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

Abordagens de Escalonamento

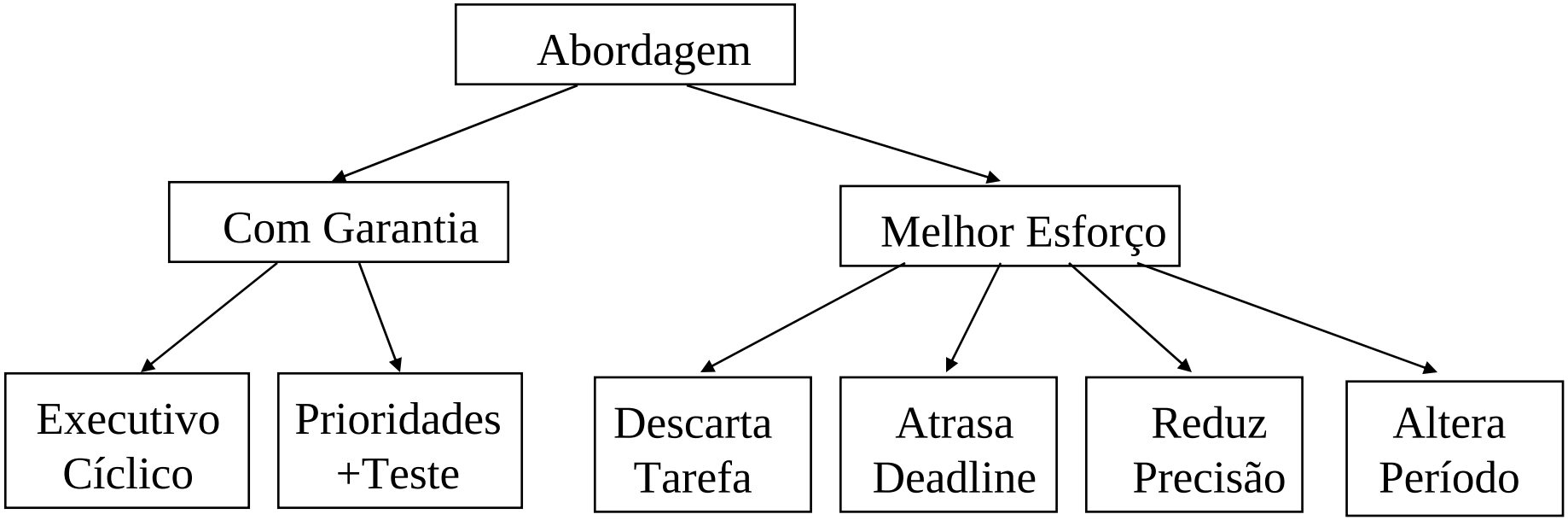
- Necessidade de Diferentes Abordagens
- Classificação das Abordagens
- Abordagens com Garantia em Projeto
- Executivo Cíclico
- Prioridades + Teste de Escalonabilidade
- Abordagens com Melhor Esforço
- Descarte de Tarefas
- Perda de Deadlines
- Redução da Precisão
- Alteração do Período

Necessidade de Diferentes Abordagens

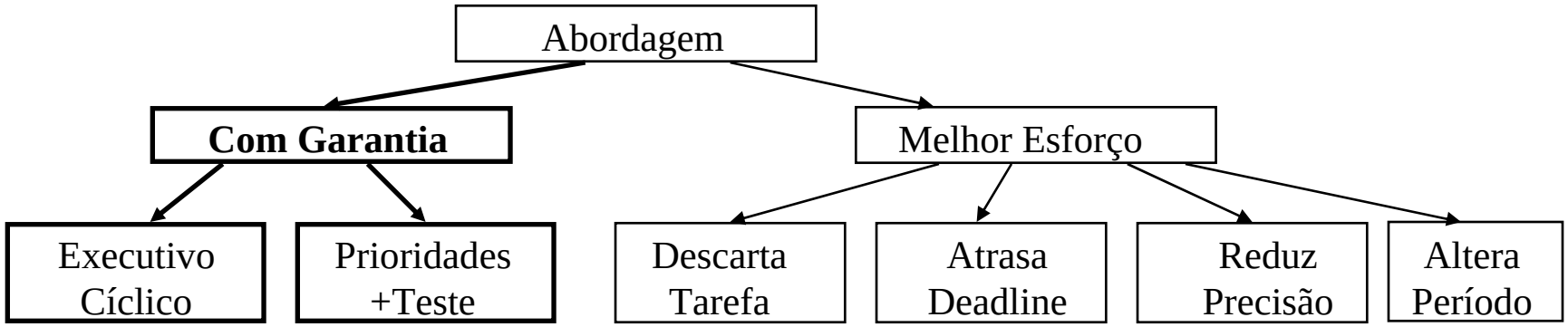
- Mercado para tempo real é amplo
- Sistemas de tempo real variam enormemente
  - Sistema de emergência em usina petroquímica
  - Controle de temperatura do freezer
  - Vídeo game
- Principais variações:
  - Crítico ou não crítico
  - Carga estática ou dinâmica
  - Importância associada com os deadlines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

Classificação das Abordagens

- Abordagens básicas para o escalonamento tempo real



Abordagens com Garantia em Projeto



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

Abordagens com Garantia em Projeto

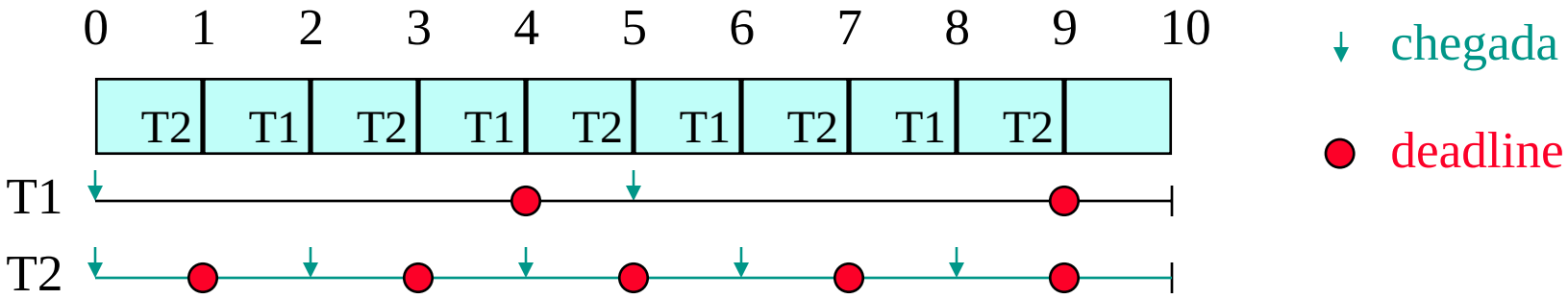
- **Vantagens**
  - Determina em projeto que todos os deadlines serão cumpridos
  - Necessário para aplicações críticas
  - Teoria serve de base para abordagens sem garantia
- **Desvantagens**
  - Necessário conhecer exatamente a carga
  - Necessário reservar recursos para o pior caso
  - Difícil determinar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

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 pode ser:
  - Preemptivo - Tarefa pode ser suspensa e depois retomada
  - Não Preemptivo - Depois que inicia tarefa vai até o fim
- Exemplo:
  - T1: P1=5 C1=2 D1=4
  - T2: P2=2 C2=1 D2=1



Prioridades + Teste de Escalonabilidade

- Cada tarefa recebe uma prioridade
- Prioridades podem ser fixas ou variáveis
- Escalonamento em geral é preemptivo
- Teste realizado antes da execução determina escalonabilidade
  - Teste considera forma como prioridades são atribuídas
  - Complexidade depende do modelo de tarefas
- Na execução:
  - Escalonador dispara as tarefas conforme as prioridades
- Vantagem: Suporta tarefas esporádicas, não tem grade
- Desvantagem: Testes limitados a alguns modelos de tarefas
- Usado em aplicações que exigem garantia mas são muito complexas para construir uma grade

Prioridades + Teste de Escalonabilidade

- Políticas mais utilizadas:
  - **Earliest Deadline First – EDF** (prioridade variável)
  - **Rate Monotonic – RM** (prioridade fixa)
  - **Deadline Monotonic – DM** (prioridade fixa)

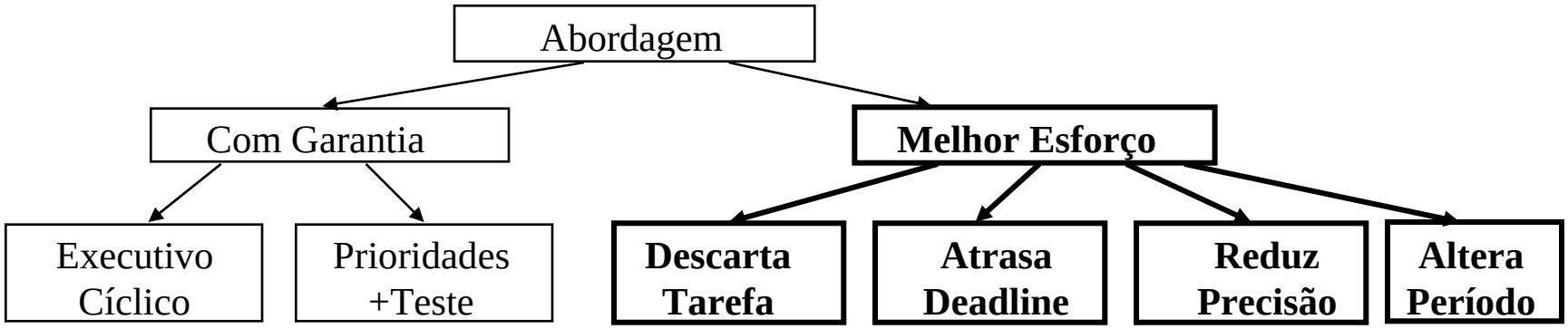
Prioridades + Teste de Escalonabilidade

- Exemplo usando RM
  - Utilização de uma tarefa:
    - Tempo máximo de computação dividido pelo período
    - T1 tem C1=12 e P1=50, então U1 = 12 / 50 = 0.24
  - Teste para Rate Monotonic, 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

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

Abordagens com Melhor Esforço

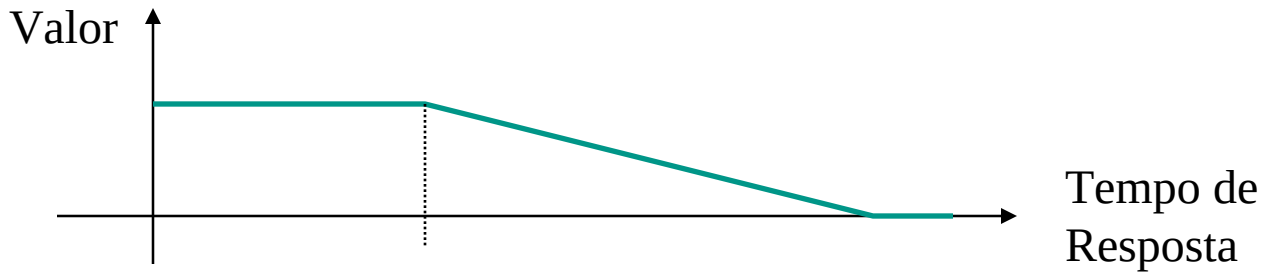
- Existe a possibilidade de Sobrecarga (*overload*)
- Sobrecarga:
  - Não é possível cumprir todos os deadlines
  - Situação natural uma vez que não existe garantia *off-line*
- Questão fundamental: Como tratar a sobrecarga ?
- Vantagens
  - Não é necessário conhecer o pior caso
  - Sistemas mais baratos, projetados para o caso médio
  - Não é necessário conhecer a carga exatamente
- Desvantagens
  - A princípio qualquer deadline poderá ser perdido

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
  - 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

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

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

Resumo

- Existe a necessidade de **diferentes abordagens** para o escalonamento tempo real
- Principal classificação é com respeito a **garantia**
- Abordagens com Garantia em Projeto
  - Executivo Cíclico
  - Prioridades + Teste de Escalonabilidade
- Abordagens com Melhor Esforço
  - Descarte de Tarefas
  - Perda de Deadlines
  - Redução da Precisão
  - Alteração do Período