Sumário.

- O problema do escalonamento.
- Classificação.
- Algoritmos.

SDAI (Aula 22) Escalonamento

O problema do escalonamento.

- O sistema deve executar um conjunto de tarefas, cada uma com a sua meta temporal.
- Cada tarefa necessita de aceder a recursos partilhados:

  Processador.

  Rede.

  Memória.
- O escalonamento consiste em definir uma atribuição destes recursos às tarefas de modo a que as metas sejam cumpridas.

©Luís E. T. Rodrigues, 1996

SDAI (Aula 22) Escalonamento

Classificação.

- Estrito (hard):

  Todas as metas garantidas previamente para todos os cenários antecipados.

  Para tal é imprescindível ter os recursos necessários.
- Lato (soft):

  Podem falhar-se metas ocasionalmente.

  É possível operar sem todos os recursos necessários.

  Soluções menos dispendiosas.

©Luís E. T. Rodrigues, 1996

SDAI (Aula 22) Escalonamento

Tempo-real estrito: dinâmicos.

- Toma decisões em tempo de execução com base nas tarefas que existem para executar.
- São flexíveis.
- Dificuldade: calcular o escalonamento pode ser substancialmente demorado.

©Luís E. T. Rodrigues, 1996

Tempo-real estrito: estáticos.

- Calcula o escalonamento antes da execução.
- Gera uma tabela de despacho em tempo de compilação.
- Necessita de informação completa acerca das tarefas (restrições de exclusão mútua, tempo de execução máximo, metas).
- Baixa sobrecarga em tempo de execução.

Preemptivo e não-preemptivo.

- Preemptivo:

  Uma tarefa pode ser interrompida caso exista uma tarefa mais urgente para executar.

  Sujeito a restrições de exclusão mútua.

- Não-preemptivo:

  Uma tarefa não é interrompida a não ser que voluntariamente liberte o processador.

  Utilizado quando existem muitas tarefas de pequena dimensão.

Escalonamento centralizado e distribuído.

- Centralizado:

  Todas as decisões são realizadas num único nó.

  Ponto de falha.

- Distribuído:

  Algoritmo coopeerativo permite tomar decisões de modo distribuído.

  A execução deste algoritmo consome recursos e é um potencial ponto de estrangulamento.

Teste de "escalonabilidade"

- Teste para determinar se existe algum escalonamento possível (que cumpra as metas) para um determinado conjunto de tarefas.

  Suficiente.

  Exacto.

  Necessário.

- Escalonador óptimo:

  Se encontra um escalonamento quando um teste exacto indica que este existe.

Escalonador óptimo

- NP-complexo (não polinomial):

  Geralmente intratável computacionalmente.

- Teste de suficiência.

  Mais simples mas por vezes desperdiça recursos.

- Testes de necessidade.

  Podem ser simples mas não chegam.

Tipos de tarefas

- Periódicas:

  São executadas periódicamente, com um período conhecido à partida.

  percentagem de utilização $= \sum \text{consumo}_i/\text{período}_i \leq$ número de processadores

- Esporádicas:

  Deve ser conhecido pelo menos o tempo mínimo entre dois pedidos de execução dessas tarefas.

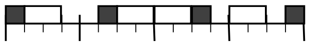
Escalonamento dinâmico óptimo

- Não é possível prever o futuro.
- Define-se óptimo como sendo:

  Um escalonador que encontra uma solução nos casos em que um escalonador com conhecimento do futuro encontraria.

- Quando exitem restrições de exclusão mútua esta tarefa é impossível.

Escalonamento dinâmico (exemplo)



Tarefa a negro deve ser executada imediatamente. Sempre possível quando se consegue prever o momento em que é lançada. Impossível de garantir no caso contrário.

Escalonamento dinâmico: rate monotonic (hipóteses).

H1: Todas as tarefas com metas estritas T_i são periódicas.

H2: Todas as tarefas são independentes (não existem restrições de exclusão mútua).

H3: As metas de T_i são iguais ao seu período.

H4: O tempo máximo de execução é conhecido e constante.

H5: O tempo de troca de contexto é desprezável.

H6: A percentagem de utilização está restringida a:

$$\mu = \sum C_i/P_i \leq n(2^{1/n} - 1) \rightarrow 0.7$$

Escalonamento dinâmico: rate monotonic

- Algoritmo:

  Prioridades fixas com base no período.

  A tarefa com menor período recebe a prioridade mais alta.

  As tarefas são escalonadas por ordem de prioridade.

  Algoritmo óptimo para sistemas com um processador.

Escalonamento dinâmico: earlier deadline first.

- Hipóteses

  H1-H5 do caso anterior.

- Algoritmo:

  Prioridades dinâmicas.

  Após cada evento, a tarefa com a meta mais próxima tem maior prioridade.

Escalonamento dinâmico: least-laxity.

- Hipóteses

  H1-H5 do caso anterior.

- Algoritmo:

  Prioridades dinâmicas.

  Após cada evento, a tarefa com menor intervalo entre o período e o tempo de computação ainda a effectuar tem maior prioridade.

Escalonamento de tarefas dependentes

- Se as tarefas são independentes:

  Podem ser executadas em processadores diferentes.

- Infelizmente, este não é geralmente o caso.

  Problema NP-complexo.

- Alternativas:

  Fornecer recursos adicionais só para executar o escalonador.

  Dividir o problema numa componente estática e numa componente dinâmica.

  Introduzir restrições ao comportamento das tarefas.

Aproximação

- Considerar que cada tarefa executa numa sucessão de quantums initerruptíveis:

  O acesso a regiões críticas faz-se dentro de cada quantum.

  A exclusão mútua é definida entre quantums.

  As restrições devem ser conhecidas de antemão.

Inversão de prioridade

- Ocorre quando uma tarefa de baixa prioridade detém o acesso a uma região e impede a execução de uma tarefa de elevada prioridade.
- Uma solução:

  Aumentar a prioridade da tarefa que detém o recurso de modo a esta não ser ultrapassada por tarefas de média prioridade.

  No limite todas têm a mesma prioridade e pode haver interblocagem.

Priority ceiling.

- Define-se tecto de uma região à maior prioridade de uma tarefa que pode aceder a essa região.
- Quando uma tarefa bloqueia outra mais prioritária, aumenta a sua prioridade para o nível da tarefa bloqueada até libertar o recurso.
- Uma tarefa só entra numa nova região se a sua prioridade é maior do que o tecto de todas as regiões em uso (não contando com a própria tarefa).

  Ideia chave: não vai impedir o progresso das tarefas que detém recursos a não ser que seja mais prioritária.

Escalonamento dinâmico distribuído

- Mais difícil devido ao acesso ao meio de comunicação.

  Uso do meio depende da ocorrência de faltas e da técnica de recuperação/mascaramento.

- Não existem soluções que ofereçam escalonamentos assegurados.
- Trabalho na área dos escalonadores de melhor esforço eficientes.

Escalonamento estático.

- Leitura do estado exterior em momentos pré-determinados.
- Sistema despoletado pelo tempo.

  De x em x unidades de tempo, o despacho é executado.

  A tabela de despacho é gerada em tempo de compilação.

- Procura em árvore:

  O escalonamento é encontrado usando técnicas de procura em árvore.

  O algoritmo requer conhecimento acerca das tarefas, períodos, tempos de execução, restrições de exclusão mútua, etc.

Construção da árvore de procura

- Cada nível corresponde a um intervalo de despacho.
- Raiz: nenhuma tarefa executando.
- Ramos: as diversas alternativas de despacho a partir desse nó.
- Profundidade: o período de escalonamento.
- Objectivo: encontrar um caminho da raiz até ao fundo da árvore que satisfaça todas as restrições.
- Problema NP-complexo: soluções heurísticas.

Aumentar a flexibilidade de escalonamentos estáticos

- Como considerar uma tarefa esporádica:

  Convertendo-a numa tarefa quasi-periódica .

  Servidor de tarefas esporádicas.

  Diferentes modos de operação.

Comparação.

- Predictabilidade:

  Naturalmente melhor nos escalonadores estáticos.

- Testabilidade.

  Também maior nos escalonadores estáticos.

- Utilização de recursos.

  Melhor no escalonador dinâmico.

- Extensibilidade.

  Complexa em ambos.

Resumo.

- Algoritmos dinâmicos.
- Algoritmos estáticos.
- Comparação.