Sistemas de Tempo Real:

Abordagens de Escalonamento

Rômulo Silva de Oliveira
Departamento de Automação e Sistemas – DAS – UFSC
romulo@das.ufsc.br
http://www.das.ufsc.br/~romulo
Maio/2009

Necessidade de Diferentes Abordagens

- Mercado para tempo real é amplo
- Sistemas de tempo real variam enormemente
  - Sistema de emergência em usina petroquímica
  - Controle de temperatura do freezer
  - Videogame
- Principais variações:
  - Crítico ou não crítico
  - Carga estática ou dinâmica
  - Importância associada com cumprimento dos deadlines
- Diferentes abordagens são necessárias

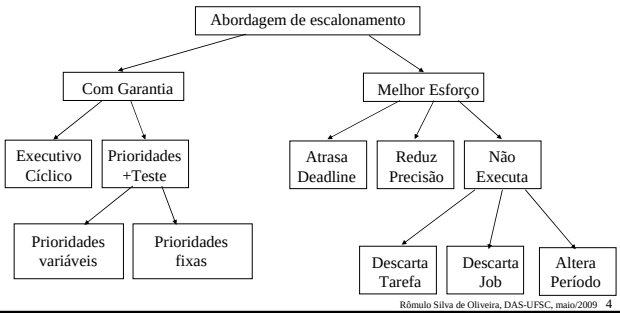
Abordagem Síncrona

- Supõe que a velocidade do computador é infinita
- Logo, tempo de computação é zero
- Tempo de resposta é zero
- Tudo é instantâneo
- Todos os deadlines estão automaticamente atendidos
- Não existe a necessidade de escalonamento

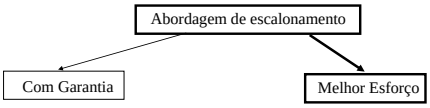
- Velocidade do computador sempre será finita, porém
  - Se o computador for muito mais rápido que o ambiente que estabelece os requisitos temporais
  - É possível assumir que a premissa é verdadeira
  - Exemplo: relógio despertador, processo químico, térmico

Classificação das Abordagens de Escalonamento

- Abordagens básicas para o escalonamento tempo real
  - Visão Assíncrona



Abordagens com Melhor Esforço



- Não existe garantia que os deadlines serão cumpridos
- O "Melhor Esforço" será feito neste sentido
- Capaz de fornecer análise probabilista
  - Simulação, teoria das filas de tempo real, etc
- Algumas abordagens oferecem Garantia Dinâmica
  - Garante o deadline (ou não) no início da ativação do job

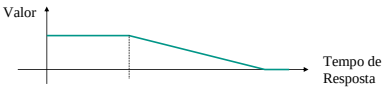
Abordagens com Melhor Esforço

- Existe a possibilidade de Sobrecarga (*overload*)
- Sobrecarga:
  - Não é possível cumprir todos os deadlines
  - Não é uma falha do projeto
  - É uma situação natural uma vez que não existe garantia antecipada
- Vantagens desta abordagem
  - Não é necessário conhecer o pior caso
  - Sistemas mais baratos, não são projetados para o pior caso
  - Não é necessário conhecer a carga exatamente
- Desvantagens
  - A princípio qualquer deadline poderá ser perdido

- Questão fundamental: Como tratar a sobrecarga ?

Perda de Deadlines na Sobrecarga

- Em sobrecarga ATRASA algumas tarefas
- Baseado em função tempo-valor (time-value function)
- Cada tarefa possui uma função que
  - Indica o valor da tarefa para o sistema em função do seu instante de conclusão
- Objetivo é maximizar o valor total do sistema
  - Valor do sistema é o somatório do valor das tarefas executadas
  - Tarefas podem possuir importância (peso) diferentes



Redução da Precisão na Sobrecarga

- Em sobrecarga DIMINUI a precisão de algumas tarefas
- Objetivo é "fazer o trabalho possível dentro do tempo disponível"
- Exemplos:
  - Ignorar bits menos significativos de cada pixel
  - Trabalhar com amostras de áudio menos precisas
  - Alterar resolução e tamanho de imagem na tela
  - Simplificar animações
  - Usar algoritmos de controle mais simples
  - Interromper pesquisa em algoritmos de inteligência artificial
- Aparece associada com a técnica de **Computação Imprecisa** (Imprecise Computation)

Computação Imprecisa

- Cada tarefa dividida em
  - Parte obrigatória (mandatory part)
  - Parte opcional (optional part)
- Parte Obrigatória gera resultado com qualidade mínima
- Parte Opcional refina resultado até qualidade máxima
- Podem existir tarefas com
  - apenas parte obrigatória ou
  - apenas parte opcional
- Situações normais: executa as duas partes de cada tarefa
- Sobrecarga: executa apenas a parte obrigatória
- Objetivo é maximar a qualidade total da aplicação, cumprindo os deadlines

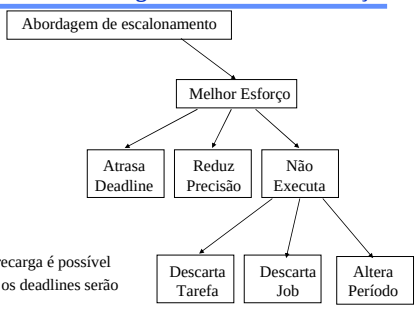
Descarte de Tarefas na Sobrecarga

- Em sobrecarga NÃO EXECUTA algumas tarefas
- As tarefas executadas cumprem o deadline
  - Mais apropriado para tarefas com deadline firm
- Pode cancelar:
  - Ativações individuais (jobs)
  - Tarefas completas
- Objetivo é maximizar o número de tarefas executadas
- Tarefas podem ter importância (peso) diferentes
  - Maximiza o somatório dos pesos das tarefas executadas
- Algumas soluções utilizam um parâmetro de descarte **s**
  - Distância temporal entre ativações descartadas deve ser **s**
- Outras permitem o descarte de até **m** a cada **k** ativações

Alteração do Período na Sobrecarga

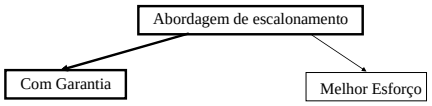
- Em sobrecarga aumenta o período de algumas tarefas
- Objetivo é não perder deadlines através da redução da utilização do processador que cada tarefa representa
- Exemplos:
  - Amostragem de variáveis em laço de controle
  - Taxa de apresentação dos quadros de um vídeo
  - Freqüência da atualização da tela do operador
- Chamado às vezes de **Rate Modulation**

Abordagens com Melhor Esforço



- Seja como for:
  - Nesta abordagem a sobrecarga é possível
  - Não existe garantia que os deadlines serão cumpridos

Abordagens com Garantia em Projeto



- Oferece previsibilidade determinista
- Análise feita em projeto
  - Carga precisa ser limitada e conhecida em projeto (Hipótese de Carga)
  - É Suposto um limite para faltas (Hipótese de Faltas)
- Dividida em duas partes:
  - Análise da escalonabilidade (cumprimento dos deadlines)
  - Construção da escala de execução

Abordagens com Garantia em Projeto

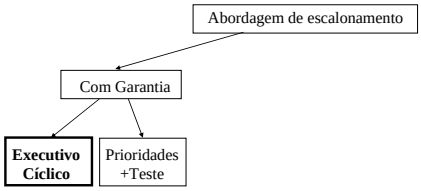
- Necessário conhecer o comportamento do sistema no pior caso, tanto software quanto hardware
- Isto significa
  - Pior fluxo de controle para cada tarefa
  - Pior cenário de sincronização entre tarefas (exclusão mútua, etc)
  - Piores dados de entrada
  - Pior combinação de eventos externos (interrupções, sensores, etc)
  - Pior comportamento das caches no hardware
  - Pior comportamento do processador (pipeline, barramentos, etc)
  - Pior tudo
- Necessário conhecer WCET de cada tarefa, C
  - Worst-case execution time (WCET)
  - Tempo de execução no pior caso

Abordagens com Garantia em Projeto

- **Vantagens**
  - Determina em projeto que todos os deadlines serão cumpridos
  - Necessário para aplicações críticas
  - Teoria serve de base para abordagens sem garantia
- **Desvantagens**
  - Necessário conhecer exatamente a carga
  - Necessário reservar recursos para o pior caso
  - Difícil determinar o pior caso em soluções COTS (commercial off-the-shelf)
  - Gera enorme sub-utilização de recursos

Abordagens com Garantia em Projeto

- Existem duas formas de obter garantia em projeto
  - Executivo cíclico
  - Prioridades + Teste de escalonabilidade

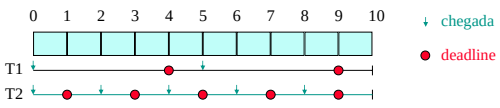


Executivo Cíclico

- Todo o trabalho de escalonamento é feito em projeto
- Resultado é uma **grade de execução** (*time grid*)
- Grade determina qual tarefa executa quando
- Garantia obtida através de uma simples inspeção da escala
- Durante a execução:
  - Pequeno programa lê a grade e dispara a tarefa aproprida
  - Quando a grade termina ela é novamente repetida
- Vantagem: Comportamento completamente conhecido
- Desvantagem: Escalonamento muito rígido, tamanho da grade
- Muito usado em aplicações embutidas (Embedded Systems)

Executivo Cíclico

- Restrições devem ser observadas na construção da grade
  - Período, tempo máximo de computação
  - Precedências, exclusões, etc
- Escalonamento é em geral não preemptivo
  - Facilita lidar com recursos compartilhados
  - Mais difícil achar uma escala satisfatória
- Exemplo:
  - T1: P1=5 C1=2 D1=4
  - T2: P2=2 C2=1 D2=1

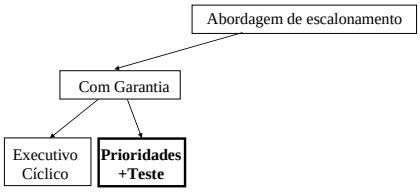


Executivo Cíclico

- Vantagens
  - É a forma tradicional para sistemas críticos
  - Comportamento completamente conhecido
  - Fácil detectar qualquer falha de projeto
  - Apropriado para tarefas periódicas, as quais são maioria nos sistemas críticos
- Desvantagens
  - Não lida bem com tarefas que não são periódicas
  - Tabela pode ficar grande, caso períodos não sejam múltiplos entre si
  - No caso de WCET mal calculado, o que fazer ?
  - Tarefas muito longas precisam ser quebradas em várias sub-tarefas

Abordagens com Garantia em Projeto

- Existem duas formas de obter garantia em projeto
  - Executivo cíclico
  - Prioridades + Teste de escalonabilidade



Prioridades + Teste de Escalonabilidade

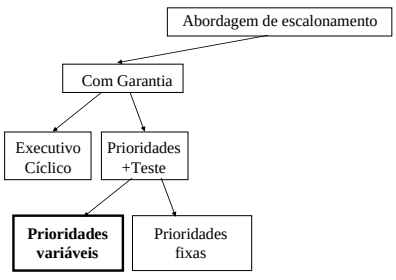
- Cada tarefa recebe uma prioridade
- Escalonamento em geral é preemptivo
- Teste realizado antes da execução determina escalonabilidade
  - Teste considera como são as tarefas (modelo de tarefas)
    - Periódica, esporádica, D<=P, bloqueios, etc
  - Teste considera forma como prioridades são atribuídas
  - Validade do teste é demonstrada como teorema
  - Complexidade do teste depende do modelo de tarefas
- Na execução:
  - Escalonador dispara as tarefas conforme as prioridades

Prioridades + Teste de Escalonabilidade

- Vantagens:
  - Suporta tarefas esporádicas com facilidade
  - Suporta tarefas aperiódicas com facilidade
  - Não é necessário gerar grade de tempo
  - Oferece determinismo para os deadlines
  - Comportamento no caso de falhas pode ser gerenciado
- Desvantagens:
  - Existem testes apenas para alguns modelos de tarefas
  - É difícil criar novos testes (significa provar um teorema)
  - Não oferece determinismo para a escala de execução
  - Mais difícil detectar faltas
- Usado em aplicações que exigem garantia mas também requerem flexibilidade na escala de execução

Prioridades + Teste de Escalonabilidade

- Existem dois tipos básicos de prioridades
  - Prioridades variáveis
  - Prioridades fixas

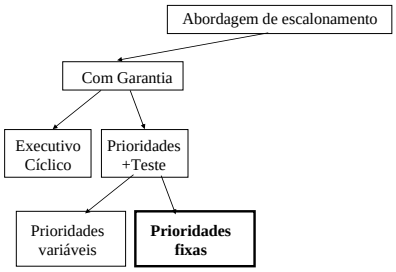


Prioridades Variáveis

- Earliest Deadline First – EDF
  - Prioridade mais alta para a tarefa com deadline absoluto menor
  - Prioridade variável
- Least Laxity First – LLF
  - Prioridade mais alta para a tarefa com menor folga
  - Folga = deadline absoluto – agora – computação que falta
  - Prioridade variável
- Outras

Prioridades + Teste de Escalonabilidade

- Existem dois tipos básicos de prioridades
  - Prioridades variáveis
  - Prioridades fixas



Prioridades Fixas

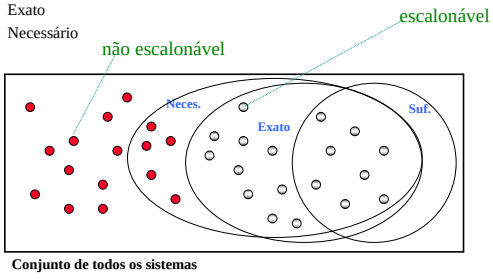
- Rate Monotonic – RM
  - Prioridade mais alta para a tarefa com período menor
  - Prioridade fixa
- Deadline Monotonic - DM
  - Prioridade mais alta para a tarefa com deadline relativo menor
  - Prioridade fixa
  - Igual ao RM quando D = P
- Outras

Prioridades + Teste de Escalonabilidade

- Apenas atribuir prioridades não basta
- É necessário um teste de escalonabilidade
- Testes podem ser classificados de diferentes formas
- Quando a cobertura
  - Teste suficiente mas não necessário (muito rigoroso)
  - Teste suficiente e necessário (exato)
  - Teste necessário mas não suficiente (muito frágil)
- Quanto ao tipo
  - Baseado em utilização
  - Baseado em tempo de resposta

Teste de Escalonabilidade: Cobertura

- Teste de Escalonabilidade pode ser
  - Suficiente
  - Exato
  - Necessário



Teste de Escalonabilidade: Utilização

- Utilização de uma tarefa:
  - Tempo máximo de computação dividido pelo período
  - Exemplo: T1 tem C1=12 e P1=50, então U1 = 12 / 50 = 0.24
- Utilização do sistema
  - Somatório da utilização de todas as tarefas
- Dado
  - Um modelo de tarefas
  - Uma política de atribuição de prioridades
- Existe um limiar de utilização para o processador, de tal sorte que:
  - Se a utilização do processador for menor que o limiar
  - Então jamais um deadline será perdido
- Limiar demonstrado como teorema

Teste de Escalonabilidade: Utilização

- Exemplo clássico [Liu & Layland 1973]
- Modelo de tarefas:
  - Tarefas periódicas, independentes
  - P=D
- Política de atribuição de prioridades
- Sistema é escalonável se:

$$\sum_{i=1}^{N} \left(\frac{C_i}{P_i}\right) < N\left(2^{1/N} - 1\right)$$

- Para N grandes utilização máxima tende para 69.3%
- Teste é suficiente mas não necessário

Teste de Escalonabilidade: Tempo de Resposta

- Dado
  - Um modelo de tarefas
  - Uma política de atribuição de prioridades
- Para cada tarefa Ti
  - Identifica o pior padrão de chegadas possível para Ti
  - Constrói o diagrama de tempo considerando o pior caso para tudo
  - Calcula o tempo de resposta da tarefa neste caso
  - Este é o tempo de resposta no pior caso Ri da tarefa Ti
- Se Ri <= Di
  - Então jamais Ti perderá um deadline
- Dificuldade: Calcular Ri quando o sistema é complexo e a escala de execução não determinista

Resumo

- Existe a necessidade de **diferentes abordagens** para o escalonamento tempo real
- Principal classificação é com respeito a **garantia**
- Abordagens com Melhor Esforço
  - Perda de deadlines
  - Redução da Precisão
  - Descarte de tarefas, de jobs, ajusta período
- Abordagens com Garantia em Projeto
  - Executivo Cíclico
  - Prioridades + Teste de Escalonabilidade