Testes de Escalonabilidade

Sérgio Campos

Testes de Escalonabilidade

- Teste de aceitação
- Períodos harmônicos
- Múltiplas frames
- Preemptabilidade
- Limite de prioridades

Teste de Aceitação

O teste de escalonabilidade exato anterior é caro:

- E se não houver tempo para calculá-lo ?

Para Rate-Monotonic, um sistema com

- n tarefas independentes;
- $D_i = p_i$;
- preempção perfeita;

pode ser escalonado se (mas não somente se):

$$\sum_{U_i} \leq U_{rm}(n) = n(2^{1/n} - 1)$$

$$n \to \infty \Rightarrow U_{RM}(n) \to ln(2) \simeq 0.693$$

Teste de Aceitação

Exemplo:

- $T = \{(1, .25), (1.25, .1), (1.5, .3), (1.75, .07), (2, .1)\}$
- $\sum U_i = .62 \leq U_{RM}(5) = .743$

Mas não é tão preciso. Por exemplo:

- $T = (3, 1), (5, 1.5), (7, 1.25), (9, .5)$;
- $\sum U_i = .85 > U_{RM}(4) = .757$
- No entanto LSD confirmou escalonabilidade.

Períodos Harmônicos

Mas .693 é muito baixo... Podemos fazer melhor:

Particione as tarefas em grupos com períodos harmônicos. e.g.:

- $T = 4, 7, 8, 14, 16, 28, 32, 56, 64$
  - $U_{RM}(9) = .712$
- $T_1 = \{4, 8, 16, 32, 64\}$; $T_2 = \{7, 14, 28, 56\}$
- O sistema é escalonável se $\sum_{U_i} \leq U_{RM}(n)$
  - $U_{RM}(2) = .828$

Pergunta: e se todas as tarefas tiverem períodos harmônicos ?

Tarefas com Múltiplas Frames

MPEG: I-frames gastam mais tempo de P e B-frames.

A cada 33ms existem 1 I-frame e 5 P e B-frames:

- I-frame gasta 1ms; P e B-frames gastam .3ms
- Considerar e = 1ms é muito pessimista!

Tarefa com Múltiplas Frames

Consideramos uma tarefa multiframe: $(p, \varepsilon, e^p, e^n)$:

- p é o período;
- e^p é o tempo de execução de pico;
- e^n é o tempo de execução normal;
- ε: cada pico é seguido por $\varepsilon - 1$ exec. normais.
- $U = \frac{(e^p + (\varepsilon - 1)e^n)}{\varepsilon p}$

Teste de aceitação

- $\Delta = min_{1 \leq i \leq n}(e_i^p / e_i^n) \rightarrow$ variação de carga.
- $U_{RM}(n, \Delta) = \Delta n((\frac{\Delta + 1}{\Delta})^{1/n} - 1)$

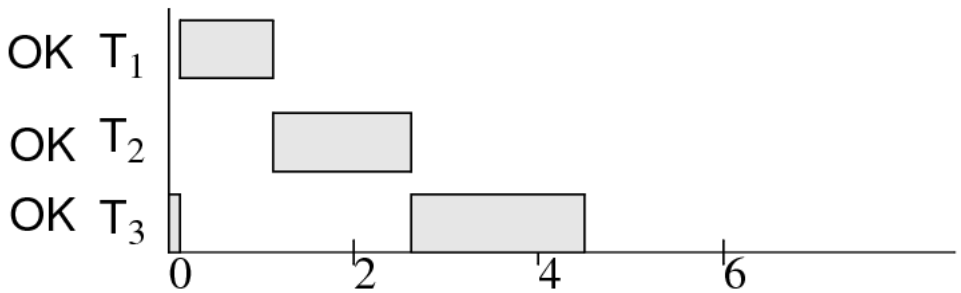
Preemptabilidade

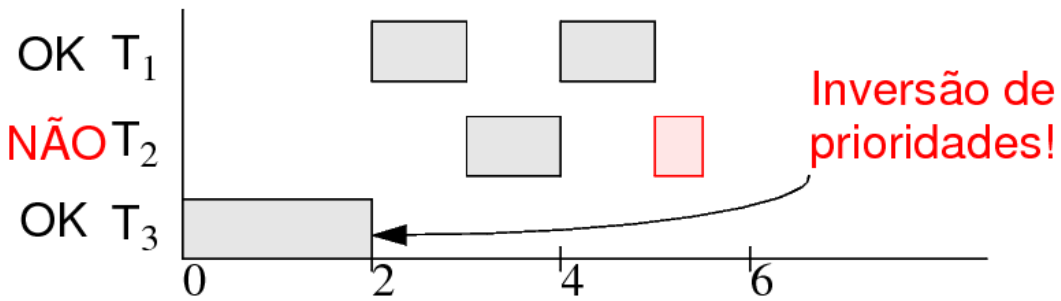
Na realidade preemptabilidade instantânea não existe

- Exclusão mútua entre processos;
- Operações como leitura em disco;
- Código do SO.

Exemplo:

- $T = (\varepsilon, 4, 1), (\varepsilon, 5, 1.5), (0, 9, 2)$.





Preemptabilidade

Tem que levar em conta tempo de não preempção:

- Seja θ_i a maior porção não interrompível de T_i.
- Seja $b_i(np) = max\theta_j, i < j \leq n$ (menor prioridade).

$b_i(np)$ é o maior tempo de não preempção de processos de baixa prioridade durante a execução de T_i.

Adiciona-se $b_i(np)$ ao tempo de execução da tarefa i e somente dela. A tarefa T_i é escalonável se:

$$\frac{e_1}{p_1} + \frac{e_2}{p_2} + \ldots + \frac{e_i + b_i}{p_i} = \frac{U_i + b_i(np)}{p_1} \leq U_{RM}(i)$$

Auto suspensão

- Tratada da mesma forma, mas por processos de prioridade mais alta.
- Assume-se self suspension time sst conhecido.
- $b_i(ss) = sst_i + \sum_{k=1..i-1} min(e_k, sst_k)$

Mas se T_i dormir k_i vezes, $b_i(np)$ afeta k_i vezes.

$$b_i = b_i(ss) + (k_i + 1)b_i(np)$$

Número Limitado de Prioridades

No mundo real prioridades são limitadas:

- IEEE token ring possui 8 níveis de prioridade;
- SO possui 256.

Um job pode bloquear outro com a mesma prioridade.

- No pior caso todos iguais têm maior prioridade!

Número Limitado de Prioridades

Mapeamento de prioridades:

- Uniforme: 1: $\{1, 2, 3\}$, 2: $\{4, 5, 6\}$, 3: $\{7, 8, 9, 10\}$
  - Utilização de escalonabilidade muito baixa.

- Proporção constante: $(pri_{i-1} + 1)/pri_i$ constante
  - Mais níveis para prioridades maiores.
  - 1: $\{1\}$; 2: $\{2, 3, 4\}$; 3: $\{5, 6, 7, 8, 9, 10\}$
  - Se g é a variação mínima entre níveis,

$$U_{Lim} = (ln(2g) + 1 - g) \text{ se } g > .5; g \text{ se } g \leq .5$$

  - Escalonabilidade relativa U_{Lim}/U_{RM} mede perda de escalonabilidade
    - 100000 níveis reduzidos para 256 $\rightarrow$ perda de 0.2%.