
Sistemas de Tempo Real: Abordagens de Escalonamento na Perspectiva da Engenharia

Rômulo Silva de Oliveira
Departamento de Automação e Sistemas – DAS – UFSC
romulo.deoliveira@ufsc.br
<http://www.das.ufsc.br/~romulo>
Julho/2014

1

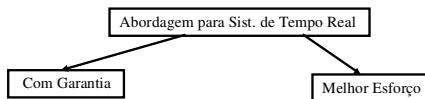
Necessidade de Diferentes Abordagens

- 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

2

Abordagens: Perspectiva Teórica

- Existem duas grandes abordagens para a questão de tempo real na perspectiva acadêmica (matemática de escalonamento)
- **Sistemas com garantia**
- **Sistemas com melhor esforço**



3

Abordagem com Garantia 1/5

- Deadlines são garantidos na construção do software
- Previsibilidade determinista
- Análise feita antes da execução
 - Carga precisa ser limitada e conhecida em projeto (Hipótese de Carga)
 - É Suposto um limite para faltas (Hipótese de Faltas)
- Para dar garantia precisa considerar o pior caso:
 - Do comportamento do software (fluxos de execução)
 - Do comportamento do hardware (tempos das instruções)

4

Abordagem com Garantia 2/5

- Necessário conhecer o comportamento do programa no pior caso
- Isto significa
 - Pior fluxo de controle para cada tarefa (if, while)
 - Piores dados de entrada
 - Pior cenário de sincronização entre tarefas (exclusão mútua, etc)
 - Pior combinação de eventos externos (interrupções, sensores, etc)
 - Pior tudo

5

Abordagem com Garantia 3/5

- Necessário conhecer o comportamento do hardware no pior caso
- Isto geralmente significa
 - Pior combinação de eventos externos (interrupções, sensores, etc)
 - Determinar os estados da memória cache
 - Determinar os estados do pipeline
 - Determinar o comportamento dos barramentos
 - Determinar o comportamento temporal seguro do hardware em relação ao pior caminho do software
 - Compor os estados em uma análise de pior caso
 - Sempre de forma segura (pessimista)
- As vezes o pior caso local não leva ao pior caso global

6

Abordagem com Garantia 4/5

- Análise dividida em duas etapas
- **Tempo de Computação C**
 - Quanto tempo esta tarefa de software levaria para executar se estivesse sozinha no computador (única tarefa, nenhuma interrupção) ?
- Para garantia é necessário o WCET (Worst-Case Execution Time)
- **Tempo de Resposta R**
 - Quanto tempo esta tarefa de software leva para executar, considerando ela própria e todas as demais atividades do sistema ?
- Para garantia é necessário o WCET de todas as tarefas do sistema, de suas taxas de recorrência, e como são suas interações

7

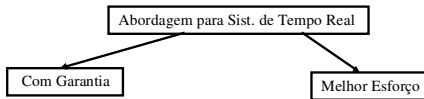
Abordagem com Garantia 5/5

- **Vantagens**
 - Determina previamente 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
 - Gera enorme sub-utilização do hardware (mais caro)
 - Difícil determinar o pior caso em soluções COTS (commercial off-the-shelf)

8

Necessidade de Diferentes Abordagens

- Existem duas grandes abordagens para a questão de tempo real na perspectiva acadêmica (matemática de escalonamento)
- **Sistemas com garantia**
- **Sistemas com melhor esforço**



9

Abordagem com Melhor Esforço 1/2

- Não existe garantia de que todos os deadlines serão cumpridos
- O “Melhor Esforço” será feito neste sentido
- Capaz de fornecer um previsão probabilista
 - Simulação, testes, etc
- 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

10

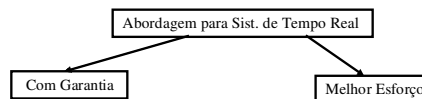
Abordagem com Melhor Esforço 2/2

- Questão fundamental: Como tratar a sobrecarga ?
 - Em sobrecarga ATRASA algumas tarefas
 - Em sobrecarga DIMINUI a precisão de algumas tarefas
 - Em sobrecarga NÃO EXECUTA algumas tarefas
 - Em sobrecarga AUMENTA O PERÍODO de algumas tarefas
- 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

11

Abordagens: Perspectiva da Engenharia 1/3

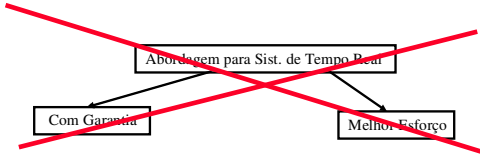
- Existem duas grandes abordagens para a questão de tempo real **EM TEORIA**
- Na prática a coisa é um pouco mais complicada
 - Entram aspectos econômicos
 - Entram aspectos do desenvolvimento como um todo e não só tempo real



12

Abordagens: Perspectiva da Engenharia 2/3

- Existem duas grandes abordagens para a questão de tempo real
EM TEORIA
- Na prática a coisa é um pouco mais complicada
 - Entram aspectos económicos
 - Entram aspectos do desenvolvimento como um todo e não só tempo real



13

Abordagens: Perspectiva da Engenharia 3/3

- Existem quatro grandes abordagens para a questão de tempo real
NA PERSPECTIVA DA ENGENHARIA
- Com garantia provada
 - **Hard real-time systems**, safety-critical systems
- Com garantia testada
 - **Pseudo-hard real-time systems**
- Melhor esforço quase garantido
 - **Firm real-time systems**
- Apenas melhor esforço
 - **Soft real-time systems**

14

Sistema de tempo real crítico verificado formalmente 1/4

- Safety-critical applications
- Não tolera nenhuma perda de deadline
- A perda de um deadline representa uma falha do sistema
- Requer algum tratamento de exceção forte
 - Tolerância a falhas via replicação ativa
 - Tolerância a falhas via propriedade construtiva (eletro-mecânica)
 - Reinicia
 - Desliga
- **Necessita verificação formal (que jamais perde um deadline)**
 - Certificação de agência fiscalizadora

15

Sistema de tempo real crítico verificado formalmente 2/4

- Safety-critical applications
 - Não tolera nenhuma perda de deadline
- Tarefas críticas em satélites
- Tarefas críticas em aviões
- Tarefas críticas em carros
- Sistemas críticos em aviões é o grande motivador da área
- Começa a ser importante para carros
- **Necessita verificação formal (que jamais perde um deadline)**
 - Às vezes certificação

16

Sistema de tempo real crítico verificado formalmente 3/4

- Análise de escalabilidade com garantia
- Ferramenta para determinar WCET
- Necessita arquitetura determinista, analisável
- Microcontrolador de pequeno ou médio porte
- Software simples, microkernel ou tudo na aplicação
 - Junta tudo, verifica o conjunto
- Certificação é o maior custo (30x mais caro que software comum)
- Tudo isto:
 - Restringe o espectro de processadores possíveis
 - Desenvolvimento é caro (ferramentas, design, verificação)
 - Justificável apenas para *safety-critical systems* com processo de certificação ou quando uma falha pode quebrar a empresa (freio do carro)

17

Sistema de tempo real crítico verificado formalmente 4/4

- Muito difícil usar multicore
 - A não ser como um conjunto de monoprocessores que estão por acaso no mesmo chip
 - Mesmo assim uma cache comum complica a análise
- Muito difícil usar processadores complexos
 - Não consegue analisar
- Novas tecnologias que poderão vir no futuro
 - Análise probabilista da escalabilidade
 - Processadores projetados especialmente para serem analisados

18

Sistema de tempo real crítico verificado por teste 1/4

- Não tolera nenhuma perda de deadline
- A perda de um deadline representa uma falha do sistema
- Requer algum tratamento de exceção forte
 - Desliga
 - Reinicia
 - Alguma tolerância a faltas passiva na construção do sistema
- **Verificação por teste (que jamais perde um deadline)**

19

Sistema de tempo real crítico verificado por teste 2/4

- Não tolera nenhuma perda de deadline
 - A perda de um deadline representa uma falha do sistema
- Mas não é *safety-critical* (pelo menos não muito)
- Sistemas de geração/transmissão de energia elétrica
 - Relés de proteção, reguladores de tensão e frequência, etc.
- Inversores elétricos
- Muitas tarefas automotivas
- Equipamentos médicos
 - Safety-critical systems, mas são lentos, é fácil cumprir deadlines
- **Verificação por teste (que jamais perde um deadline)**

20

Sistema de tempo real crítico verificado por teste 3/4

- Necessita arquitetura quase determinista
- Código simples, quase determinista
 - Não utiliza algoritmos iterativos, por exemplo
- WCET obtido através de medições
- Não tem certificação
- Ênfase em testes de stress
 - Busca as condições nas quais deadlines poderiam ser perdidos
- Microkernel determinista ou tudo na aplicação
- Folgas grandes para as tarefas críticas
- Teoria de escalonamento hard real-time pode ser usada com valores aproximados como ferramenta auxiliar do desenvolvedor

21

Sistema de tempo real crítico verificado por teste 4/4

- Existe um trade-off no projeto
- Quanto mais safety-critical for a tarefa:
 - Mais determinista é o código
 - Mais simples é o processador (pode ter mais de um)
 - Maiores são as folgas
 - Mais rigorosos são os testes
- Prioridade por importância e não por período, deadline, etc:
 - Tarefas críticas recebem prioridade fixa mais alta
 - O tempo que sobra é para as demais tarefas
 - Folga delas corresponde a todo o resto do tempo do processador que elas próprias não usam
 - Melhor a tela travar um pouco do que o motor explodir

22

Sistema de tempo real quase garantido 1/3

- O que é um sistema de tempo real quase garantido ?
- Tolerar a perda de deadlines se estas forem suficientemente raras
- O que é raro ?
- Depende da especificação do sistema:
 - Tarefa não pode perder x deadlines seguidos
 - Tarefa não pode perder mais que x deadlines em y ativações
 - Tarefa não pode perder mais que x deadlines em y segundos
 - Etc, a lista é grande
- A perda de um deadline não representa a falha do sistema
 - A perda de um deadline isolado não requer tratamento de exceção

23

Sistema de tempo real quase garantido 2/3

- A perda de um deadline não representa a falha do sistema
- A perda de um deadline isolado não requer tratamento de exceção
 - Não gera a falha do sistema
- Controle realimentado em aplicações industriais não críticas
 - A inércia da planta mascara a perda de um deadline (existem limites)
- Muitos exemplos no mundo industrial e doméstico
 - Controle de um forno industrial
 - Liga/desliga de chaves em fábrica (manufatura)
 - Linha branca
 - Controlador semafórico
 - Centrais telefônicas
- **Verificação por teste (frequência de perda de deadlines)**

24

Sistema de tempo real quase garantido 3/3

- Arquitetura qualquer, depende dos requisitos da aplicação
 - Desde de pequeno microcontrolador até PC
- Testes principalmente em condições normais
- Em geral usa microkernel, mas pode ser usado desde nada até Linux de tempo real
 - Depende das funcionalidades da aplicação
- Design e testes dependem de quanto deadline pode perder sem isto ser percebido como uma falha

25

Sistema de tempo real brando 1/2

- Soft Real-Time System
 - Tolera perda de deadlines
 - Perda de deadlines faz a percepção da utilidade do sistema cair
 - A perda de deadlines jamais requer tratamento de exceção
- Áudio e vídeo (maioria)
- Aplicações financeiras
- **Verificação por teste (distribuição estatística dos atrasos)**

26

Sistema de tempo real brando 2/2

- Arquitetura PC, ou microcontrolador sofisticado
- Desenvolvimento de software tradicional, com alguns cuidados
- Até mesmo máquinas virtuais são utilizadas
- Linux e Linux Preempt-rt são populares
- Teste em condições de carga normal
- Avaliação do desempenho em termos estatísticos
- Muitas vezes, além de ter um deadline, quanto menor o tempo de resposta, melhor

27

Teoria de Escalonamento Tempo Real 1/4

- Existe uma vasta teoria de escalonamento tempo real hard
 - Milhares de artigos
- Quase toda a teoria (matemática) de escalonamento tempo real visa sistemas de tempo real hard
 - Garantia para os deadlines provada formalmente
- A demanda de prova formal limita o espaço de projeto do software
 - Apenas técnicas com pior caso razoável
 - Não pode usar tabela hash
- A demanda de prova formal limita o espaço de projeto do hardware
 - Tools para análise de wcet suportam poucos processadores
 - Caches, barramentos são problemas
- A demanda de prova formal aumenta custos de desenvolvimento
 - Subutilização do hardware, ferramentas mais caras, mais atividades

28

Teoria de Escalonamento Tempo Real 2/4

- Os custos e as limitações da prova formal a tornam aceitável somente em sistemas safety-critical
 - Principalmente se houver certificação
- Mercado restrito:
 - Aviões
 - Satélites
 - Carros de luxo, futuristas (tende a crescer)
- Aplicação da teoria depende da evolução dos processadores
 - Processadores com tempo de execução determinista
 - Comercialmente improvável
 - Processadores com tempo de execução aleatório
 - Independente da história da execução

29

Teoria de Escalonamento Tempo Real 3/4

- No outro extremo ...
- Métodos tradicionais de engenharia de software conseguem lidar apropriadamente com sistemas de tempo real soft
 - Cuidados gerais visando desempenho
- Métodos tradicionais de engenharia de software conseguem lidar com sistemas de tempo real quase garantidos, desde que:
- Acompanhados com testes de stress
- Projetados levando em consideração os aspectos temporais
 - Impedimentos ao atendimento dos deadlines devem ser removidos do projeto
 - Por exemplo, grandes contenções causadas por mecanismos de sincronização
 - Algoritmos de escalonamento não apropriados

30

Teoria de Escalonamento Tempo Real 4/4

- Sistemas de tempo real com garantia testada demandam cuidados especiais
- Design do software e do hardware precisa levar em consideração a necessidade de determinismo
- Testes de stress são absolutamente necessários para a confiabilidade do produto
- Resultados teóricos válidos para sistemas críticos podem ser usados como heurísticas no projeto de sistemas firmes
- Não existe garantia formal para os deadlines
- Mecanismos para o tratamento de exceções temporais devem ser embutidos na aplicação
 - Tais como reiniciar ou levar para um estado seguro

31

Principais Desafios 1/5

- Quais são os principais desafios da pesquisa considerando cada uma das 4 abordagens ?
- Hard Real-Time Systems
- Pseudo-Hard Real-Time Systems
- Firm Real-Time Systems
- Soft Real-Time Systems

32

Principais Desafios do Hard Real-Time 2/5

- Estender as ferramentas de wcet para arquiteturas mais sofisticadas do que as suportadas atualmente
- Flexibilizar os modelos de tarefas usados em análises de escalonabilidade para permitir a modelagem de sistemas reais
 - Overheads, tarefas auxiliares, interações inesperadas
- Avançar a teoria no contexto de multiprocessadores
 - Considerando todas as suas peculiaridades

33

Principais Desafios do Pseudo-Hard Real-Time 3/5

- Estabelecer uma metodologia de teste de stress para aplicações de tempo real onde
 - Não existe garantia formal de que nenhum deadline será perdido
 - Existe a necessidade da crença de que nenhum deadline será perdido
- Combinar medições de partes do código com análise de escalonabilidade
 - Para que o desenvolvedor possa identificar situações raras, não observadas facilmente em testes, mas que levarão a perda de um deadline
- Criar ferramentas que permitam ao desenvolvedor da aplicação identificar as fontes de atraso no sistema (aplicação e kernel)
- Melhorar os kernels de SO para que os mesmos possam ser usados neste tipo de aplicação
 - Até mesmo Linux preempt-rt com deadlines fáceis

34

Principais Desafios do Firm Real-Time 4/5

- Determinar formas para especificar requisitos temporais que impõem limites às perdas de deadlines
 - De acordo com a semântica da aplicação
- Criar métodos para desenvolver os respectivos casos de teste a serem usadas na verificação da implementação
- Criar ferramentas que permitam ao desenvolvedor da aplicação identificar as fontes de atraso no sistema
 - Tais fontes podem estar na aplicação e/ou no kernel
- Melhorar os kernels de SO para que os mesmos não comprometam o comportamento temporal da aplicação
 - Linux preempt-rt com deadlines não tão fáceis

35

Principais Desafios do Soft Real-Time 5/5

- Adaptar o kernel dos sistemas operacionais com funcionalidade completa para que o seu comportamento temporal seja mais previsível
 - Como o que o preempt-rt faz com o Linux
- Criar mecanismos de paralelização e sincronização que permitam as aplicações aproveitarem a crescente disponibilidade de cores
 - Sem o aparecimento de pontos de severa contenção
- Ou seja, como desenvolver aplicações de tempo real soft escaláveis

36

Resumo

- Existe a necessidade de **diferentes abordagens** para o escalonamento tempo real
- Principal classificação é com respeito a **garantia dos deadlines**
- Hard Real-Time Systems
 - Verificação formal de que todos os deadlines são cumpridos
 - Mais caro, impõe severas restrições ao hardware
- Pseudo-Hard Real-Time Systems
 - Verificação por teste de que todos os deadlines são cumpridos
 - Design cuidadoso, impõe restrições ao SO
- Firm Real-Time Systems
 - Verificação por teste de que poucos deadlines são perdidos
 - Desenvolvimento quase convencional, alguns cuidados a mais
- Soft Real-Time Systems
 - Verificação por teste da distribuição dos tempos de resposta
 - Desenvolvimento convencional visando alto desempenho