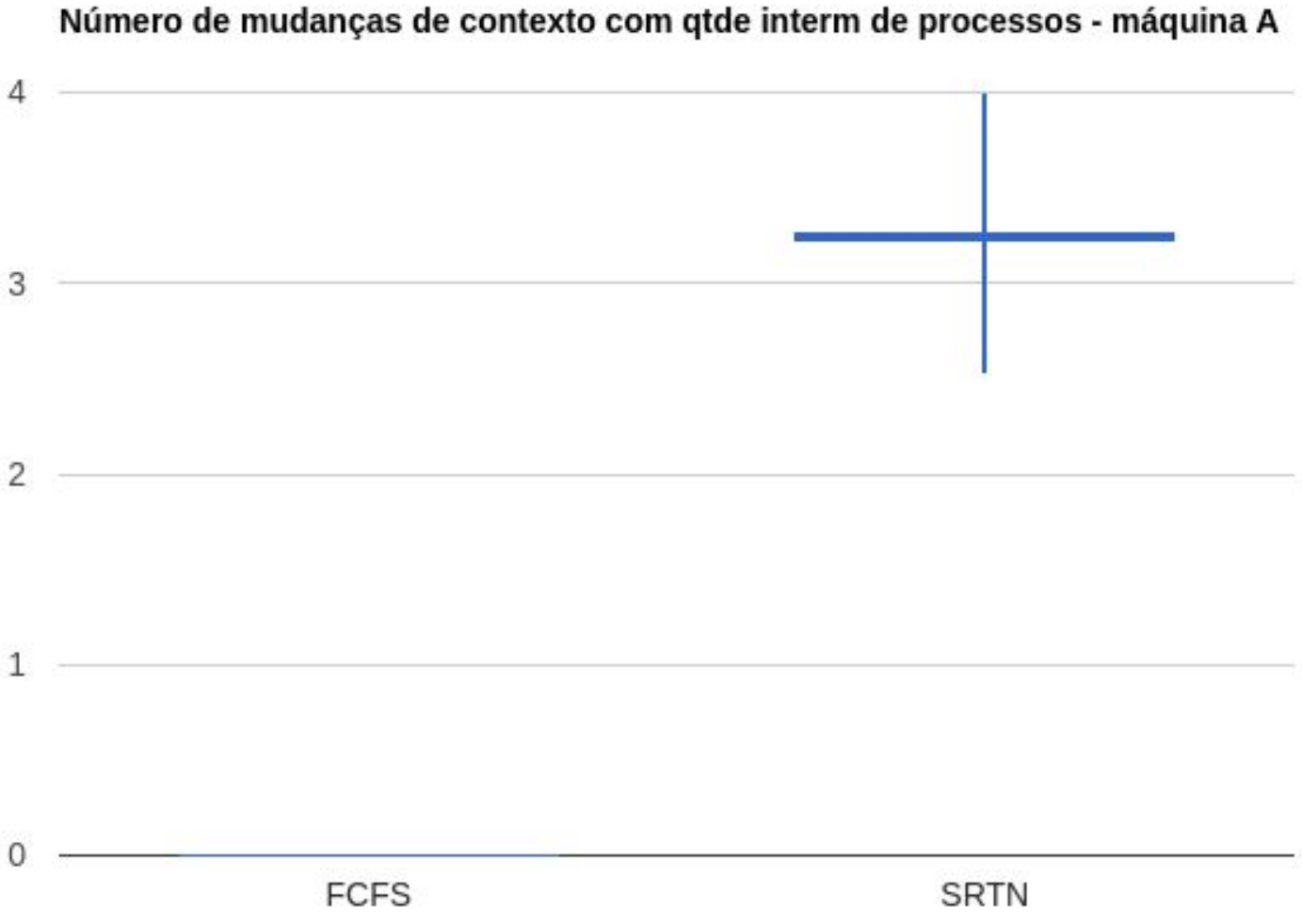

Número de deadlines estourados com qtde interm. de processos - máquina B
50
47,5
45
42,5
40
FCFS
SRTN

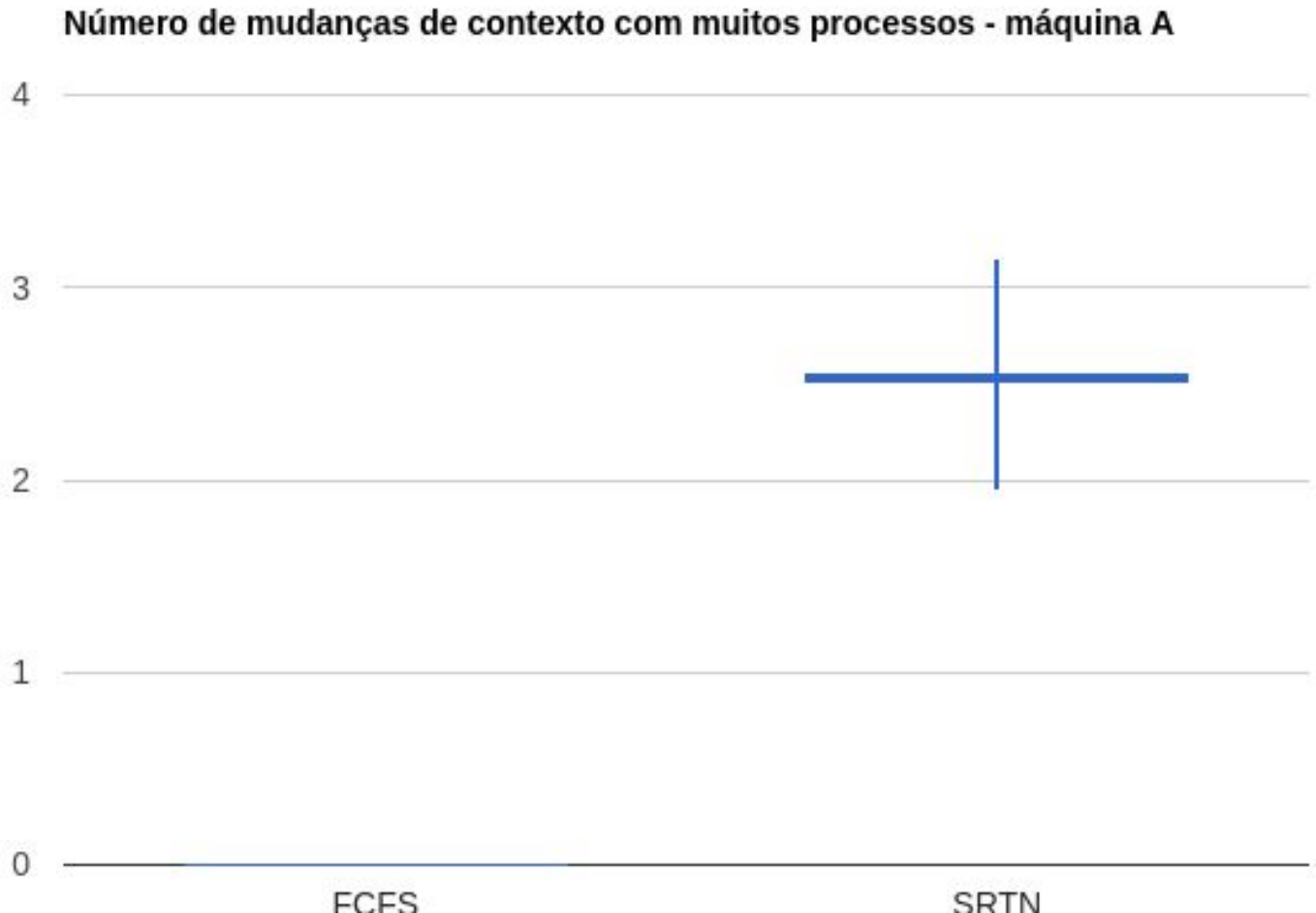

Número de deadlines estourados com muitos processos - máquina B
100
97,5
95
92,5
90
FCFS
SRTN

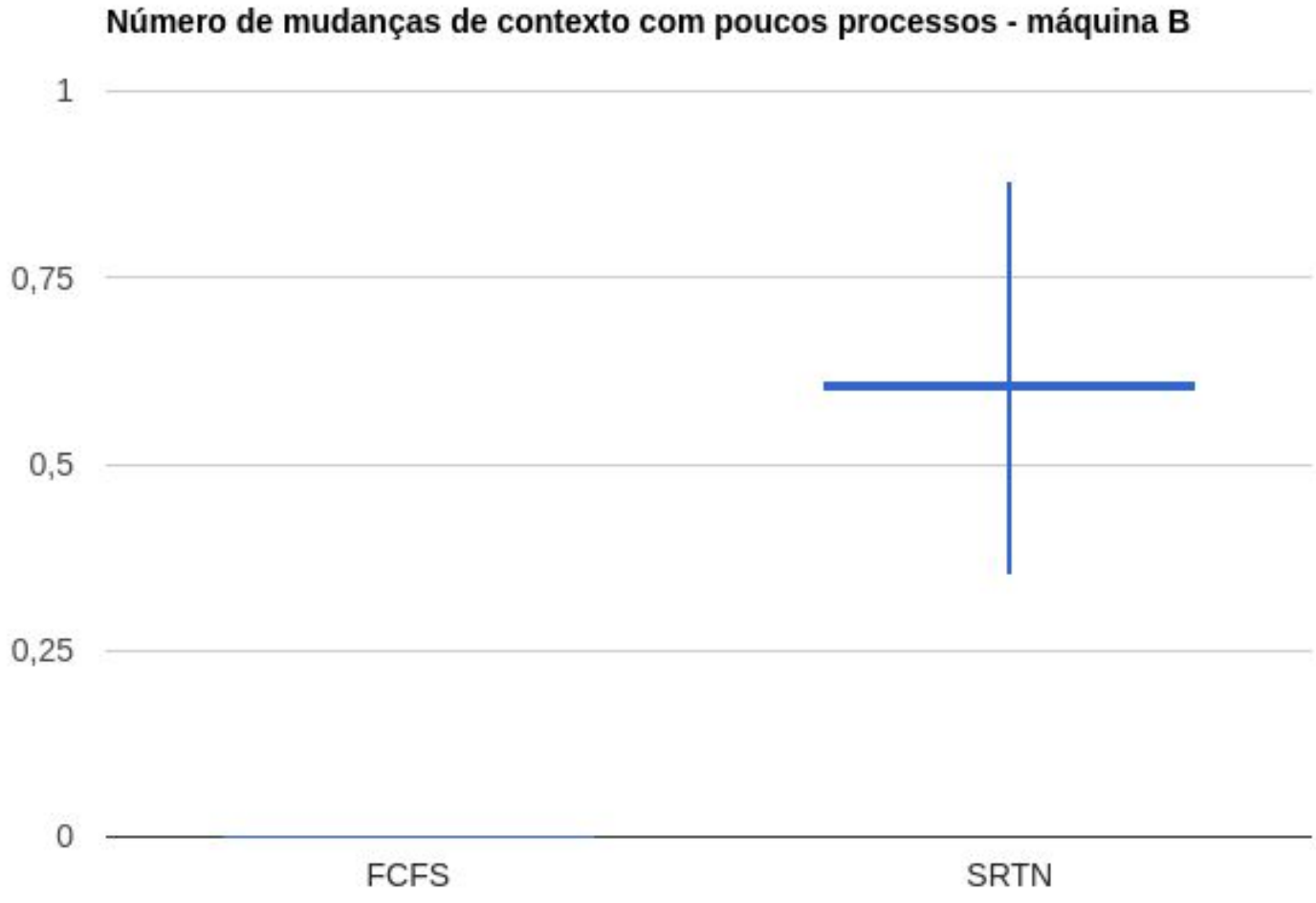
Redução do # de deadlines estourados (SRTN em relação ao FCFS)

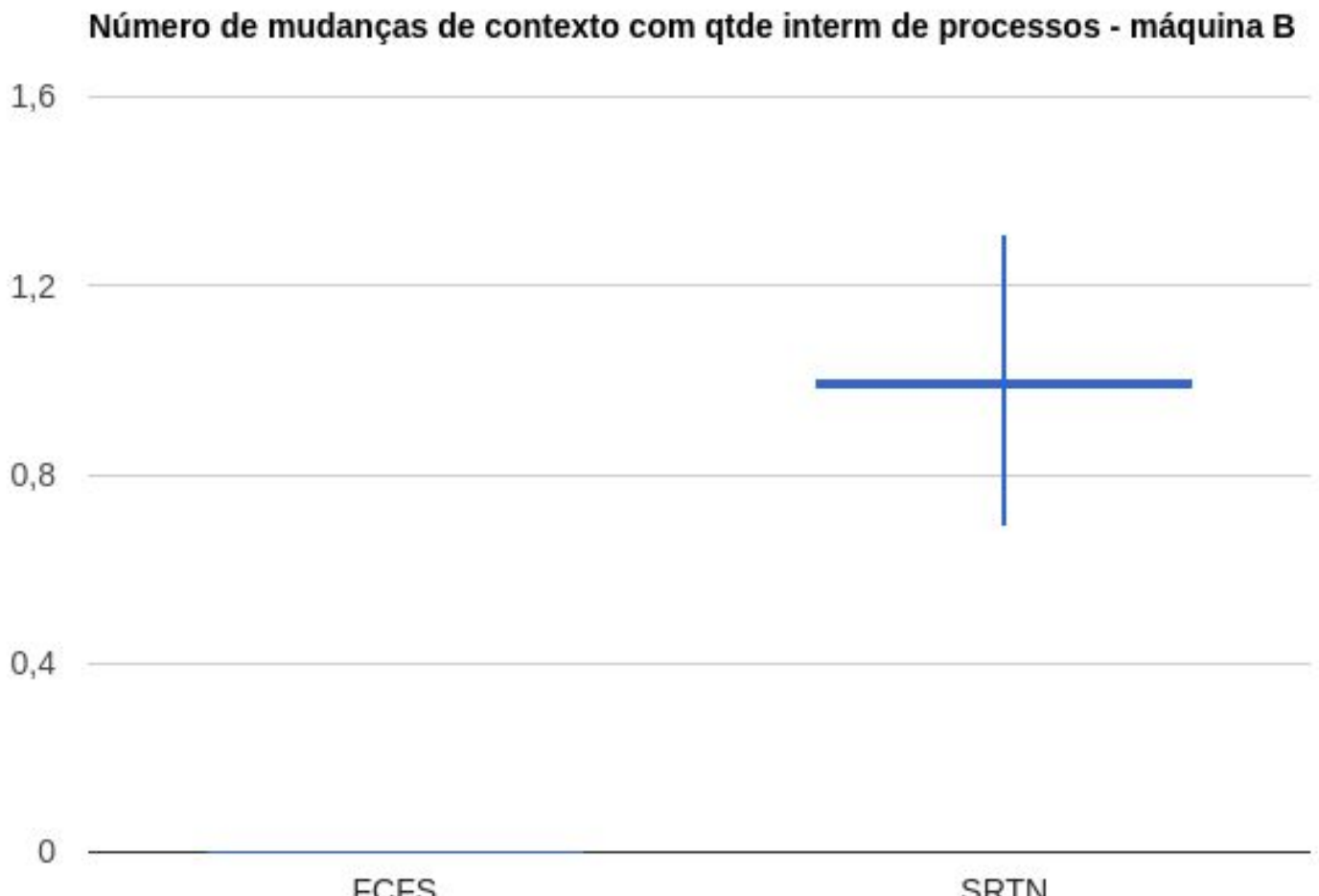


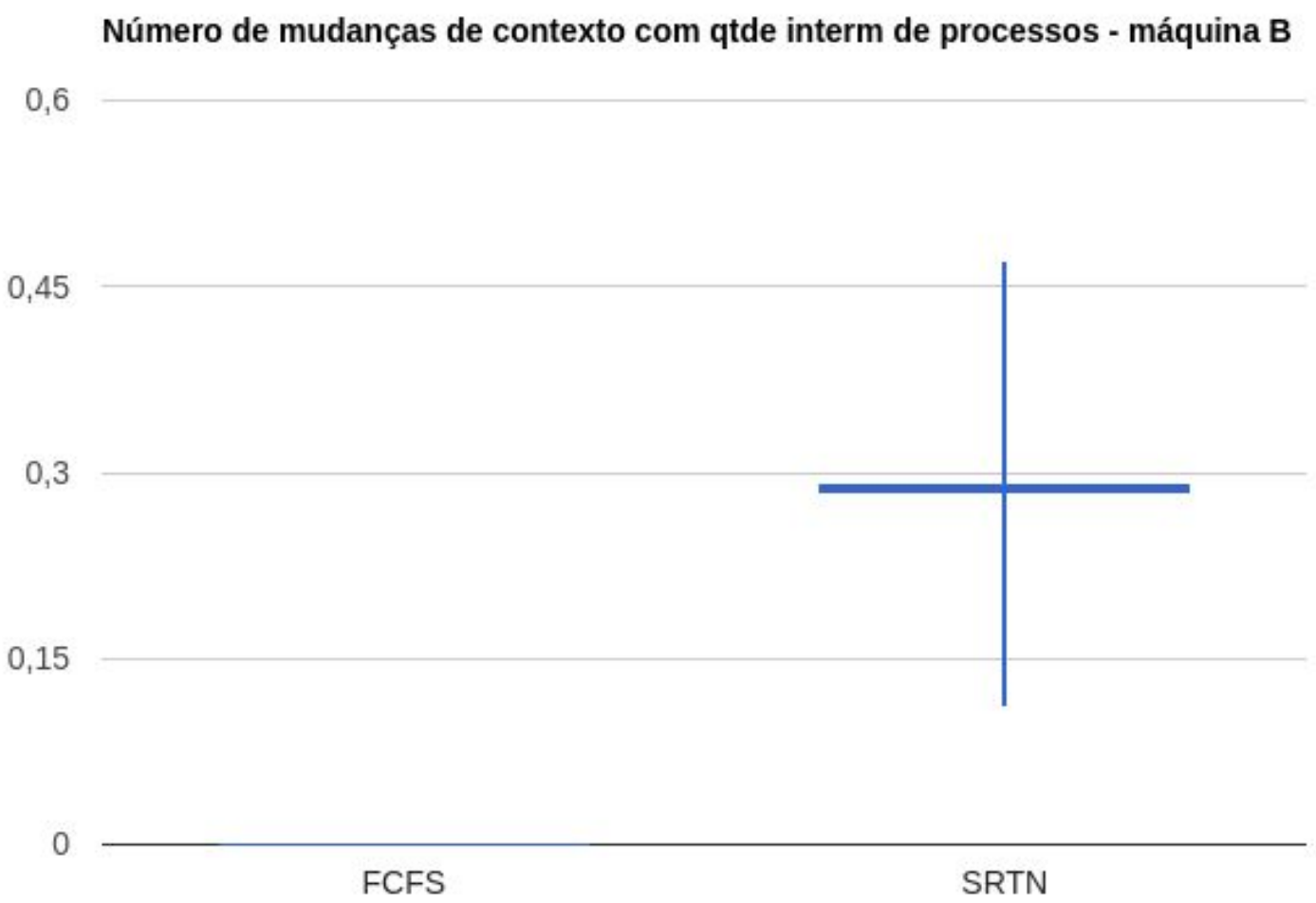

Número de mudanças de contexto com poucos processos - máquina A
1,2
0,9
0,6
0,3
0
FCFS
SRTN


Número de mudanças de contexto com qtde interm de processos - máquina A
4
3
2
1
0
FCFS
SRTN


Número de mudanças de contexto com muitos processos - máquina A
4
3
2
1
0
FCFS
SRTN


Número de mudanças de contexto com poucos processos - máquina B
1
0,75
0,5
0,25
0
FCFS
SRTN


Número de mudanças de contexto com qtde interm de processos - máquina B
1,6
1,2
0,8
0,4
0
FCFS
SRTN


Número de mudanças de contexto com qtde interm de processos - máquina B
0,6
0,45
0,3
0,15
0
FCFS
SRTN

Escalonadores

- Para a implementação dos escalonadores, utilizou-se threads representando os processos, e uma função representando cada escalonador, responsável por invocar as threads
- Dentro das threads se usou semáforos para resolver o problema das seções críticas

First Come First Served

- O primeiro tipo de escalonamento é o mais simples entre eles e é executado na ordem que os processos chegam ao sistema. Basicamente, se esperou que um processador ficasse ocioso para invocação da próxima thread da lista; então, sempre que um processo termina, outro já entra em sequência para ser executado

Shortest Remaining Time First

- Para este escalonador, antes de invocar a thread de cada processo, se pré calculou o tempo disponível que as threads possuíam para rodar antes do acontecimento de um novo evento
- Os eventos possíveis são **término de um processo** ou **chegada de um novo processo**

Shortest Remaining Time First

- Diante da ocorrência de um evento, as threads eram interrompidas de forma que se pudesse recalcular quais processos seguiriam em execução e quais iriam para a lista de espera (baseou-se no tempo de execução restante para cada processo em execução ou na lista de espera)

Shortest Remaining Time First

- Após o cálculo de quais processos seriam colocados em execução, se pré calcula novamente o tempo de execução disponível e em seguida as threads são invocadas novamente