Sistemas de Tempo Real:

Extensões da Análise para Prioridades Fixas

Rômulo Silva de Oliveira
Departamento de Automação e Sistemas - DAS – UFSC

romulo@das.ufsc.br
http://www.das.ufsc.br/~romulo
Maio/2010

Prioridade Fixa – Análise do Tempo de Resposta

- O tempo máximo de resposta de Ti é Ri = Ci + Ii

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil \times C_j$$

- Equação é recursiva
- Calculada através de iterações sucessivas, até:
  - Tempo de resposta passar do deadline
  - Resultado convergir, iteração x+1 igual a iteração x

$$w_i^{x+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^x}{P_j} \right\rceil \times C_j \qquad w_i^0 = C_i$$

Prioridade Fixa – Análise do Tempo de Resposta

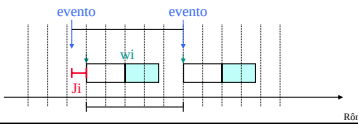
- Equação supõe um conjunto de premissas (modelo de tarefas)
  - Tarefas são periódicas ou esporádicas
  - D <= P
  - Tarefas são independentes
  - Não existe qualquer tipo de overhead do sistema
  - Qualquer atribuição de prioridade fixa pode ser usada
  - A tarefa está apta no momento que chega (início do período)
  - Não existem recursos compartilhados
  - Tarefas podem ser preemptadas a qualquer momento
- Este modelo de tarefas é simples
- Não corresponde à realidade da maioria dos sistemas
- Como extender a análise do tempo de resposta para lidar com modelos de tarefas mais realistas (mais complexos) ?

Modelo de Tarefas Extendido

- Release Jitter
- Bloqueio
- Pontos de Preempção
- Tolerância a Faltas
- Chaveamento de Contexto
- Deadlines arbitrários
- Atribuição Ótima de Prioridades

Release Jitter

- Suponha uma tarefa periódica
  - Início do período é indicado por interrupção do timer em hardware
  - Mas interrupções ficam desabilitadas por um tempo x
  - Haverá um atraso de x entre chegada e liberação da tarefa
- Suponha uma tarefa esporádica liberada por evento externo
  - Eventos podem ser amostrados periodicamente
  - Liberação da tarefa será atrasada em relação à ocorrência do evento
- **Release Jitter:** Atraso entre a chegada da tarefa e a sua liberação
  - Liberação significa inclusão na fila de aptos, passa a disputar processador



Release Jitter

- Seja Ji o release jitter máximo que a tarefa Ti pode experimentar
- Como incluir este termo nas equações do tempo de resposta ?

$$R_i = J_i + w_i$$

$$w_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i + J_j}{P_j} \right\rceil \times C_j$$

Bloqueio

- Podem ocorrer bloqueios devido a relações de exclusão mútua
  - Estruturas de dados compartilhadas
  - Dispositivos compartilhados
- Suponha T1 e T2, T1 com maior prioridade
- Se T2 fica esperando por T1
  - Ok, T1 tem mesmo prioridade superior, é intereferência normal
- Se T1 fica esperando por T2, é dito que T1 foi **bloqueada** por T2
- Cálculo do tempo de resposta deve incluir o tempo total de bloqueio máximo Bi

$$w_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i + J_j}{P_j} \right\rceil \times C_j$$

$$R_i = J_i + w_i$$

Bloqueio

- Interferência:
  - Tarefa de alta prioridade atrapalha a tarefa de baixa prioridade
- Bloqueio
  - Tarefa de baixa prioridade atrapalha a tarefa de alta prioridade
- Como calcular Bi ?
  - Complexo
  - Depende de como os recursos são gerenciados
  - Existem protocolos específicos para sistemas de tempo real
  - Aula específica sobre isto

Pontos de Preempção

- Escalonamento preemptivo exige mecanismos de sincronização
  - Proteção de seções críticas com mutexes, etc
- Possível usar escalonamento não preemptivo, mas
  - Tarefas longas vão atrasar todas as demais
- Solução intermediária: Pontos de Preempção
  - Também chamado de "Preempção Cooperativa" ou "Preempção Postergada"
- Código é em geral não preemptivo
- Programador informa pontos onde é seguro chavear o contexto
- Exclusão Mútua automática entre pontos de preempção
  - Para um único processador
- Preserva cache da tarefa executando, facilita cálculo do WCET

Pontos de Preempção

- Uma tarefa T_i não sofre interferência durante o seu último segmento de código (duração F_i)
- Mas este último segmento de código precisa ser executado
- Ela sofre bloqueio B_{MAX} devido a segmento não preemptável de tarefa de mais baixa prioridade

$$w_i^{n+1} = B_{MAX} + C_i - F_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{P_j} \right\rceil C_j$$

$$R_i = w_i^n + F_i$$

Tolerância a Faltas

- Quando uma falta é detectada
  - É possível executar um tratador de exceção ou um bloco de recuperação
  - Trazer o sistema para um estado correto, tolerando assim a falta
  - Mas isto significa computações extras
- Em sistemas de tempo real tolerantes a faltas
  - deadlines precisam ser garantidos
  - mesmo na presença de um certo nível de faltas
- O nível de tolerância a faltas é conhecido como "modelo de faltas"
- Seja o tempo de computação extra requerido pelo tratador de exceção de uma falta cometida pela tarefa Ti designado por C_i^f
- Seja `hep(i)` o conjunto de tratadores de exceção com prioridade igual ou superior à tarefa Ti

Tolerância a Faltas

- Seja o tempo de computação extra requerido pelo tratador de exceção de uma falta cometida pela tarefa Ti designado por C_i^f
- Seja `hep(i)` o conjunto de tratadores de exceção com prioridade igual ou superior à tarefa Ti
- Tempo de resposta caso apenas uma falta seja tolerada

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j + \max_{k \in hep(i)} C_k^f$$

Tolerância a Faltas

- Se F é o número de faltas toleradas

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j + \max_{k \in hep(i)} FC_k^f$$

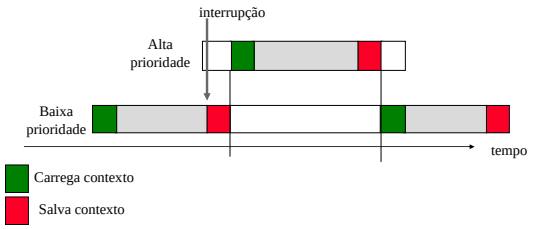
Tolerância a Faltas

- Se existe um intervalo mínimo I_f entre a ocorrência de faltas

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{P_j} \right\rceil C_j + \max_{k \in hep(i)} \left(\left\lceil \frac{R_i}{I_f} \right\rceil C_k^f \right)$$

Chaveamento de Contexto

- Toda tarefa precisa incluir no seu C a carga do contexto
- Se a tarefa Ti é preemptada por outra
  - Interferência sofrida por Ti inclui o tempo de chaveamento de contexto
  - Na verdade dois chaveamentos de contexto



Deadlines Arbitrários

- Lidam com situações onde D (e possivelmente R) > P
- Ativação [q] de Ti pode ser atrapalhada (sofrer interferência) da ativação [q-1]
  - A qual ainda não terminou quando a ativação [q] é liberada
- Job é usado como sinônimo de ativação

- Não é possível saber a princípio se o pior caso da tarefa vai ocorrer no job 0, no job 1, no job 2, etc
- Por exemplo, se Ci > Pi, os tempos de resposta crescem para sempre
- Mas se Ri<Pi, o pior caso acontece no job 0

Deadlines Arbitrários

$$w_i^{n+1}(q) = B_i + (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n(q)}{P_j} \right\rceil C_j$$

$$R_i(q) = w_i^n(q) - qP_i$$

- O número de liberações é limitado pelo menor valor de q para o qual a seguinte relação é verdadeira $R_i(q) \le P_i$
  - A ativação q+1 não sofrerá interferência da ativação q pois esta acabou antes de Pi
- O tempo de resposta no pior caso é então o máximo valor encontrado para cada q

$$R_i = \max_{q=0,1,2,\ldots} R_i(q)$$

Deadlines Arbitrários

- A inclusão de release jitter é similar ao caso D <= P

$$w_i^{n+1}(q) = B_i + (q+1)C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n(q) + J_j}{P_j} \right\rceil C_j$$

$$R_i(q) = w_i^n(q) - qP_i + J_i$$

Atribuição Ótima de Prioridades

- Com as extensões do modelo
  - DM não é mais ótimo
- É possível provar que:
  - Se uma tarefa T recebe a menor prioridade e é escalonável então,
  - Se uma ordenação escalonável de prioridades existe para todo o conjunto de tarefas
  - Uma ordenação existe com a tarefa T recebendo a prioridade mais baixa

```
for k in 1..N {
  for x in k..N {
      Swap(Set, k, x);
      Process_Test(Set, k, ok);
      if(ok) break;
      }
  if(!ok) return –1; //não encontrou solução
  }
```


Resumo

- Análise do tempo de resposta para prioridades fixas pode ser extendida
- Objetivo é analisar modelos de tarefas mais realistas
  - Mais complexos
- Foram citadas extensões para lidar com
  - Release jitter
  - Bloqueio
  - Pontos de Preempção
  - Tolerância a Faltas
  - Chaveamento de Contexto
  - Deadlines arbitrários
  - Atribuição Ótima de Prioridades