Três Enfoques

Sérgio Campos

Três Enfoques

- Clock driven
- Weighted round-robin
- Prioridades

Clock Driven Design

O instante em que cada job executa é decidido a priori

- Todos os jobs são conhecidos;
- Assim como suas características;
- Estático;
- Potencialmente difícil de implementar
  - e.g. hiperperíodo calculado manualmente.
- Off-line scheduling;
- Duas maneiras:
  - Síncrona: tarefas periódicas;
  - Assíncrona: tarefas aperiódicas.
- Não precisa nem do SO:
  - Cada job conhece seu sucessor;
  - Um timer é inicializado com o release time do próximo job.
  - O job atual executa, na hora do próximo ele é acordado automáticamente.
- Útil principalmente em aplicações críticas.

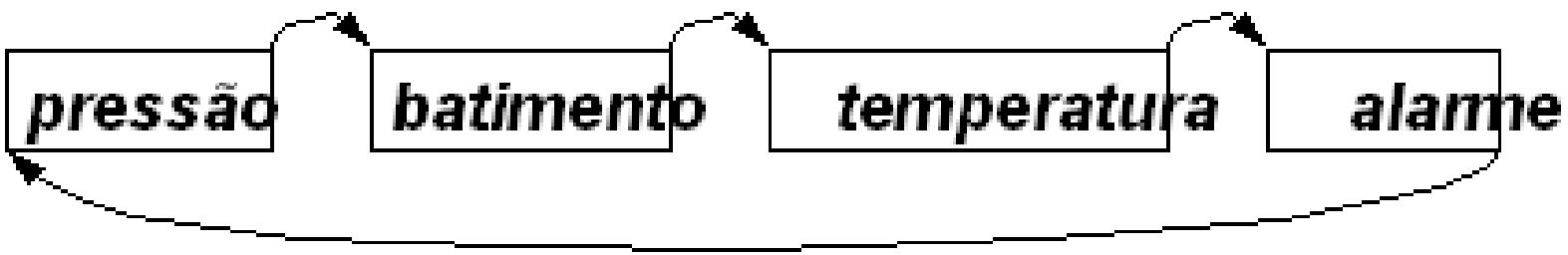
Exemplo: Monitor médico

Tarefas:

- medir pressão;
- medir batimento cardíaco;
- medir temperatura;
- checar condições de alarme.

Ordem de execução:

- pressão, batimento, temperatura, alarme.
- A soma dos tempos de execução é menor que o tempo necessário para bater as botas.
- Tempo de resposta: soma das execuções



Weighted Round-Robin

Round-robin aonde cada job tem um multiplicador m.

- Cada job j_i recebe m_i quantuns a cada instância

Desta forma podemos priorizar alguns processos.

Pode aumentar a eficiência em alguns casos, mas não em todos:

- Se um job precisa do resultado final do outro não ajuda;
- Mas se um job pode usar dados intermediários pode ser vantajoso:
  - Por exemplo, redes de alta velocidade;
  - Filas de prioridade rápidas são caras;
  - Mas filas round-robin não;
  - weighted round-robin implementa priori-dades sem filas de prioridade.

Prioridades

Event driven: tenta alocar os recursos de forma ótima quando eles são necessários.

- Guloso: otimizações locais
- Todo recurso que pode ser utilizado será pelo job mais importante que o quiser.

Jobs têm prioridades — definem quem é mais importante

- Aqueles com maior prioridade executam antes.
- Simples não ? Por isto algoritmos de escalonamento por prioridades são definidos pelas prioridades que atribuem aos jobs.
- Pensando bem, os seguintes algoritmos podem ser vistos como escalonadores de prioridade ?
  - FIFO
  - SJF
  - Round-robin ?

Exemplo: Escalonamento com Prioridades

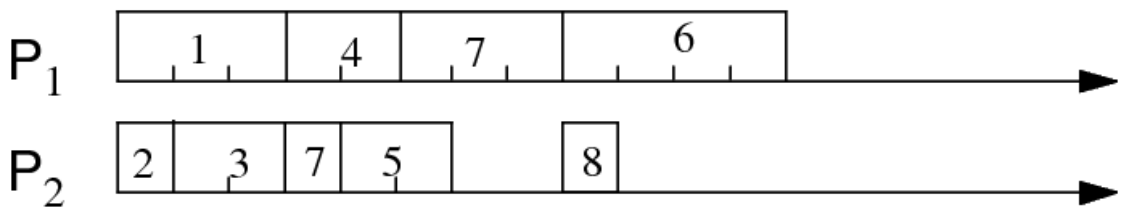
Prioridades: $J_1 > J_2 > J_3 > \ldots > J_8$

Tempos de execução: $J_1 = 3; J_2 = 1; J_3 = J_4 = J_5 = 2;$
$J_6 = J_7 = 4; J_8 = 1$

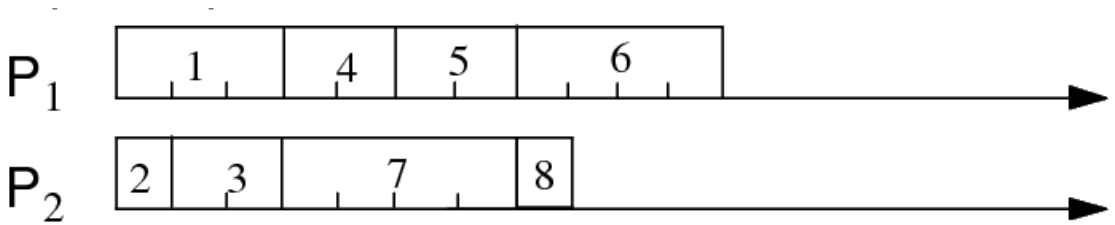
Release times: $J_i = 0, i \neq 5; J_5 = 4$

Precedências: $J_2 \rightarrow \{J_3, J_6\}; J_3 \rightarrow J_4; J_5 \rightarrow \{J_6, J_8\}$ $J_7 \rightarrow \{J_6, J_8\}$

Preemptivo:



Não preemptivo:



Earliest Deadline First

Escalonamento através da regra:

- Jobs são mais prioritários quando estão mais perto de suas deadlines.

Algoritmo ótimo:

- Se preempção é permitida e não existe contenção por recursos, EDF produz um escalonamento factível **sse** existe um.
- Prioridades dinâmicas:
  - Possivelmente difíceis de implementar;
  - Comportamento difícil de prever.

Sim, mas é ótimo mesmo ?
Optimalidade (Optimalitude, Optimalisse) não resiste muito. EDF não é ótimo se:

- Não houver preempção;
- Houver mais de um processador.

Priority x Clock Driven

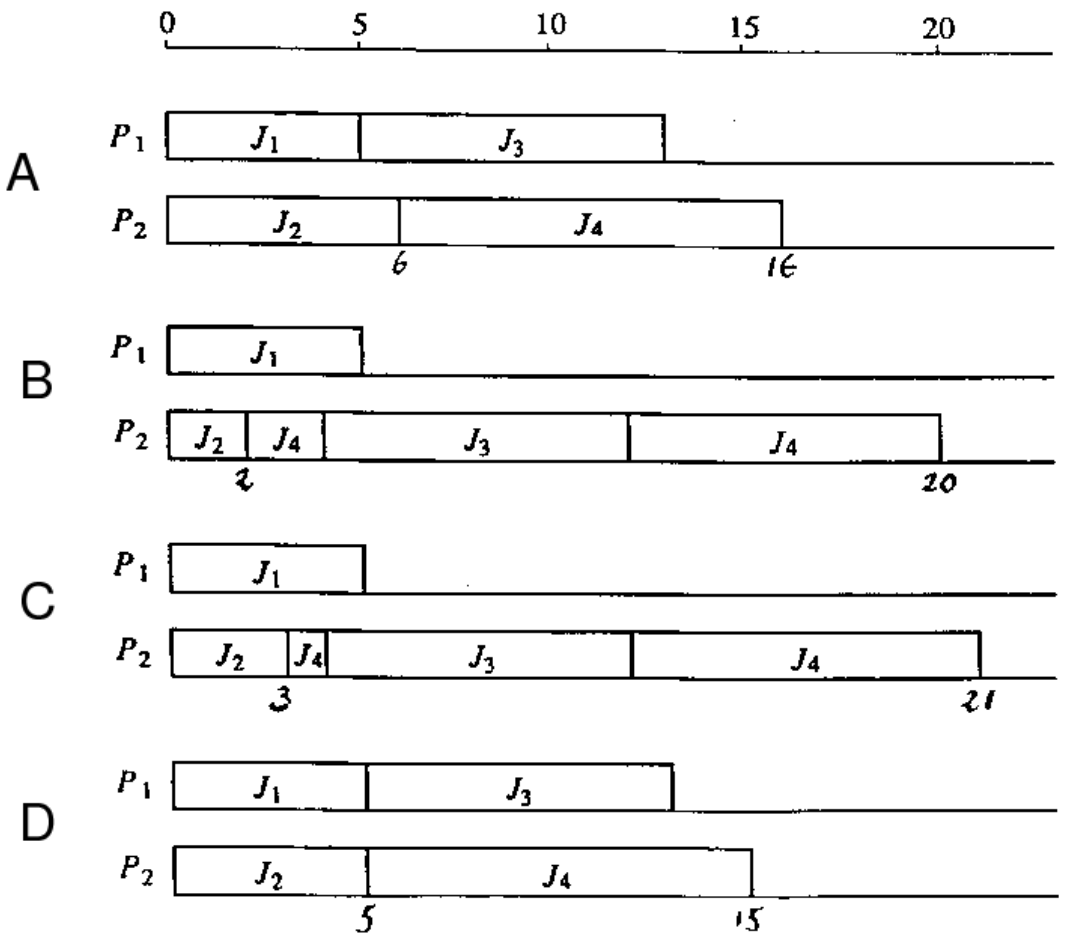
Prioridades:

- Mais fácil de implementar: algoritmos de prioridades são simples.
- Mais flexível;
- Comportamento difícil de prever.

Clock driven:

- Implementação difícil: escalonamento tem que ser feito todo antes.
- Contudo validação é mais simples
- Bastante usado por isto.

Imprevisibilidade



	R	D	E
J_1	0	10	5
J_2	0	10	[2,6]
J_3	4	15	8
J_4	0	20	10

$J_1 > J_2 > J_3 > J_4$