Apresentação ep1 SO

Octávio Gomes Carneiro
13831609

Informações sobre os testes

-> A máquina A possui 8 cores e a B possui 4 cores.

-> Isso, contudo, não deve ser importante já que evitei race conditions na implementação, ou seja: o processo de decisão de quem qual thread vai rodar é determinístico e mutuamente exclusivo.

-> Os testes foram rodados ~45 vezes em cada máquina. (umas 4 horas de processamento em cada).

->Não foram simuladas interrupções por E/S. A tarefa executada pelas threads é incrementar um contador local.

Sobre os traces

->Trace pequeno (5 processos):

->Pensado para otimizar o escalonador com prioridade.

->Processos que têm o t0 similar e uma deadline próxima, e outros processos que estarão rodando ao mesmo tempo com deadlines mais distantes.

-> A prioridade “foca” nos processos com t0 igual e deadline próxima enquanto o SJF e o RR “perdem tempo” em outros que tem uma deadline mais leniente.

*otimizar := maximizar deadlines cumpridas.

Sobre os traces

->Trace médio (10 processos):

-> Feito para otimizar o Round Robin.

-> Processos com t0s esparsos e deadlines distantes, porém próximas.

-> RR processa cada um um pouco por vez, enquanto os outros se concentram em certos processos. Nessa “concentração”, as deadlines próximas passam.

Sobre os traces

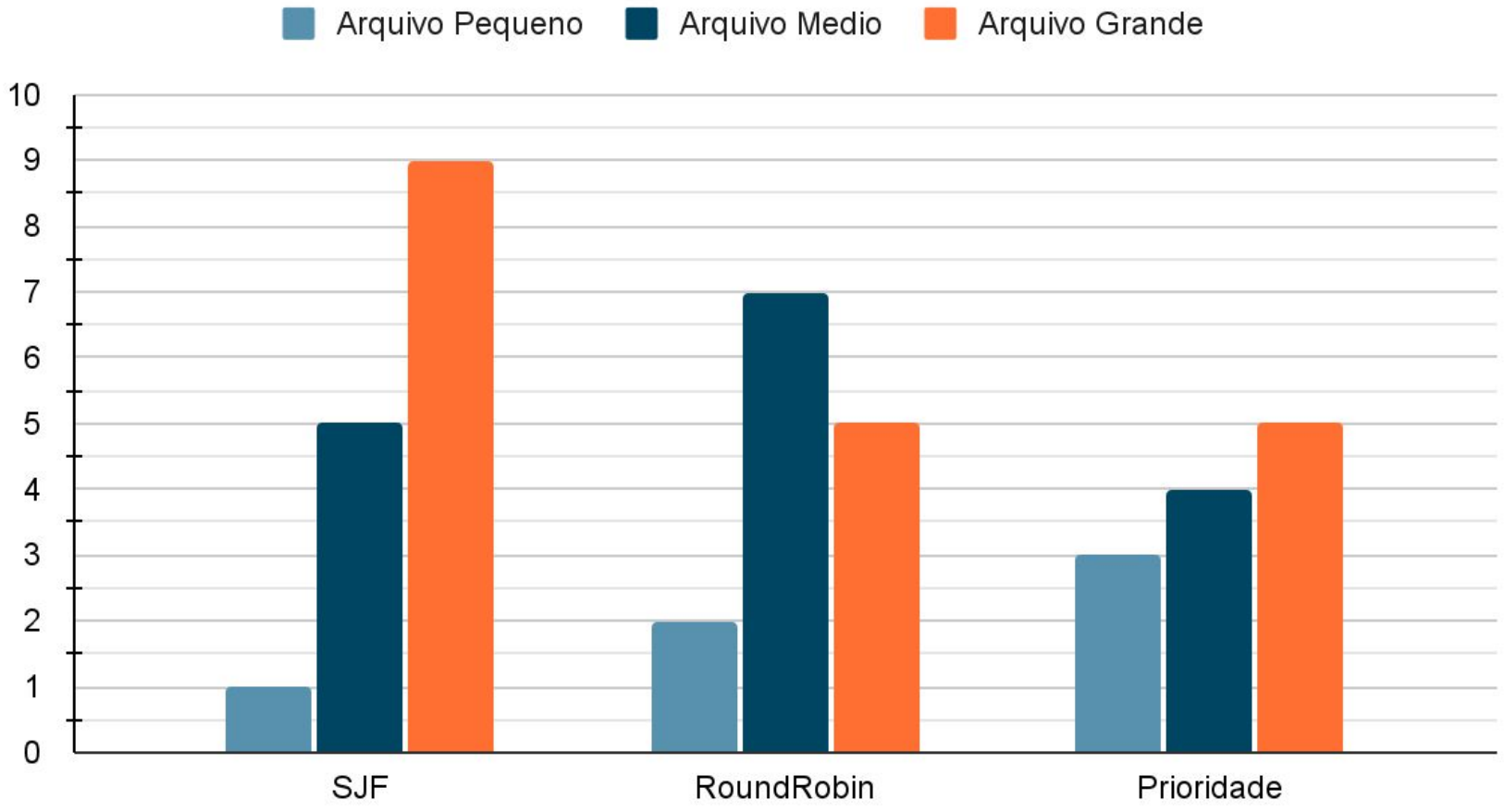
->Trace grande (20 processos):

->Feito para otimizar o SJF.

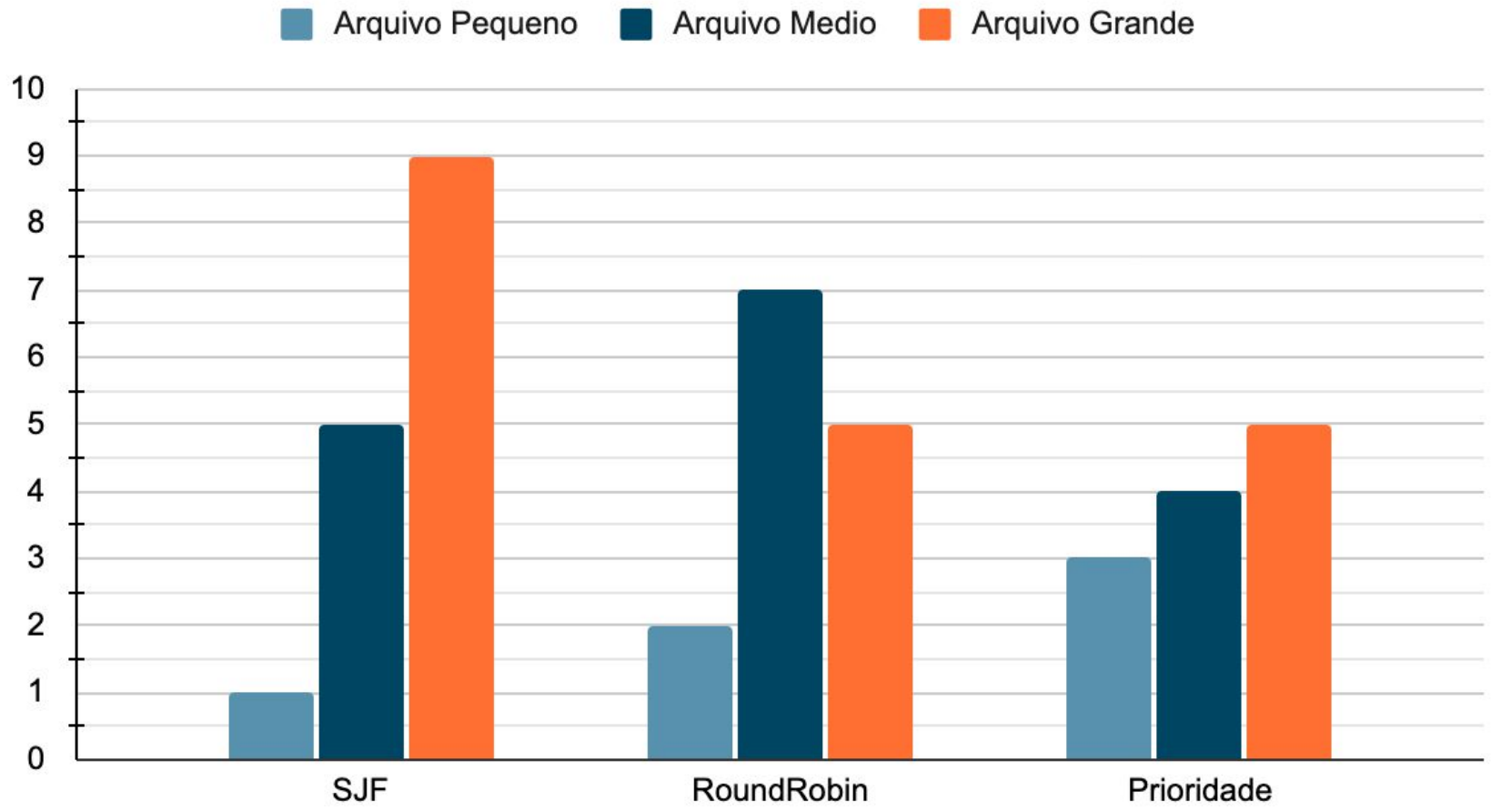
->Processos com t0 e deadlines estritas, que sugerem uma certa ordem de execução.

->Escalonadores com preempção distribuem a execução, e perdem muitas dessas deadlines.

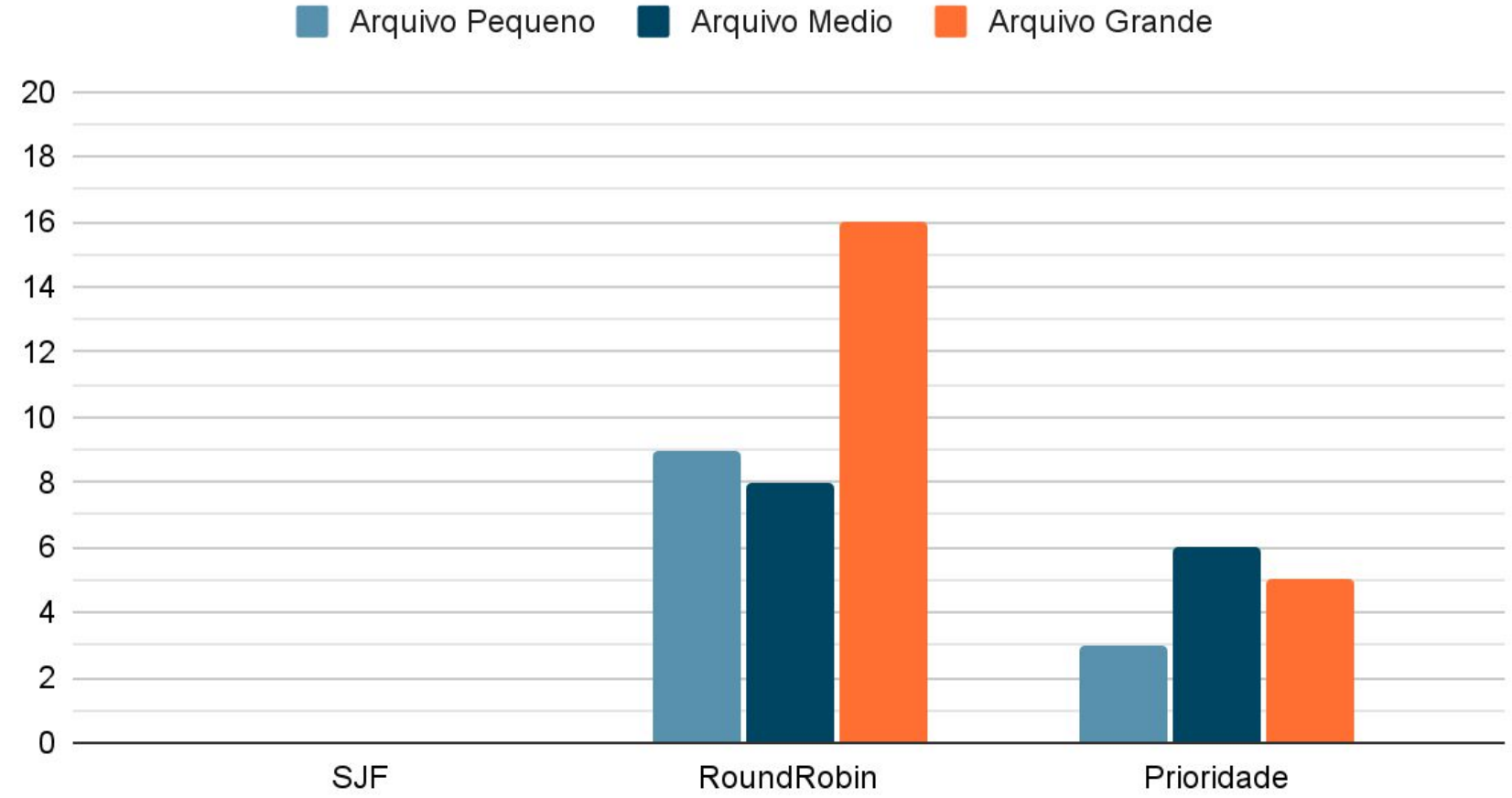
Cumprimento de Deadlines Maquina A



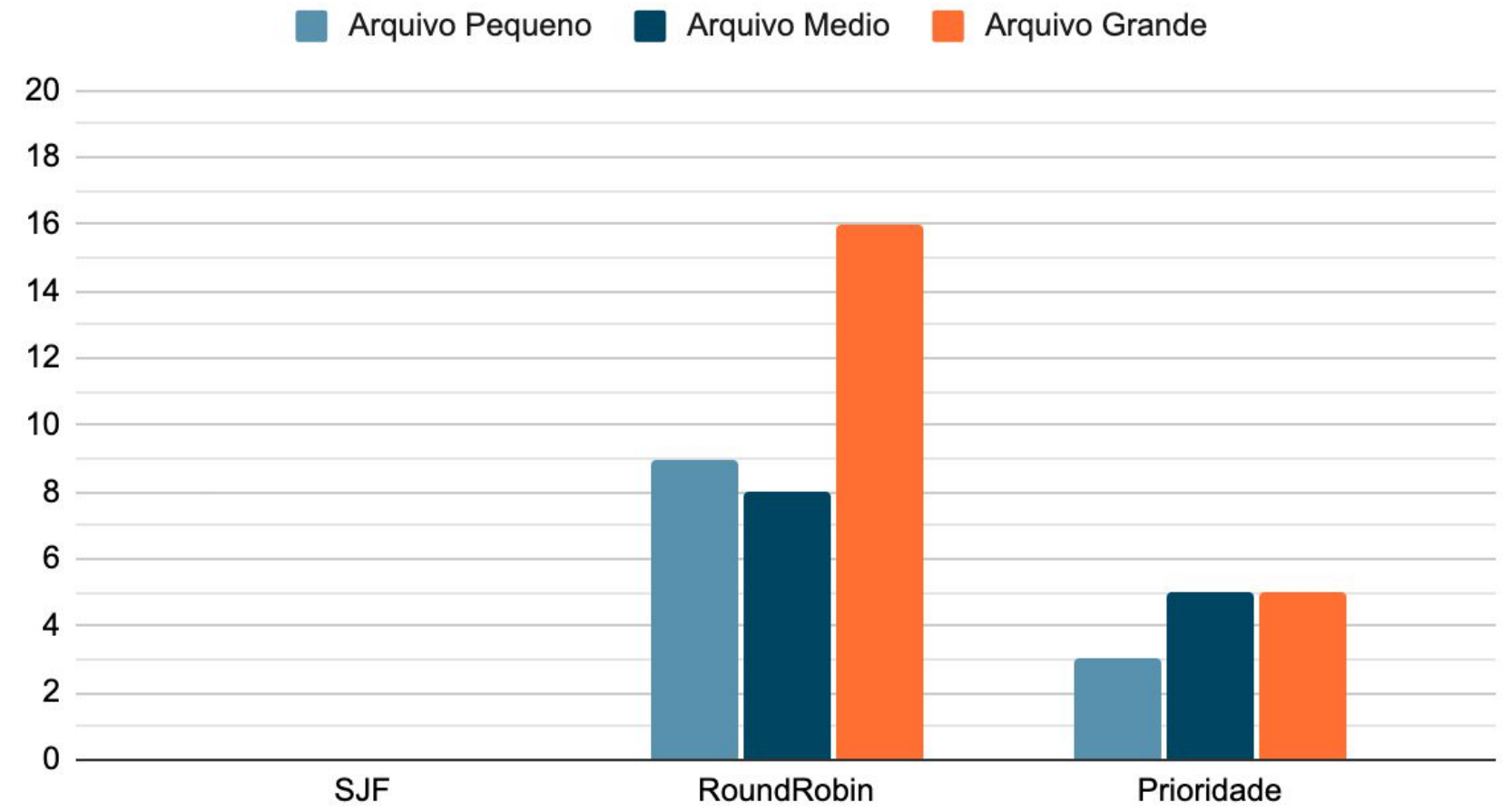
Cumprimento de Deadlines Maquina B



Mudanças de contexto na Máquina A



Mudanças de contexto na Máquina B



Resultados

-> Os traces e os escalonadores se comportaram conforme o esperado, de acordo com as explicações dadas antes dos gráficos.

->Sobre as mudanças de contexto, os resultados foram esperados também: SJF nunca muda, já que não tem preempção, round robin muda de contexto mais que o escalonador com prioridade, já que o RR tem, sempre, quantums menores ou iguais.

Diferença entre as máquinas

-> 1 diferença, apenas. Na máquina de 4 cores, o trace Médio teve 5 preempções com prioridade, enquanto outra máquina teve 6. Isso acontece durante a execução do p1 e do p2 logo nos primeiros passos do trace. Na máquina de 4 cores, p2 roda antes de p1, e como a dt de p2 é menor que um quantum, tem uma preempção a menos.

->Isso deve acontecer por diferenças dos sistemas operacionais (A é ubuntu 22.04 e B é macOs 11.7.10), que tem compiladores de C diferentes. Ou por causa dos processadores com CLOCKS_PER_SECOND diferentes, resultando em compreensões diferentes do instante atual dentro das threads e durante a montagem da fila de processos.