EP01 - SIMULADOR DE PROCESSOS

MAC0422 - Sistemas Operacionais

Professor: Daniel Batista

Aluno: Francisco Membrive

Sobre o determinismo

Ao realizar os experimentos, com 30 execuções em cada máquina, foi demonstrado que o simulador de processos desenvolvido não é determinístico, mas apresenta pequenas variações no número de preempções e no número de deadlines cumpridas. Isso condiz com o esperado já que, embora na maior parte do código não exista margem para comportamento inesperado, no lançamento as threads competem livremente, de modo que a ordem de execução de processos com o mesmo t0 pode variar. Além disso, caso um processo tente entrar na fila de prontos (foi preemptado) ao mesmo tempo que outro, as threads também competem livremente. Portanto, arquivos de trace que não apresentem esses dois casos geram comportamentos determinísticos no simulador.

Algoritmo por Prioridade

- Assim como solicitado, o algoritmo não ordena a fila de prontos, sendo do tipo FIFO.
- Cada vez que há um núcleo livre, o algoritmo percorre a fila distribuindo quantums conforme o critério definido.
- O processo então é preemptado ao fim daquela quantidade de quantums, a não ser que seja concluído.

Critérios de distribuição de Quanta

- Basicamente, há quatro casos:
  - Se a deadline do processo já estiver “vencida”, isto é, impossível de cumprir, o processo recebe 1 segundo.
  - Se o tempo disponível entre o tempo atual e a deadline for menor que duas vezes o dt do processo, o processo recebe 10 s.
  - Se a quantidade de processos na fila for maior que o tempo entre a deadline e a conclusão da execução do tempo recebido na rodada, recebe 10 segundos.
  - Se não entrar em nenhum desses casos, recebe 1 segundo.

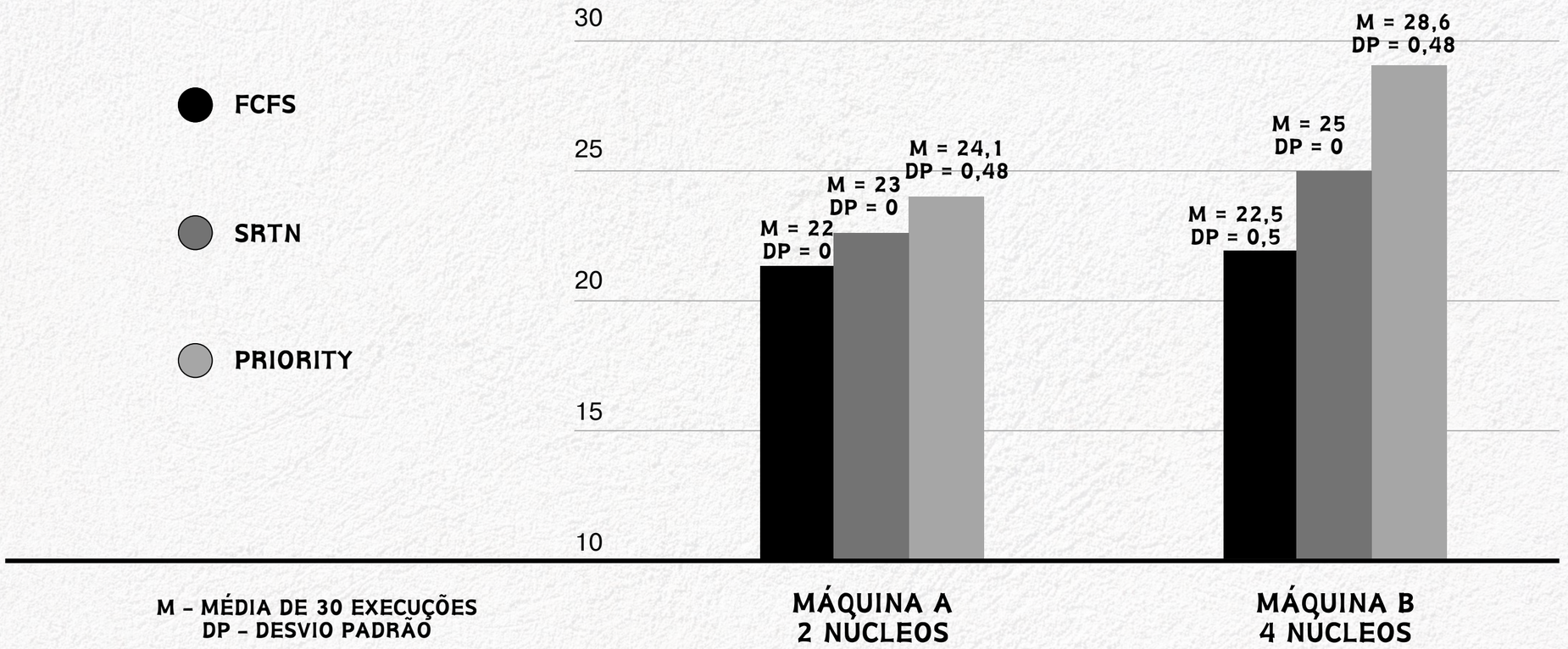
Arquivos de Entrada

- Para desenvolver os arquivos de entrada esperado e inesperado, foi necessário introduzir processos que ocupassem o escalonador usando o seu critério de prioridade como fraqueza.
- Na entrada esperada, garantiu-se o desempenho melhor do PRIORITY através de processos curtos com deadline longa que ocupavam o SRTN e processos longos com deadline longa que ocupavam o FCFS. Assim, só o PRIORITY resolvia os processos intermediários com deadline mais curta.
- Já na inesperada, lançamos múltiplos processos para receberem o quantum mínimo de PRIORITY, de modo que ele não conseguisse cumprir a deadline dos que chegassem depois. Além disso, processos com deadline apertada e duração maior que o quantum máximo, de modo a também prejudicar o SRTN.
- Como os arquivos de entrada não geram preemptações no SRTN, vou marcar as preemptações do PRIORITY nos gráficos de deadline.

Máquinas

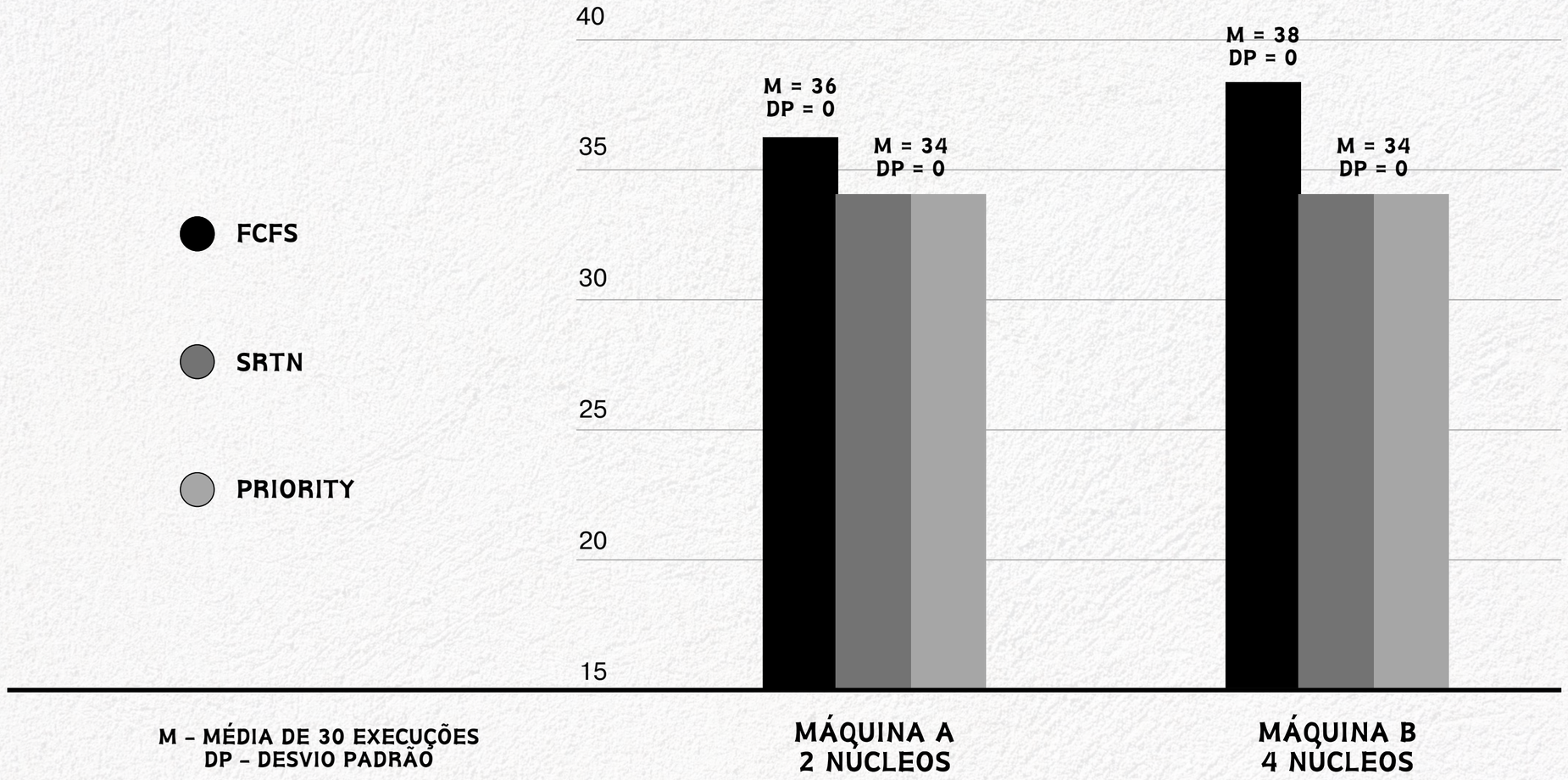
- A máquina A corresponde a uma máquina com processador de 2 núcleos Intel i3 4170 e 12GB de RAM DDR3 que executou no Windows 10 Pro usando WSL com Ubuntu 20.04,
- A máquina B corresponde a uma máquina com processador de 8 núcleos Ryzen 5 4800HS e 16GB de RAM DDR4 que teve seu uso limitado a 4 núcleos para ser comparável à outra máquina. O Sistema Operacional é Ubuntu 24.04.

NÚMERO DE DEADLINES CUMPRIDAS – TRACE ESPERADO
TOTAL: 29 PROCESSOS



NA MÁQUINA A, PRIORITY PREEMPTOU EM MÉDIA 104, 4 VEZES, COM DESVIO PADRÃO DE 0,8.
JÁ NA MÁQUINA B, PREEMPTOU UMA MÉDIA DE 94,4 VEZES COM DESVIO PADRÃO DE 0,8.

NÚMERO DE DEADLINES CUMPRIDAS – TRACE INESPERADO
TOTAL: 38 PROCESSOS



NA MÁQUINA A, PRIORITY PREEMPTOU 58 VEZES EM TODOS OS 30 TESTES.
JÁ NA MÁQUINA B, PREEMPTOU 40 VEZES EM TODOS OS 30 TESTES.

Conclusão

- É possível desenvolver arquivos trace que beneficiam qualquer um dos escalonadores.
- No geral, o escalonador PRIORITY deixa menos processos em espera longa e permite mais interatividade. Ainda assim, em um sistema real, deve ser levado em conta o custo de se fazer um escalonador preemptivo, de modo que a escolha ideal depende muito do objetivo do projeto.
- O SRTN desempenhou muito bem no universo desse trabalho com limite de 120 s de durações inteiras. Mas um cuidado ao utilizá-lo é observar que ele pode deixar processos muiro longos em espera eterna.

OBRIGADO