Análise Temporal da Implementação da Fila de Aptos como Lista Ordenada e Lista Desordenada

Ricardo Bacha Borges, Rômulo Silva de Oliveira
LCMI - DAS
Universidade Federal de Santa Catarina
{rbb,romulo}@das.ufsc.br

Introdução - Sistemas de Tempo Real

- Caracterizados pela existência de requisitos de natureza temporal explícitos e não triviais em sua especificação
- Aspectos temporais não estão limitados a questões de desempenho, mas sim à própria utilidade do sistema
- Certas tarefas possuem deadlines
- Não atendimento do deadline pode resultar:
  - em catástrofes (sistema crítico, *hard deadline*)
  - na inutilidade do sistema desenvolvido (sistema crítico à missão, *firm deadline*)

Introdução – Sistemas Operacionais

- Aplicações de tempo real são mais facilmente construídas com os serviços de um sistema operacional
- Porém, ...
- Tanto a aplicação como o SO compartilham os mesmos recursos do hardware
- Comportamento temporal do SO afeta o comportamento temporal da aplicação
- Por exemplo, considere a rotina que trata as interrupções do temporizador em hardware:
  - Projetista da aplicação pode ignorar a função desta rotina,
  - mas não pode ignorar o seu efeito temporal

Introdução – Ready Queue

- A **fila de aptos (*ready queue*)** é uma estrutura de dados quase sempre usada, em sistemas operacionais
  - Mantém a fila das tarefas que estão esperando para executar
- Existem muitas formas de implementar a fila de aptos
- Forma clássica: lista encadeada unindo os descritores das tarefas
- Tipicamente: lista encadeada é ordenada conforme a prioridade
- Implementação alternativa: lista encadeada desordenada
  - liberação de uma tarefa implica em uma inserção em qualquer lugar da lista
  - seleção da próxima tarefa a executar implica em percorrer toda a lista para localizar a tarefa com maior prioridade

Objetivo do artigo

- **Objetivo: Análisar temporalmente a implementação da fila de aptos**
- A mesma análise é realizada para as implementações como
  - lista ordenada
  - lista desordenada
- A análise mostra que, para sistemas de tempo real, pode ser melhor manter a fila de aptos desordenada
- Este é um resultado não intuitivo
  - possível através de uma modelagem algébrica do comportamento

Arquitetura de Software Considerada

- Cada tarefa do sistema é representada por um descritor
- Fila de aptos é implementada como uma lista encadeada
- Os próprios descritores de tarefa são utilizados como elementos da lista
- Escalonamento preemptivo baseado em prioridades fixas
  - O sistema operacional não altera as prioridades das tarefas definidas pelo projetista da aplicação
- Tarefas da aplicação são periódicas ou esporádicas
- Esperádicas: apresentam um intervalo mínimo de tempo entre cada duas liberações consecutivas
- Periódicas: espera até o próximo instante de liberação é feita através de uma função de temporização (*sleep*)

Arquitetura de Software Considerada

- A função de temporização (*sleep*) utiliza uma lista encadeada de descritores de espera
- Esta lista é mantida ordenada conforme o instante de acordar
- A primeira tarefa a ser acordada é indicada pelo primeiro descritor desta lista
- **Temporizador em hardware** opera em **"modo de disparo único"** (*single-shot*)
- Temporizador em hardware é programado para gerar uma interrupção apenas no próximo instante de interesse

Análise Temporal - Premissas

- Prioridade fixa é associada a cada tarefa antes da execução
- Por exemplo: taxa monotônica ou deadline monotônico
- Testes de escalonabilidade em tempo de projeto calculam o tempo máximo de resposta para cada tarefa
- Compara-se o tempo de resposta com o deadline

Análise Temporal - Premissas

- Sistema formado por N tarefas
- Cada tarefa Ti é periódica com período Pi
  - ou esporádica com intervalo mínimo entre ativações Pi
- Cada tarefa Ti possui
  - prioridade fixa
  - tempo máximo de computação Ci
  - deadline Di, Di≤Pi
  - variação na liberação máxima (*release jitter*) Ji
- Em cada ativação da tarefa Ti ela pode ser bloqueada por tarefas de menor prioridade no máximo Bi
- O conjunto HP(i) é formado por todas as tarefas com prioridade maior ou igual a prioridade da tarefa Ti.

Análise Temporal – Equação Básica

- Nestas condições, temos que:
  - Burns, A., Tindell, K., Wellings, A. (1995) “Effective Analysis for Engineering Real-Time Fixed-Priority Schedulers” IEEE Trans. on Soft. Eng., Vol. 21, pages 475-480.

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

$$R_i = J_i + W_i$$

Modelagem da Arquitetura Considerada

- As execuções do tratador de interrupções do *timer* precisam ser modeladas
- Cada tarefa da aplicação dá origem a uma pseudo-tarefa
  - associada com a execução do tratador de interrupções do *timer* quando ele acorda a tarefa da aplicação em questão
- **Para cada tarefa Ti existe uma pseudo-tarefa Hi**, cujo período P(Hi) é o mesmo período da tarefa Ti
- O tempo de execução no pior caso C(Hi) é o tempo que o tratador de interrupções do *timer* demora, no pior caso, quando a tarefa Ti é acordada
- A prioridade de qualquer tarefa Hi será sempre maior que a prioridade de qualquer tarefa Ti (Hi representa o tratador de interrupções do *timer)*

Modelagem da Arquitetura Considerada

- Vamos considerar Bi=0 para todas as tarefas
- Toda execução da tarefa Ti deve necessariamente ser precedida por uma execução de Hi
  - tratador da interrupção de timer que libera a tarefa Ti
- Isto pode ser modelado como Ji = C(Hi)
- Qualquer período de interrupções desabilitadas representa um possível atraso de liberação para os tratadores de interrupções
  - Desta forma, temos que J(H1)=J(H2)=J(H3)=J(H4)=J(H5)=JH

Modelagem da Arquitetura Considerada

- Quando uma tarefa da aplicação conclui sua execução
  - o processador fica livre
  - necessário selecionar a próxima tarefa para execução
- Pode-se modelar isto substituindo Cj por CAj
- CAj representa
  - o tempo de execução no pior caso da tarefa Tj
  - acrescido pelo tempo necessário para escolher o seu sucessor na fila de aptos (varia se fila ordenada ou desordenada)

Modelagem da Arquitetura Considerada

- Cada tarefa Ti será afetada por Hk da seguinte forma:
- Possivelmente várias vezes se $k < i$
  - Tk executa várias vezes enquanto Ti executa uma
- Nenhuma vez se $k = i$
  - o efeito de Hk está modelado pelo Ji
- No máximo uma única vez se $k > i$
  - Tk não pode executar antes de Ti terminar

Modelagem da Arquitetura Considerada

- Pode-se reescrever as equações do tempo de resposta em função da modelagem da arquitetura
  - PHP(i), Pseudo Higher Priority:
    conjunto de tarefas {Hk | k < i}
  - PLP(i), Pseudo Lower Priority:
    conjunto de tarefas {Hk | k > i}

$$W_i = C_i + \sum_{j \in HP(i)} \left\lceil \frac{J_j + W_i}{P_j} \right\rceil \times CA_j + \sum_{k \in PHP(i)} \left\lceil \frac{J_H + W_i}{P_{Hk}} \right\rceil \times C_{Hk} + \sum_{k \in PLP(i)} C_{Hk}$$

$$R_i = J_i + W_i$$

Modelagem – Lista Ordenada

Pseudo-tarefa	Período	Tempo de execução
H1	P(H1) = P1	C(H1)=remove_p+insere(0)
H2	P(H2) = P2	C(H2)=remove_p+insere(1)
H3	P(H3) = P3	C(H3)=remove_p+insere(2)
H4	P(H4) = P4	C(H4)=remove_p+insere(3)
H5	P(H5) = P5	C(H5)=remove_p+insere(4)

- Quando uma tarefa da aplicação conclui sua execução
  - processador fica livre
  - necessário selecionar qual será a próxima tarefa a executar
  - CAj = Cj + remove_p.
- No pior caso as interrupções poderão ficar desabilitadas
  - JH = remove_p+insere(N-1)

Modelagem – Lista Desordenada

Pseudo-tarefa	Período	Tempo de execução
H1	P(H1) = P1	C(H1)=remove_p +insere_p
H2	P(H2) = P2	C(H2)=remove_p +insere_p
H3	P(H3) = P3	C(H3)=remove_p +insere_p
H4	P(H4) = P4	C(H4)=remove_p +insere_p
H5	P(H5) = P5	C(H5)=remove_p +insere_p

- Quando uma tarefa da aplicação conclui sua execução
  - processador fica livre
  - necessário selecionar qual será a próxima tarefa a executar
  - CAj = Cj + remove(N-j)
- No pior caso as interrupções poderão ficar desabilitadas
  - JH = remove(N-1)

Modelagem - Comentários

- Lista ordenada
  - Operação de escolha da próxima tarefa é rápida (usa a primeira)
  - Tarefa mais prioritária está executando,
    interrupções do *timer* liberam tarefas menos prioritárias
  - Pára temporariamente a execução da tarefa mais prioritária

- Lista desordenada
  - Inserção desordenada é mais rápida
  - Reduz interferência sofrida pela tarefa de mais alta prioridade
  - Preço a ser pago é um maior tempo para escolher sucessor
  - É necessário percorrer toda a fila de aptos para decidir qual a próxima tarefa a executar

Exemplo Numérico

Operação	Tempo em microsegundos
Insere_p	0.9
Insere(x)	0.9 + 0.6 × x
Remove_p	0.7
Remove(x)	0.7 + 0.7 × x

Tarefa	Período (uS)	Execução (uS)
T1	50	4
T2	50	10
T3	300	30
T4	500	50
T5	500	50

Tarefa	**Tempo de resposta (uS) Lista Ordenada**	**Tempo de resposta (uS) Lista Desordenada**
T1	**18.0**	**12.0**
T2	**28.7**	**25.5**
T3	**78.6**	**81.8**
T4	**148.5**	**180.9**
T5	**237.6**	**279.3**

Conclusões

- Foi apresentada a modelagem temporal da implementação da fila de aptos como lista ordenada e como lista desordenada
- Foram construídas equações que permitem calcular o tempo de resposta das tarefas da aplicação a partir das características básicas do sistema
- Lista desordenada diminui o tempo de resposta no pior caso das tarefas de mais alta prioridade
- Preço a ser pago é um maior tempo de resposta para as tarefas de prioridade mais baixa
- No caso de sistemas exigentes
  - fila de aptos pode ser implementada com estruturas de dados mais complexas, como discutido na seção 2 do artigo