MAC0422 - Sistemas Operacionais

EP1- USPShell + Simulador de Escalonador de processos

Nome: Matheus Silveira Feitosa

NºUSP: 11836692

Considerações Gerais da Implementação

O problema foi interpretado como uma extensão do clássico problema dos produtores e consumidores, mas dessa vez com um intermediário, este intermediário sendo justamente o escalonador de processos

Com isso, todos os escalonadores faziam uso de uma thread auxiliar, responsável por criar processos no instante em que eles chegam, e avisar o escalonador sobre isso. Após ter alocado todos os processos daquele instante, suspendemos esta thread para ela não competir com outros processos e efetuamos o escalonamento dos processos nas CPUs para execução propriamente dita.

Escalonamento por Prioridade

Para além dos escalonadores clássicos FCFS e SRTN, ainda foi implementado um terceiro escalonador a ser simulado baseado em prioridade.

O escalonamento por prioridade foi projetado segundo os princípios de:

- **A prioridade é definida, considerando o tempo sobrando entre a deadline e o instante em que o processo foi inserido na fila de prontos somado a seu tempo de execução. Se não houver tempo sobrando, então a prioridade é mínima.**
- **A prioridade máxima é -20 e a mínima é 19.**
- **A fila de prontos é ordenada de modo a ter os processos de maior prioridade na frente. Em caso de prioridades iguais, priorizamos o de menor tempo de execução.**
- **O número de quanta de cada processo é calculada de maneira linear onde o máximo de quanta recebível é 3 e o mínimo é 1.**
- **Por fim, a cada 3 rodadas recebidas sem conseguir finalizar, a prioridade do processo é reduzida em uma unidade a fim de evitar Starvation.**

*Cada quantum equivale a 1 segundo de execução

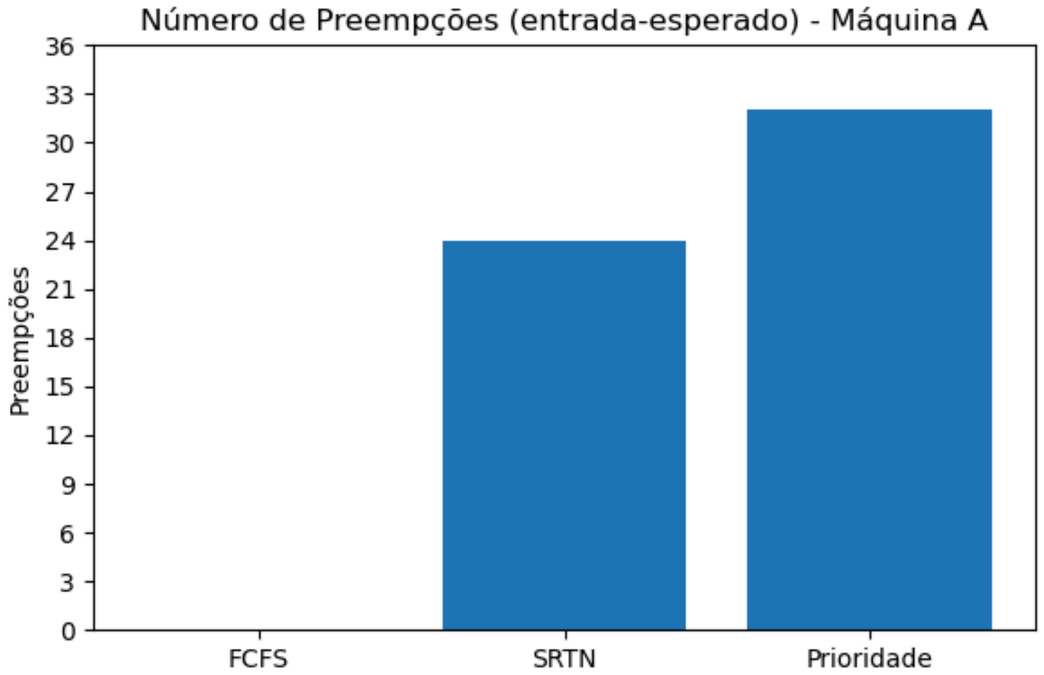
Máquinas de teste

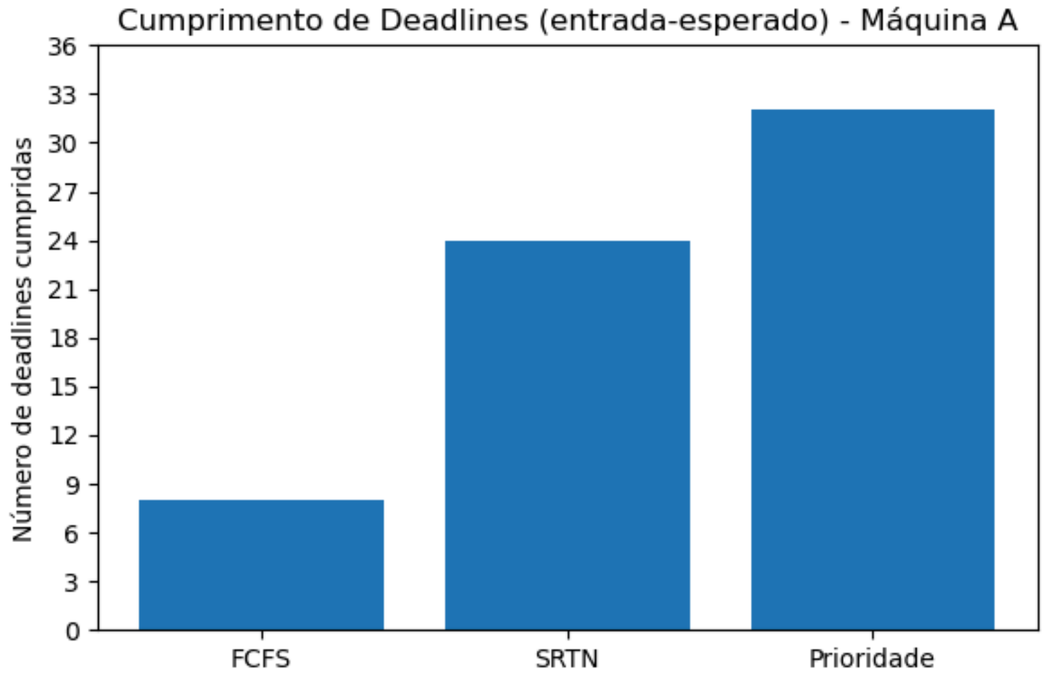
Com o objetivo de se estudar a influência da máquina nos resultados do escalonador, foram feitos testes com duas máquinas diferentes quantidades de CPUs utilizáveis. Tais especificações se encontram nas tabelas abaixo:

Máquina A	
Processador	Processador 11th Gen Intel(R) Core(TM) i5-11300H @ 3.10GHz, 3110 Mhz.
Número de Processadores	4 Núcleos, 8 Processadores Lógicos
Sistema Operacional	Debian GNU/Linux 12 (bookworm) 5.15.167.4-microsoft-standard-WSL2 #1
Memória RAM	Kingston Fury Impact, 8GB, 3200MHz

Máquina B	
Processador	Processador Intel(R) Atom(TM) x5-Z8350 @ 1.44GHz, 1450 Mhz.
Número de Processadores	1 Núcleo, 4 Processadores Lógicos
Sistema Operacional	Linux Mint 21.3 5.15.0-131-generic #141-Ubuntu
Memória RAM	Hynix Semiconductor, 2GB, 1066MHz

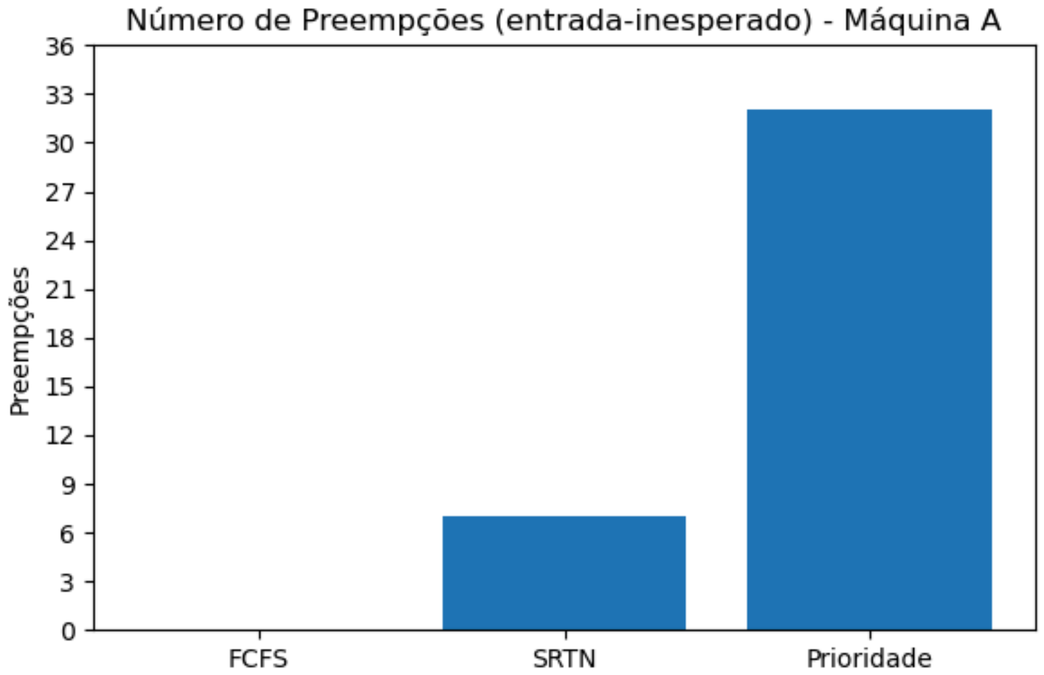
Resultados Esperados
Máquina A

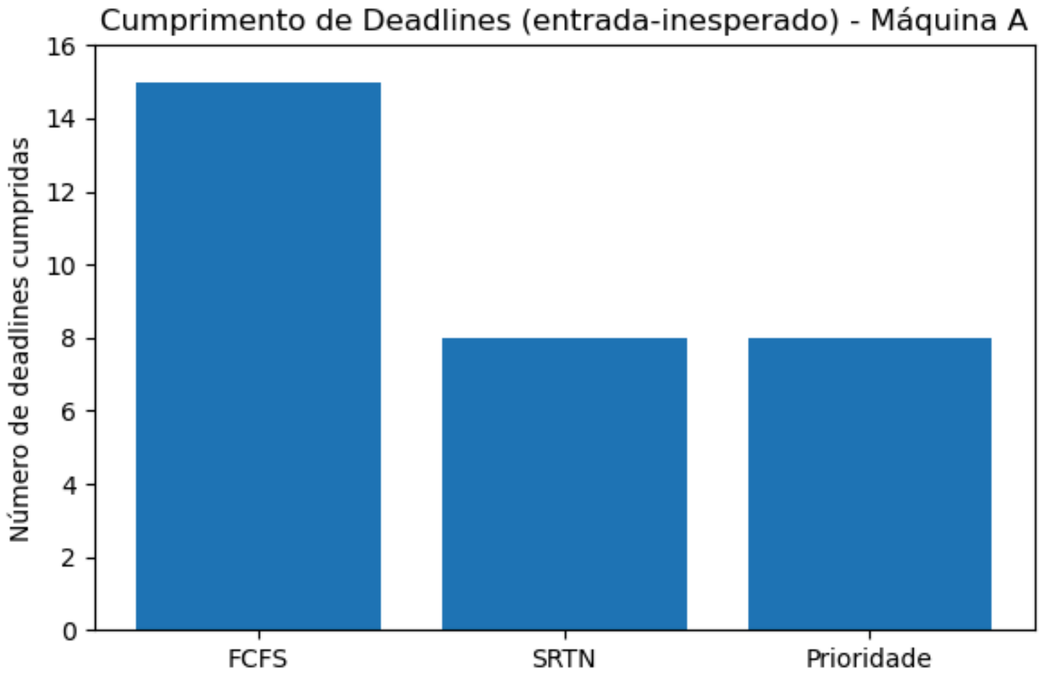




O arquivo de trace utilizado tinha 33 processos a serem simulados

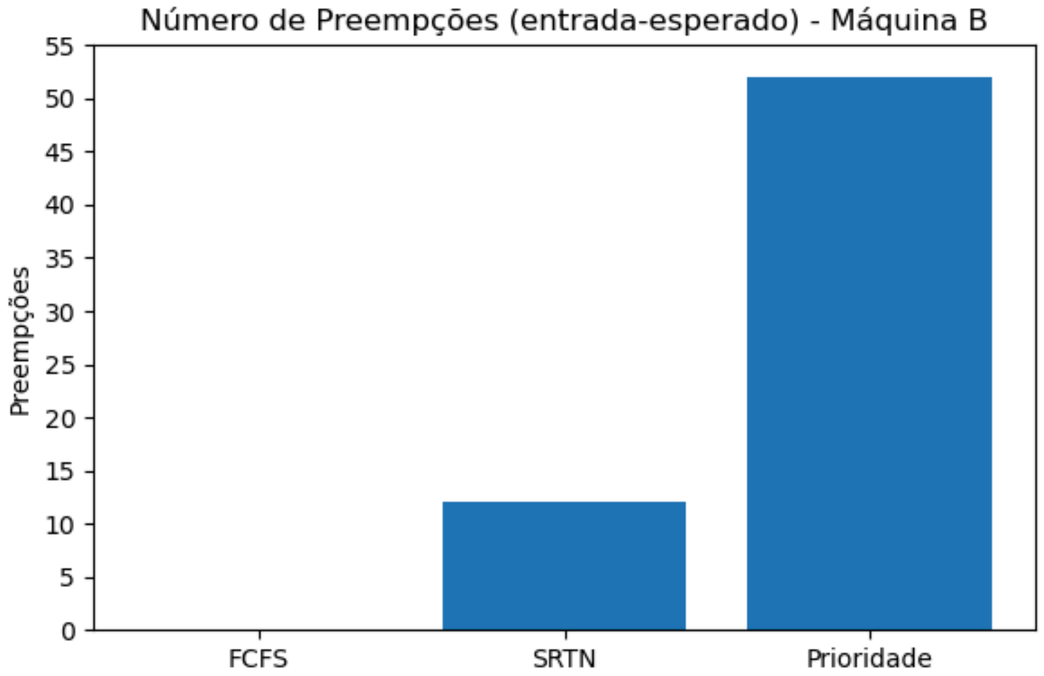
Resultados Inesperados
Máquina A

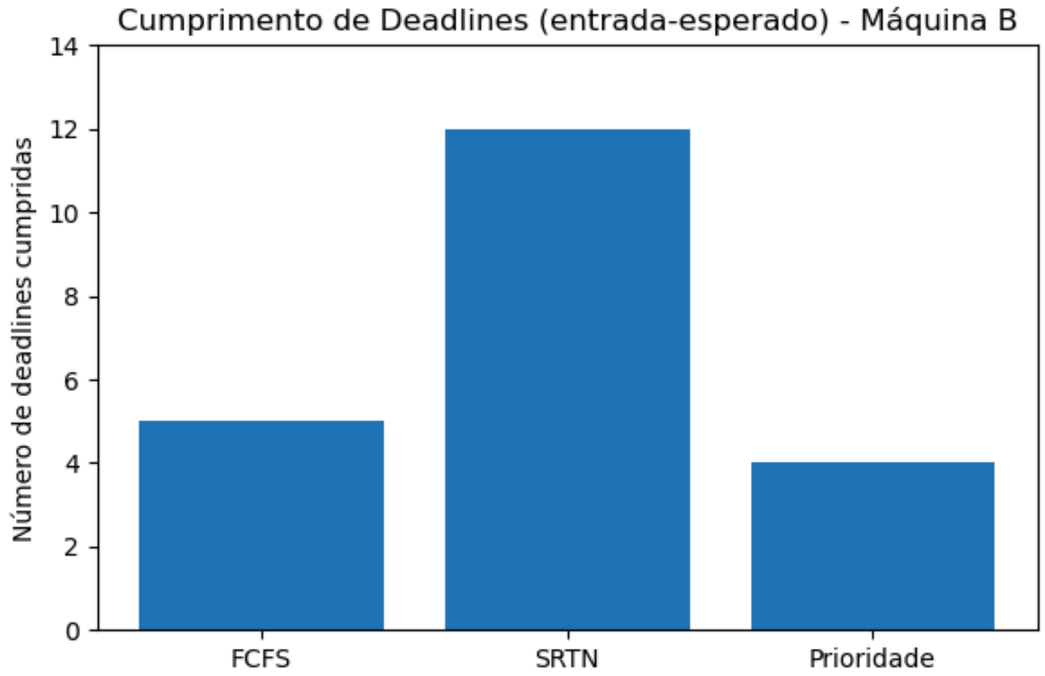




O arquivo de trace utilizado tinha 16 processos a serem simulados

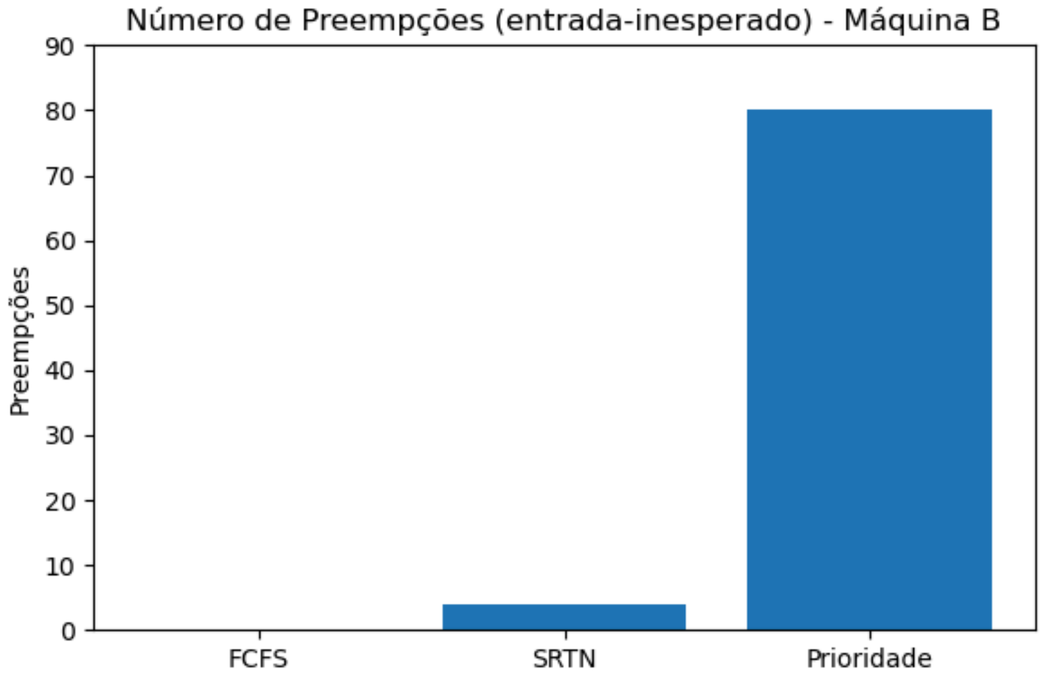
Resultados Esperados
Máquina B

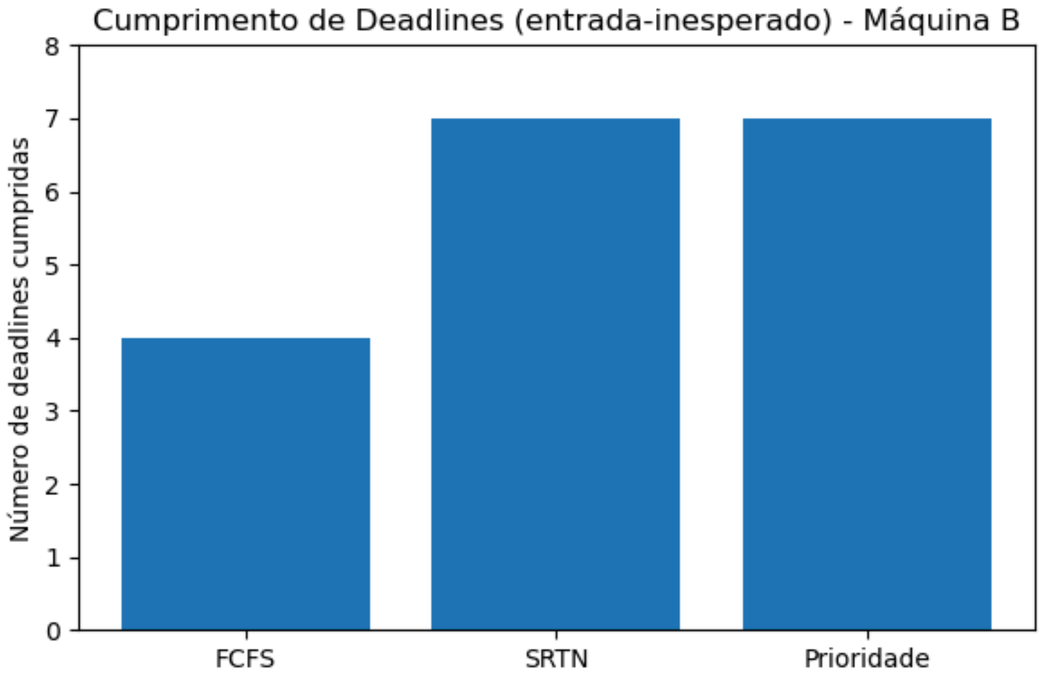




O arquivo de trace utilizado tinha 33 processos a serem simulados

Resultados Inesperados
Máquina B





O arquivo de trace utilizado tinha 16 processos a serem simulados

Análise dos Resultados

Máquina A

De forma geral, os testes foram construídos focando nesta máquina em específico.
Com isso, definindo folga como a diferença entre a deadline e a soma entre o tempo de chegada e o tempo de execução, podemos destacar o seguintes pontos:

- O FCFS executa os processos na ordem de chegada, consequentemente o seu número de deadlines cumpridas, depende exclusivamente da criticidade com a qual os processos chegam, isto é, se primeiro chegarem os processos com menor folga e em seguida processos com folga maior, então o FCFS irá performar de forma igual ou superior aos demais assim como visto em entrada-inesperado.
- O SRTN sempre irá parar processos longos em prol dos mais curtos, com isso se os processos mais curtos tiverem folgas maiores, é esperado que o SRTN consiga atendê-los, mais irá sacrificar os processos longos de forma desnecessária, assim como também visto em entrada-inesperado.
- Por fim, o escalonamento por prioridade, por considerar não só a o tempo de execução mas também a deadline, corrigimos parcialmente o problema dos processos curtos de folgas maiores, de forma tal a cumprir todas as deadlines esperadas, entretanto, ele tem a fragilidade de que, se um processo longo e com pouca folga perder sua prioridade, então processos mais curtos de folgas maiores poderão ser executados primeiro, sacrificando novamente os processos longos de forma desnecessária em prol da interatividade prevista na teoria do escalonamento por prioridade, assim como também observado nos resultados inesperados.

Análise dos Resultados

Máquina B

Neste outro caso, a máquina possuía metade dos núcleos em relação a máquina para a qual os testes foram originalmente pensados, e com isso, destacamos os seguintes pontos:

- Para o FCFS, em ambos os casos tanto o esperado quanto o inesperado, temos que o número de cores não é suficiente para atender outras deadlines além das 4 primeiras (e uma quinta para o caso esperado), justificando o motivo do número similar de deadlines atendidas em ambas as entradas.
- Já para o SRTN em ambos os casos, o mesmo fenômeno para o caso da máquina A foi observado, entretanto, com o número de reduzido de cores, vemos que, no caso esperado, metade das deadlines curtas e críticas foram atendidas, uma vez que nem todos os processos podiam ser executados ao mesmo tempo, em seguida para as deadlines curtas com folga maior, todas foram atendidas. Para o caso inesperado, vemos que apenas os processos curtos foram concluídos uma vez que no instante em que eles chegam, todos os processos longos são interrompidos impedindo que estes tenham chance de cumprir sua deadline, o que foi uma vantagem já que caso estes não fossem interrompidos, então apenas 4 processos longos seriam atendidos.
- Por último, no escalonamento por prioridade, observamos que, no caso inesperado, se não houvessem processos curtos, então nenhum processo atenderia a deadline, isto porque, a característica interativa deste escalonador, exige que cada grupo de 4 processos alterne entre suas execuções, e com isso, como eles precisam rodar o seu tempo de execução completo, estes não seriam atendidos. Mas como há processos curtos, a fragilidade vista em 8 cores foi uma vantagem para o caso de 4 cores. Em contraponto a isso, vale destacar que o caso esperado diferiu consideravelmente de sua proposta original, dado que, o mesmo fator que foi uma vantagem no caso inesperado, agora foi uma desvantagem para o caso esperado pois, caso o processo critico chegue no meio do quantum de outro processo, então quando conseguir iniciar sua execução, já não seria mais possível atender sua deadline, e com isso, apenas os processos que chegassem no exato instante do fim de um quantum puderam ser executados a tempo.

Máquina A vs Máquina B

Máquina A (8 Cores)

- Maior número de preempções
- Maior número de deadlines atingidas em média
- FCFS atinge mais deadlines
- É mais facilmente garantida o atendimento as deadlines com interatividade

Máquina B (4 Cores)

- Menor número de preempções
- Menor número de deadlines atingidas em média.
- FCFS atinge menos deadlines
- Atendimento as deadlines mais fortemente influenciado pelo tempo de duração, do que por outros fatores.
- Mais difícil de manter interatividade com atendimento a deadlines de processos.

No fim, assim como esperado, quanto menos cores nossa máquina dispõe, mais impactante é a escolha do escalonador utilizado, entretanto, independente do número de cores, a decisão a respeito de qual escalonador utilizar é fortemente influenciada pelos objetivos da máquina.
Em adição a isso, também como esperado, quanto mais cores dispomos, mais facilmente podemos atender as deadlines de um mesmo número de processos, mas mais importante que isso, eventualmente mais cores não implicam mais deadlines cumpridas dependendo do escalonador utilizado, assim como pode ser observado vendo que, para o caso inesperado, a diferença entre deadlines atingidas com o dobro de cores, foi de apenas 1 processo, tanto para o modelo com prioridade quanto para o SRTN.

Considerações sobre Determinismo

- **Para os arquivos de teste, a ordem específica na qual as preempções foram feitas não era impactante, fazendo com que o número de preempções e o número de deadlines atingidas fossem constantes.**
- **Entretanto, analisando estritamente a unicidade de cada processo, vemos que para processos com tempos idênticos, como não há um critério bem definido para o caso de todas as CPUs estarem ocupadas tanto para o escalonamento por prioridade quanto para o SRTN, a escolha entre qual processo irá rodar não é bem determinada, fazendo com que em cada execução, processos diferentes sejam parados e/ou continuados.**
- **Em especial para o SRTN como a deadline não é levada em consideração, caso hajam, em dado instante, diferentes processos rodando com o mesmo tempo restante, e chegue um processo mais curto, a escolha entre qual deles deve ser parado não é bem definida, e com isso, caso estes processos tenham diferentes deadlines, podem ocorrer também, diferenças no número de deadlines atingidas tornando assim, o escalonador SRTN não determinístico em termos de deadlines atingidas, o que não foi o caso para os arquivos de trace utilizados haja vista similaridades nas deadlines.**
- **Para o FCFS, a simulação sempre gera o mesmo resultado, independente do teste.**

Desta forma, podemos concluir que o FCFS é determinístico, entretanto, o algoritmo por prioridade não o é em termos da unicidade de cada processo, e o SRTN pode não ser determinístico, em situações onde há empate na escolha da preempção entre processos com tempos restante similares.