MAC0422

SISTEMAS OPERACIONAIS

EP1

Thainara de Assis Goulart
13874413

O simulador é determinístico?

Para as entradas testadas, sim!

- Controle explícito de tempo de chegada, cada processo só entra na fila após atingir seu t0.
- O acesso à fila é protegido com mutex, garantindo acesso exclusivo e sincronizado.
- Variáveis de controle protegidas com mutex (rest, ready, start, etc.), prevenindo condição de corrida.
- As threads só executam quando suas condições são satisfeitas.
- Cada thread é fixada a um único núcleo, evitando migração e reduzindo a variação de tempo real.
- A ordem de chegada e criação de threads está controlada.

Teste empírico de determinismo

- Foi executado um script que roda o simulador 7 vezes com a mesma entrada, para cada escalonador.
- Todas as saídas foram idênticas, confirmando o comportamento determinístico para os casos testados.

Escalonamento com Prioridade

Cálculo da “margem”

Para cada processo **p**, no instante de tempo **t** atual, calculamos a margem que o processo tem para ser concluído, ou seja, pegamos o tempo que falta para chegar no deadline (p.deadline - t) e subtraímos **p.rest**, o resto de tempo que falta para o processo concluir:

margem = p.deadline − t − p.rest

Algoritmo

- **margem < 0:**
  - Deadline não pode mais ser cumprido → **quantum = 1** (o menor possível, 1s)
- **margem == 0:**
  - Ainda há chance exata de cumprir deadline → **quantum = rest**
- **margem > 0:**
  - Ainda há folga → **quantum = (1 / margem) * 10**
  - Quanto maior a margem, menor o quantum necessário para aquele instante.
  - Multiplicamos por 10 pois queremos um tempo discreto.
  - Além disso, para evitarmos valores fora do esperado
    - se quantum < 1 → quantum = 1 (o menor possível, 1s)
    - se quantum > p.rest → quantum = p.rest (roda exatamente o que falta)

Resultados

Configuração das Máquinas

	Máquina A	Máquina B
Número de CPUs	• 4	• 8
Modelo do Processador	• Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz	• 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz
Arquitetura	• x86_64	• x86_64
RAM	• 8 GB	• 16 GB
Sistema Operacional	• Ubuntu 22.04.1	• Ubuntu 24.04.1

Resultados

Traces

Entrada Esperado

Temos **variados dt e deadline**, ou seja, temos processos longos, com dt altos e deadlines generosos e processos curtos com deadlines apertados e dt menores. Com essa variação conseguimos criar processos com “margens” pequenas ou grandes. Além disso, a **variação de chegada t0** dos processos força disputas de CPU.

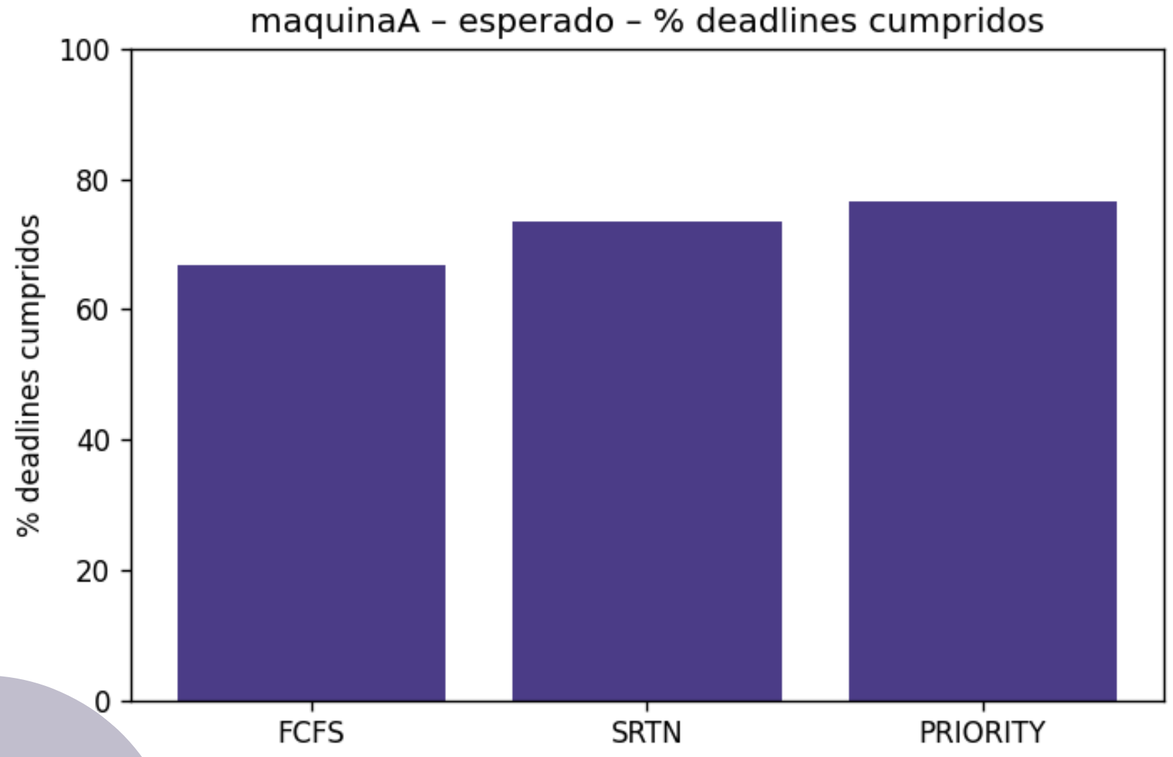
Entrada Inesperado

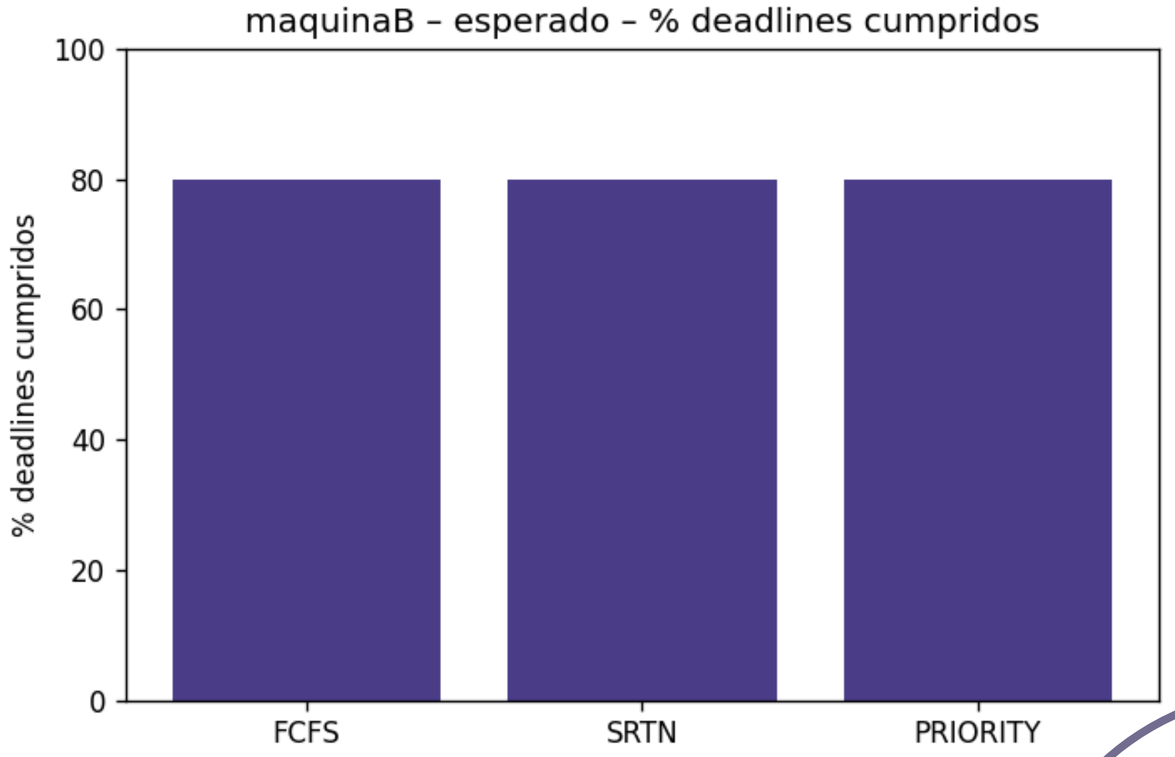
Temos **dt e deadlines idênticos** e **chegadas t0 concentradas em poucos instantes**. Assim, conseguimos eliminar diferenças de “margem” e também, com dt idênticos, evitamos preempções no SRTN.

Resultados - Entrada Esperado

Prioridade > SRTN > FCFS

% de processos que cumpriram deadline

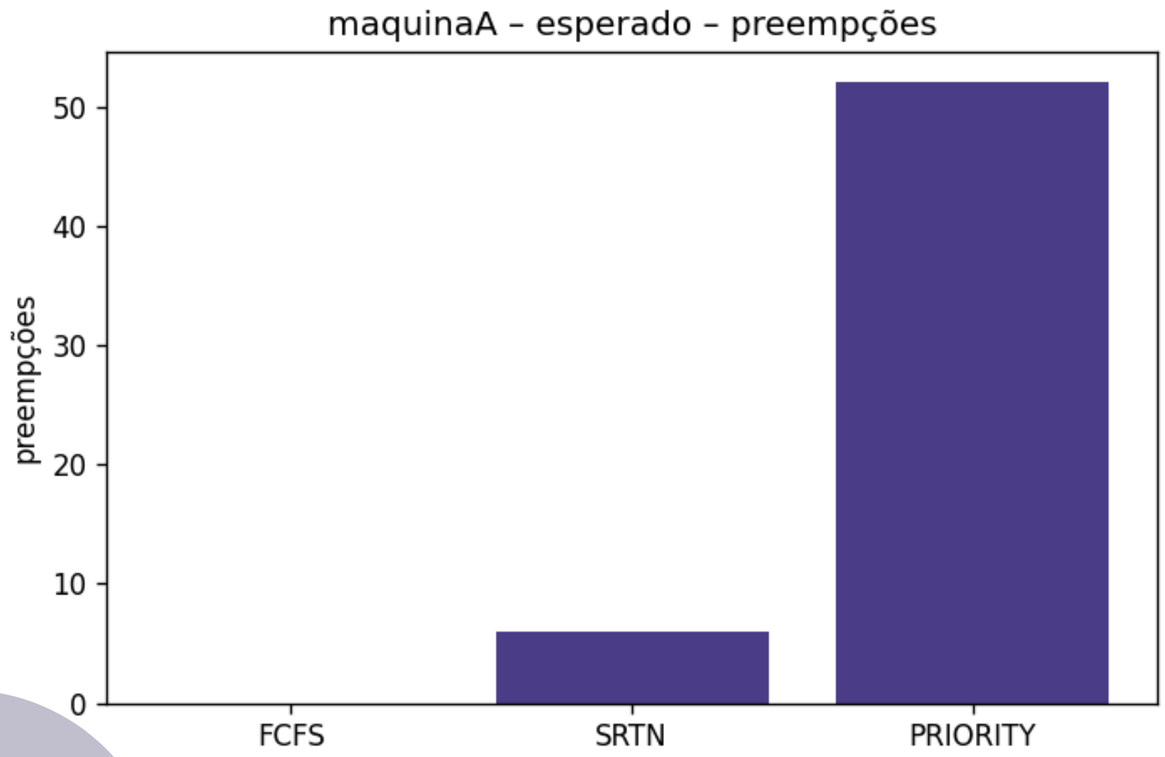


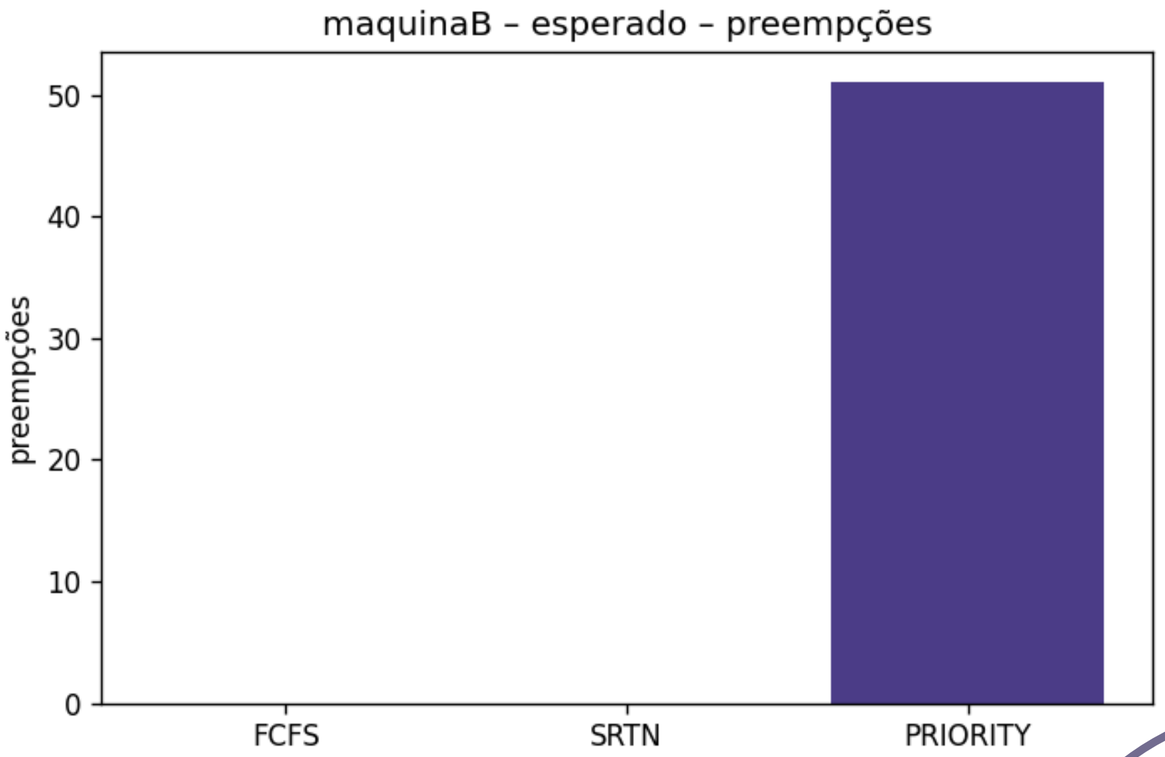


Resultados - Entrada Esperado

Prioridade > SRTN > FCFS

número de preempções





Resultados - Entrada Esperado

Prioridade > SRTN > FCFS

Análise - Máquina A

Impacto na % de deadlines cumpridos

- FCFS – 67% | SRTN – 73% | PRIORITY – 77%
- O escalonamento com prioridade aloca quantums maiores a quem tem menor “margem”, cumprindo mais deadlines. O SRTN fica no meio, pois, apesas de ser preemptivo, não considera o deadline na sua métrica. Já o FCFS fica por último, cumprindo menos deadlines, uma vez que apenas executa em ordem de chegada, sem ponderações, como esperado.

Impacto no número de preempções

- FCFS – 0 | SRTN – 6 | PRIORITY – 52
- O FCFS não preempta nunca. O SRTN só preempta quando chega alguém de dt menor, o que ocorre, já que temos dt variados. E o escalonamento com prioridade preempta muito mais, pois ajusta o quantum sempre o processo vai rodar.

Resultados - Entrada Esperado

Prioridade > SRTN > FCFS

Análise - Diferença de comportamento entre Máquina A e Máquina B

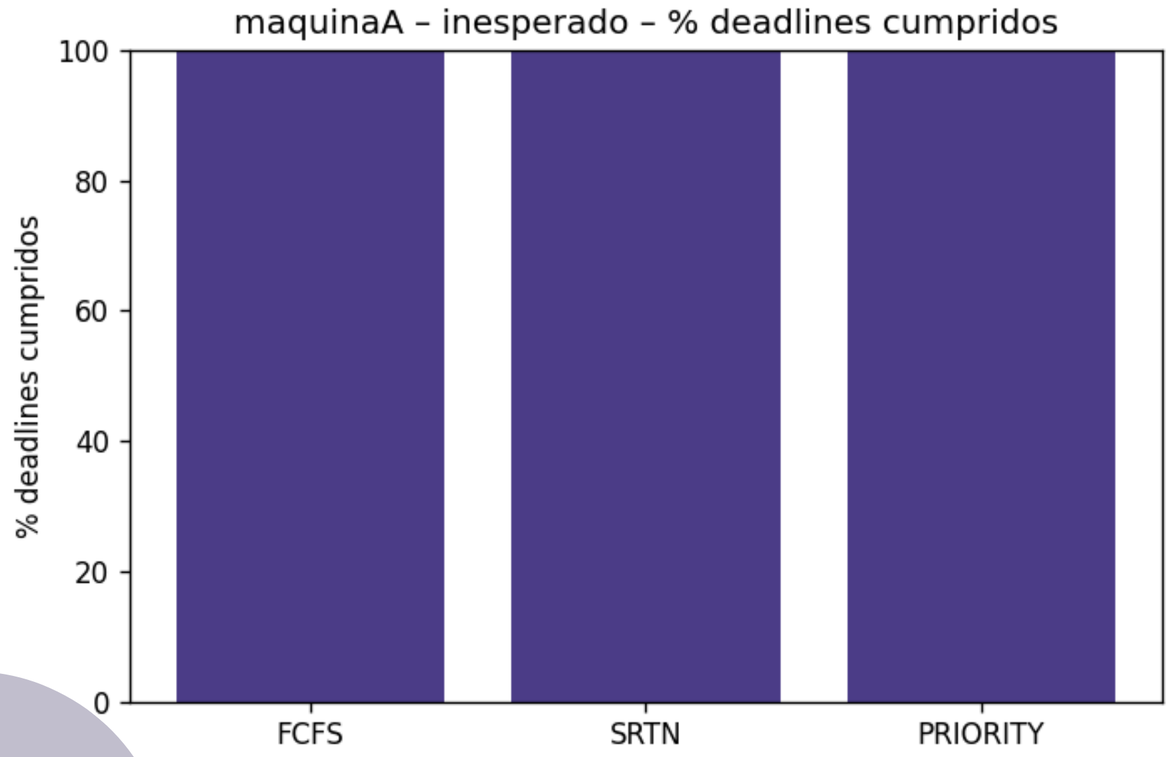
- **Ganho de paralelismo:** Na Máquina B, com 8 CPUs, temos que a % de deadlines cumpridos, para todos os escalonadores, sobe para 80%. Isso ocorre, pois, mais núcleos reduzem a espera por CPU, permitindo que processos longos iniciem simultaneamente e reduzam atrasos. Além disso, com núcleos a mais, o uso do escalonamento com prioridade não traz ganhos adicionais, uma vez que a paralelização já atende praticamente todos os prazos.

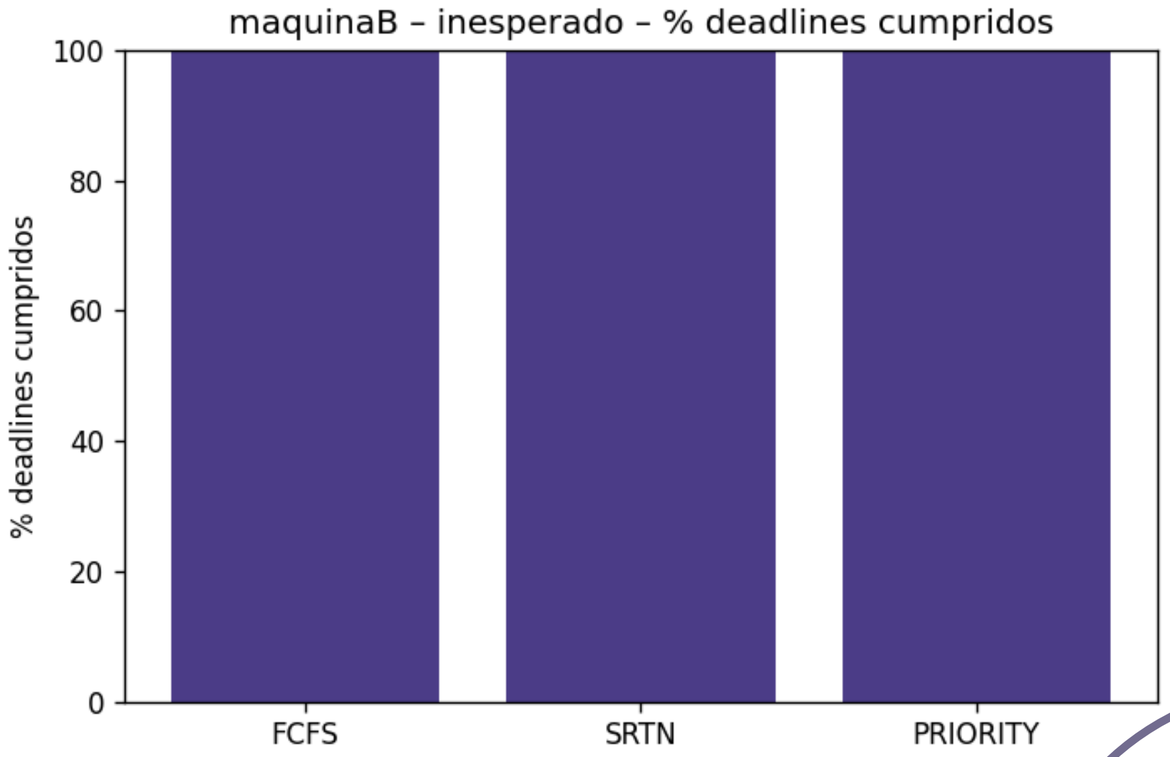
- **Preempções:** O número de preempções do escalonamento com prioridade é praticamente o mesmo entre as máquinas, porque o mecanismo de quantum adaptativo dispara as mesmas trocas, independentemente de quantos núcleos existam. Já o SRTN preempta apenas quando chega processo com dt menor, porém como na Máquina B temos mais CPUs, não houve essa necessidade de preempção, pois haviam núcleos vazios para executar os processos quando chegavam.

Resultados - Entrada Inesperado

FCFS >= SRTN >= Prioridade

% de processos que cumpriram deadline

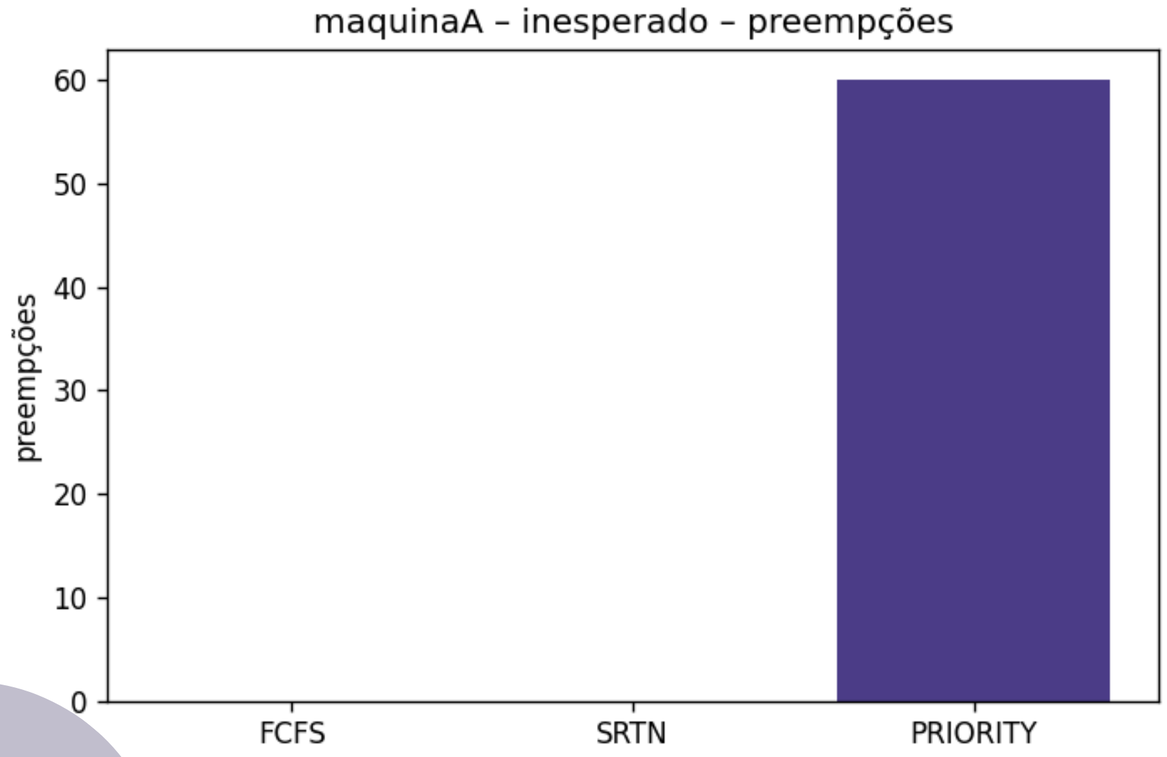


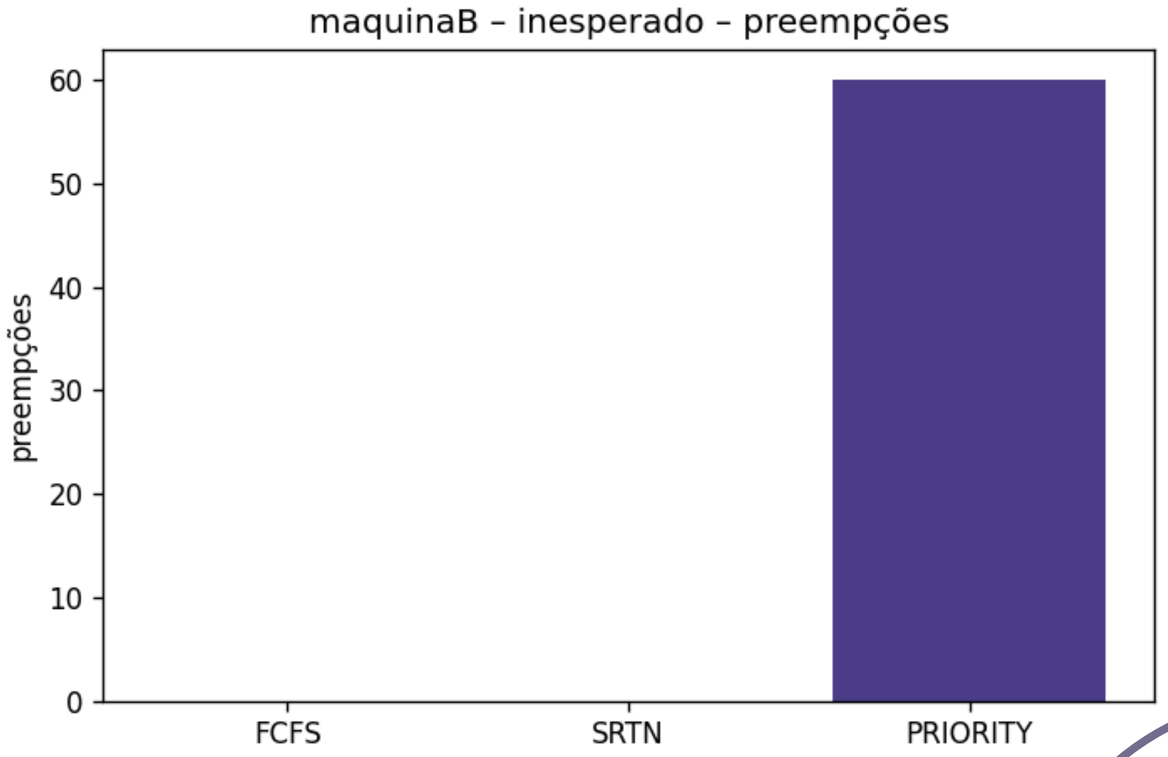


Resultados - Entrada Inesperado

FCFS >= SRTN >= Prioridade

número de preempções





Resultados - Entrada Inesperado

FCFS >= SRTN >= Prioridade

Análise - Ambas Máquinas

Impacto na % de deadlines cumpridos

- Todos cumprem os deadlines de todos processos, para ambas máquinas.
- Como não há diferença real de urgência entre processos, não ocorre necessidade priorizar alguns processos, de maneira que todos escalonadores se comportam de maneira semelhante. Assim, o FCFS tende a ter igual ou melhor % deadline, como deveria ocorrer no cenário “inesperado”.

Impacto no número de preempções

- FCFS – 0 | SRTN – 0 | PRIORITY – 60, para ambas máquinas.
- O FCFS não preempta nunca. Já o SRTN, como não há variação de dt, também não preempta. Por fim, no escalonamento com prioridade ocorre muitas preempções, uma vez que todos os processos possuem margens semelhantes e altas, dessa forma, pelo nosso algoritmo, um processo irá rodar muitas vezes por apenas 1s sem melhorar o cumprimento de prazo.