Sistemas de Tempo Real:

Abordagens para o Desenvolvimento de Sistemas de Tempo Real

Rômulo Silva de Oliveira
Departamento de Automação e Sistemas – DAS – UFSC
romulo.deoliveira@ufsc.br
http://www.romulosilvadeoliveira.eng.br
Outubro/2016

Sumário

Necessidade de Diferentes Abordagens 1/2

- Mercado para tempo real é amplo
- Sistemas de tempo real variam enormemente
  - Sistema de emergência em usina petroquímica
  - Controle de temperatura do freezer
  - Videogame
- **Diferentes abordagens são necessárias**

Necessidade de Diferentes Abordagens 2/2

- Tipos de Criticalidade
  - Elevada criticalidade, exige certificação, ex: aviônica
  - Crítico, porém testes são suficientes, ex: relé de proteção elétrica
  - Não é crítico, tolera perdas eventuais de deadlines, ex: videogame
- Tipos de Sistema Operacional
  - Não usa SO, hardware simples, aplicação pequena (laço + trat. interrupções)
  - Microkernel, hardware médio sem MMU, aplicação multitarefa
  - Linux, hardware complexo, aplicação grande requer serviços de SO
- Código fonte
  - Em geral é disponível
  - Liberdade limitada para o design, código legado, bibliotecas compradas

Tempo de Resposta 1/6

- **Tarefa: σ_i** (task)
  - Segmento de código cuja execução possui atributo temporal próprio (período, deadline, etc)
  - Exemplo: método em OO, função C, trecho de um programa
- **Tempo de resposta: R_i** (response time)
  - Intervalo de tempo entre a chegada (arrival) da tarefa e sua conclusão
- **Deadline: D_i**
  - Tempo de resposta máximo desejado/aceitável, especificação
- **Objetivo: $R_i \leq D_i$**
- Maioria dos requisitos temporais aparecem como deadlines
  - Deadline pode ser igual ao período ou não
  - Às vezes existe requisito de simultaneidade

Tempo de Resposta 2/6

- Dados:
  - O código da aplicação
  - O hardware usado
  - O sistema operacional

Como saber que nenhum deadline da tarefa σ_i será perdido ?

- Vai depender dos tempos de resposta da tarefa σ_i
- Tempos de resposta variam, não são sempre iguais
- Precisamos considerar o tempo de resposta no pior caso
  Worst-Case Response Time (WCRT)

Tempo de Resposta 3/6

- Por que a tarefa não exibe sempre o mesmo tempo de resposta ?
  1) Fluxo de controle da própria tarefa varia
- Dados de entrada variam
  - Afetam o fluxo de controle
  - IF, FOR
- Estado do sistema (valor das variáveis globais) varia
  - Afeta o fluxo de controle
  - IF, FOR

Tempo de Resposta 4/6

- Por que a tarefa não exibe sempre o mesmo tempo de resposta ?
  2) Os tempos do hardware variam
- Estado dos componentes da arquitetura do computador
  - Conteúdo atual da cache, do pipeline, afetam o tempo de execução das instruções de máquina
- Barramentos compartilhados com controladores de periféricos ou outros processadores
  - Conflitos nos barramentos geram atrasos
- **Tempo de computação: C_i** (execution time)
  - Worst-Case Execution Time (WCET)
  - Captura variações do fluxo de controle e do hardware

Tempo de Resposta 5/6

- Por que a tarefa não exibe sempre o mesmo tempo de resposta ?
  3) A influência que ela sofre de outras tarefas varia
- Necessidade de sincronização (exclusão mútua)
  - Afeta o tempo de espera por recursos (tempo de bloqueio)
- Outras tarefas da aplicação com prioridade igual ou superior
  - Geram interferência
- Outras tarefas do sistema operacional, threads de kernel, tratadores de interrupção
  - Geram interferência
- Gerência do processador pelo SO consome tempo
  - Troca de contexto, algoritmo de escalonamento

Tempo de Resposta 6/6

- Mesmo conhecendo:
  - O código da aplicação
  - O hardware usado
  - O sistema operacional
- Tempos de resposta variam, não são sempre iguais
- Precisamos considerar o tempo de resposta no pior caso
  Worst-Case Response Time (WCRT)
- Que abordagens existem para verificar se nenhum deadline será perdido durante a execução futura da aplicação ?

Verificação da Escalonabilidade 1/14

- Worst-Case Response Time (WCRT) da tarefa:
  - Sua própria computação
  - Interferências
  - Bloqueios
  - Qualquer tipo de atraso que ela sofra
- Worst-Case Execution Time (WCET) da tarefa:
  - Apenas a sua própria computação
  - Tarefa executando sem interrupções
- O que desejamos conhecer é o WCRT
  - Para comparar com o deadline

Verificação da Escalonabilidade 2/14

- Como obter o WCRT da tarefa ?
- **mWCRT = medição + margem de segurança**
  - É a prática na industria em geral
  - Muito ligado aos conceitos de teste de software
- **pWCRT = medição + teoria dos valores extremos**
  - TVE em geral não é aplicável neste caso
  - Falta smoothness e monotonicidade na cauda dos histogramas
- **aWCRT = modelos analíticos**
  - WCET das tarefas + modelo completo do sistema
  - Abordagem clássica para sistemas críticos
  - Não possível com SO complexo (Linux)
  - São as equações estudadas nas disciplinas de tempo real

Verificação da Escalonabilidade 3/14

- Como obter o WCET da tarefa ?
- **mWCET = medição da tarefa sem interrupções + margem de segurança**
  - Que dados de entrada usar ?
  - Qual o estado do programa no início da execução ?
  - Como incluir falta de isolamento entre tarefas no hardware ?
- **pWCET = medição da tarefa sem interrupções + teoria de valores extremos**
  - Que dados de entrada usar ?
  - Qual o estado do programa no início da execução ?
  - Como incluir falta de isolamento entre tarefas no hardware ?
  - TVE nem sempre aplicável (falta smoothness e monotonicidade)
- **aWCET = modelos analíticos**
  - WCET dos blocos básicos + grafo do fluxo de controle
  - Análise de valor para variáveis ou anotações do programador
  - Precisa incluir efeito das preempções de alguma forma
  - Clássicos para sistemas críticos

Verificação da Escalonabilidade 4/14

- Como obter o WCET do bloco básico ?
  - Bloco Básico é uma sequência de instruções, a qual pode ser alcançada somente pela primeira instrução e a única saída é pela última instrução
- **mWCET = medição do bloco básico sem interrupções + margem de segurança**
  - Existem soluções no mercado: Rapita
- **pWCET = medição do bloco básico sem interrupções + teoria de valores extremos**
  - Como incluir falta de isolamento entre tarefas no hardware ?
  - TVE em geral aplicável
- **aWCET = modelos analíticos do hardware**
  - WCET das instruções de máquina que compõe o bloco básico

Verificação da Escalonabilidade 5/14

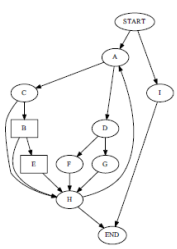
- **O tempo de execução de um bloco básico é contante ou variável ?**
- 1) Processador simples + computador simples
  - Tempo de execução é constante
- 2) Processador simples + computador com elementos ativos (ex:DMA)
  - Tempo de execução varia
  - Variação independe da história da tarefa
- 3) Processador complexo (ex:cache) + computador simples
  - Tempo de execução varia
  - Variação depende da história da tarefa
- 4) Processador complexo (ex:cache) + computador com elementos ativos (ex:DMA)
  - Tempo de execução varia
  - Variação parcialmente depende da história da tarefa
  - Variação parcialmente independe da história da tarefa

Verificação da Escalonabilidade 6/14

- **Quando a variação do tempo de execução depende da história da tarefa**
  - Ela depende do caminho da execução até o bloco básico
- **Quantos caminhos existem ?**
- 1) Sem desvio nem laço
  - Apenas 1 caminho
  - Então o tempo de execução é constante
- 2) Com desvios mas sem laços
  - Um certo número de caminhos, depende da combinação dos desvios
  - Porém ainda um numero tratável explicitamente (ex: menor que 1 milhão)
- 3) Sem desvios mas com laços
  - Um certo número de caminhos, depende do número de iterações
  - Porém ainda um numero tratável explicitamente (ex: menor que 1 milhão)
- 4) Com desvios e com laços
  - Um imenso número de caminhos (ex: laço de 100 com 4 ramos, 4^100)
  - Número intratável explicitamente (ex: 10^60)

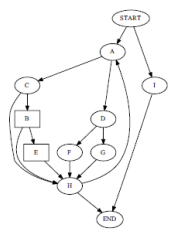
Verificação da Escalonabilidade 7/14

```
if( START )
  I;
else {
  do {
    if( A ) {
      if( C )
        if( B )
          E;
      }
    else {
      if( D )
        F;
      else
        G;
      }
    }while( H );
  }
END;
```



Verificação da Escalonabilidade 8/14

- Quantos caminhos diferentes existem entre START e END ?
  - Ramo da direita faz I
  - Ramo da esquerda tem um laço contendo 5 ramos:
    - A D F H
    - A D G H
    - A C H
    - A C B H
    - A C B E H
  - Precisa um limite para o número de iterações do laço



Verificação da Escalonabilidade 9/14

- Precisa um limite superior para o número de iterações do laço
  - Vamos supor 6, repete o laço de 1 a 6 vezes

  - Executa o laço 1 vez: 5^1 possibilidades
  - Executa o laço 2 vezes: 5^2 possibilidades
  - Executa o laço 3 vezes: 5^3 possibilidades
  - Executa o laço 4 vezes: 5^4 possibilidades
  - Executa o laço 5 vezes: 5^5 possibilidades
  - Executa o laço 6 vezes: 5^6 possibilidades

Verificação da Escalonabilidade 10/14

- Número de caminhos do laço: $\Sigma_{i=1}^{M} (N^i)$
  - Limite do laço são M vezes
  - Número de ramos no corpo do laço é N

- Dominado por N^M

- Por exemplo, N=4 e M=100
  - Resulta em 4^{100} o que é aproximadamente 10^{60} !!!
  - Completamente intratável processar cada caminho explicitamente
  - Análise estática usa Programação Inteira para resolver o problema, através de podas de caminhos que são menores do que algum caminho já conhecido

Verificação da Escalonabilidade 11/14

- O que define qual caminho é executado ?
  - Variáveis de entrada do programa
  - Variáveis permanentes alteradas em execuções anteriores
  - Certos estados de variáveis permanentes somente são atingidos após muitas execuções da tarefa

- Estes são os caminhos sintaticamente possíveis

- Alguns desses caminhos são semanticamente impossíveis
- Impossível pela semântica do programa
  - Análise de valor pode identificar (parcialmente)
- Impossível pela semântica do ambiente
  - Entradas impossíveis de acontecer na prática

Verificação da Escalonabilidade 12/14

- Como obter o WCET do bloco básico ?
  - Bloco Básico é uma sequência de instruções, a qual pode ser alcançada somente pela primeira instrução e a única saída é pela última instrução

- **mWCET = medição do bloco básico sem interrupções + margem de segurança**
  - Existem soluções no mercado: Rapita

- **pWCET = medição do bloco básico sem interrupções + teoria de valores extremos**
  - Como incluir falta de isolamento entre tarefas no hardware ?
  - TVE em geral aplicável

- **aWCET = modelos analíticos do hardware**
  - WCET das instruções de máquina que compõe o bloco básico

Verificação da Escalonabilidade 13/14

- Como obter o WCET das instruções de máquina ?
  - Depende completamente da arquitetura/organização do computador em questão

- **Processador muito simples:**
  - Manual do processador informa duração de cada instrução de máquina

- **Processador com maior desempenho:**
  - Inclui componentes cujo tempo de execução varia
- São incluídos no processador:
  - Cache de instruções
  - Cache de dados
  - Pipeline
  - Pipeline superescalar
  - Barramentos diversos
  - Memória dinâmica
  - Branch predictor
  - Etc, etc, etc

Verificação da Escalonabilidade 14/14

- Como obter o WCET das instruções de máquina ?
  - Depende completamente da arquitetura/organização do computador em questão

- É possível obter um limite superior (upper-bound) analítico quando os componentes do hardware, embora não deterministas, são:
  - Livres de anomalias temporais
  - Composables
- Limite superior para o WCET do bloco básico requer a análise de todas as possibilidades
  - Ou aproximações pessimistas
- Existem produtos no mercado para monoprocessadores de capacidade média
  - aiT suporta ARM Cortex-M3 (roda FreeRTOS)

Abordagem Clássica para Sistemas Críticos

- Obtem upper-bound determinista para elementos do hardware
  - Limita a complexidade do hardware
  - Desejo da industria seria usar hardware mais complexo que este
  - Multicore usado em PC é impossível
- Obtem aWCET do BB
- Obtem aWCET da tarefa
  - Programação inteira para achar o pior caminho no grafo de fluxo de controle
- Obtem aWCRT da tarefa
  - Com modelo completo do sistema operacional e demais tarefas
  - Possível apenas para SO simples
  - Linux é impossível

Abordagem Probabilista para Sistemas Críticos

- Nova proposta probabilista usa Teoria de Valores Extremos (TVE)
  - Faz uma amostragem dos tempos de execução
  - Projeta probabilisticamente estes tempos para obter um pWCET
  - Existe uma probabilidade de 10^{-12} do WCET real ser maior que pWCET
- Obtem pWCET do BB com medição e teoria de valores extremos
  - Comportamento do hardware deve ser bom para a TVE
- Obtem aWCET da tarefa
  - Programação inteira para achar o pior caminho no grafo de fluxo de controle
- Ou obtem pWCET da tarefa diretamente com medição e TVE
  - Comportamento do hardware deve ser bom para a TVE
- Obtem aWCRT da tarefa
  - Com modelo completo do sistema operacional e demais tarefas
  - Possível apenas para SO simples
  - Linux é impossível

Abordagem Comum para Sistemas Críticos

- Obtem mWCRT diretamente
  - Emprega medição + margem de segurança
  - Na medição busca levar o sistema aos limites (teste de stress)
  - Hardware com complexidade limitada (microcontroladores)
  - Sistema operacional simples ou não tem
- Esta é a prática geral da indústria mesmo para sistemas críticos
  - Não serve para aviões
  - Não serve para futuros automóveis autônomos
- Questões principais:
  - Como gerar os casos de testes ?
  - Quando parar de medir ?
  - Como estimar a confiabilidade deste mWCRT ?

Dilema Fundamental dos Sistemas Críticos

- Processadores não são projetados para limitar o tempo de resposta no pior caso
  - São feitos para melhorar o tempo de resposta no caso médio
  - Pior caso é ignorado por ter probabilidade muito pequena
- Querer garantir deadline neste cenário é "forçar a barra"
  - Possível somente com processador bem simples
- E se existisse um processador para tempo real crítico ?
  - Até hoje mercado muito pequeno para justificar investimento
  - Carros autônomos poderão quem sabe gerar a demanda que vai justificar este investimento
  - Vai tirar o emprego dos professores de tempo real

Análise de Escalonabilidade das Tarefas 1/5

- Em geral significa obter o WCRT a partir dos WCETs
  - **Teste de Utilização**
  - **Análise do Tempo de Resposta**
  - **Executivo Cíclico**
- Prova a escalonabilidade a partir do WCET das tarefas e um modelo do sistema
- Como obter o modelo do sistema em termos de tarefas ?
  - Desenvolve o software a partir do modelo
  - Cria o modelo manualmente a posteriori
  - Extrai o modelo automaticamente do código
- Precisa dos tempos máximos de execução das tarefas (WCET)
  - Medição ou modelos analíticos para o hardware
- Análise é possível apenas quando o design do software é elegante e segue as premissas do método em questão

Análise de Escalonabilidade das Tarefas 2/5

- Métodos de **Verificação Formal** também poderiam ser usados a princípio
- Obtém um modelo apropriado para verificação formal
  - Timed Petri Net, Timed Automata, etc
- Prova formalmente que o deadline nunca é perdido
  - Pelo menos nunca no modelo
- Como obter o modelo ?
  - Desenvolve a partir do modelo
  - Cria o modelo manualmente a posteriori
  - Extrai o modelo automaticamente do código
- Precisa dos tempos de execução no pior caso (WCET)
- Problemas:
  - Confiabilidade do modelo
  - Mesmo modelos pequenos podem levar a uma explosão de estados

Análise de Escalonabilidade das Tarefas 3/5

- **Simulador Adversário (Simulação Dirigida)**
- Obtém um modelo apropriado para uma simulação controlada
- Usa um simulador adversário para obter o maior tempo de resposta possível
  - Lower Bound (limite inferior) para o WCRT
  - Meta-heurísticas/heurísticas para guiar a simulação
- Como obter o modelo ?
  - Desenvolve a partir do modelo
  - Cria o modelo manualmente a posteriori
  - Extrai o modelo automaticamente do código
- Precisa dos tempos de execução no pior caso (WCET)
- Problema:
  - É apenas um valor aproximado, menor do que o verdadeiro WCRT

Análise de Escalonabilidade das Tarefas 4/5

- Prática mais comum é o **Teste Convencional**
- Mede os tempos de resposta
  - Submete o sistema a testes de stress
  - Mede os tempos de resposta
  - Assume que viu tudo
  - Se nunca perder deadline nos testes, assume que nunca vai perder
- Teoria dos valores extremos pouco aplicável neste caso em função da distribuição dos tempos de resposta (histogramas)
- Problema:
  - É apenas um valor aproximado, possivelmente menor do que o WCRT
  - Como gerar os casos de testes ?
  - Quando parar de medir ?
  - Como estimar a confiabilidade deste mWCRT ?

Análise de Escalonabilidade das Tarefas 5/5

- Com garantia provada (Hard Real-Time System)
  - Não pode perder deadline nunca, necessário certificação
  - Teste de Utilização, Análise do Tempo de Resposta, Executivo Cíclico
  - Exemplo: sistemas aviônicos
- Com garantia testada (Quasi-Hard Real-Time System)
  - Não pode perder deadline nunca, não precisa certificação
  - Testes mostrando que não perde deadline
  - Exemplo: Inversor elétrico, relé de proteção elétrica
- Sem garantia (Soft Real-Time System, Best-effort)
  - Precisa estipular serviço mínimo (percentual perdido, atraso máximo, etc)
  - Testes mostrando que é bom o suficiente
  - Exemplo: Central telefônica, controle realimentado de processo lento
- Não é tempo real, requisito temporal é trivial
  - Folha de pagamentos

Sumário

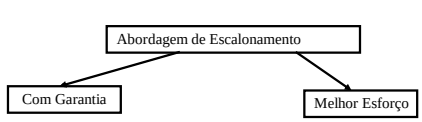
- Verificação da Escalonabilidade
  - Descrição do problema
  - Abordagens para verificar a escalonabilidade
  - Análise de escalonabilidade das tarefas
- **Escalonamento de tarefas**
- Sistemas de tempo real na perspectiva da engenharia
- Conclusão

Escalonamento de Tarefas 1/2

- Independentemente da abordagem para verificar escalonabilidade precisa de alguma abordagem de escalonamento
- São coisas diferentes
- Por exemplo:
  - Pode usar a mesma abordagem de escalonamento com verificação formal ou com testes simples

Escalonamento de Tarefas 2/2

- Existem duas grandes abordagens de escalonamento para sistemas de tempo real na perspectiva acadêmica
  - Matemática de escalonamento
- **Sistemas com garantia**
- **Sistemas com melhor esforço**



Escalonamento de Tarefas com Garantia 1/9

- 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)

Escalonamento de Tarefas com Garantia 2/9

- 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

Escalonamento de Tarefas com Garantia 3/9

- 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

Escalonamento de Tarefas com Garantia 4/9

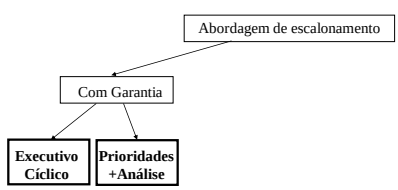
- Análise usualmente dividida em duas etapas
- **Tempo de Computação C**
  - Quanto tempo esta tarefa de software levaria para executar se estivesse sozinha no computador (única tarefa, nenhuma interrupção) ?
- Para garantia é necessário o WCET (Worst-Case Execution Time)
- **Tempo de Resposta R**
  - 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, e de suas taxas de recorrência, e como são suas interações
  - Para obter o WCRT (Worst-Case Response Time)

Escalonamento de Tarefas com Garantia 5/9

- **Vantagens**
  - Determina previamente que todos os deadlines serão cumpridos
  - Necessário para aplicações críticas certificadas
  - 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)

Escalonamento de Tarefas com Garantia 6/9

- Existem duas formas de escalonamento para obter garantia
  - Executivo cíclico
  - Prioridades + Análise de escalonabilidade



Escalonamento de Tarefas com Garantia 7/9

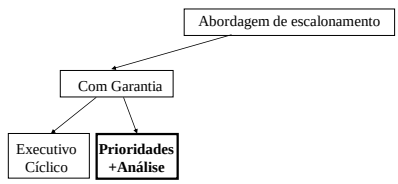
- Todo o trabalho de escalonamento é feito em projeto
- Resultado é uma **grade de execução** (*time grid*)
- Grade determina qual tarefa executa quando
- Garantia obtida através de uma simples inspeção da escala

- 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, funciona melhor com tudo periódico, apenas interrupções de timer

Escalonamento de Tarefas com Garantia 8/9

- Existem duas formas de escalonamento para obter garantia
  - Executivo cíclico
  - Prioridades + Análise de escalonabilidade



Escalonamento de Tarefas com Garantia 9/9

- Cada tarefa recebe uma prioridade
- Escalonamento em geral é preemptivo
- Análise matemática antes da execução determina escalonabilidade
  - Complexidade da análise depende do modelo de tarefas

- Na execução:
  - Escalonador dispara as tarefas conforme as prioridades

- Vantagens: Suporta tarefas esporádicas com facilidade
- Desvantagens: Existem análises apenas para sistemas simples

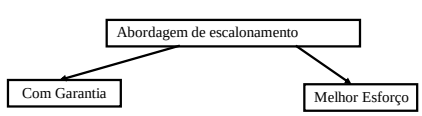
- Usado em aplicações que exigem garantia mas também requerem flexibilidade para acomodar tarefas esporádicas, sistemas maiores

Escalonamento de Tarefas

- Existem duas grandes abordagens de escalonamento para sistemas de tempo real na perspectiva acadêmica
  - Matemática de escalonamento

- **Sistemas com garantia**

- **Sistemas com melhor esforço**



Escalonamento de Tarefas 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

Escalonamento de Tarefas 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

Sumário

- Verificação da Escalonabilidade
  - Descrição do problema
  - Abordagens para verificar a escalonabilidade
  - Análise de escalonabilidade das tarefas
- Escalonamento de tarefas
- **Sistemas de tempo real na perspectiva da engenharia**
- Conclusão

Sistemas de Tempo Real: Perspectiva da Engenharia

- Literatura oferece várias maneiras para
  - Verificar a escalonabilidade
  - Escalonar os sistemas
- Na prática a coisa é um pouco mais complicada
  - Entram aspectos econômicos
  - Entram aspectos do desenvolvimento como um todo e não só tempo real
  - Entram as limitações das propostas acadêmicas

Sistema de Tempo Real com Garantia Provada 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 faltas via replicação ativa
  - Tolerância a faltas passiva via propriedade construtiva (eletro-mecânica)
  - Reinicia
  - Desliga

Sistema de Tempo Real com Garantia Provada 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 prova matemática de que jamais perde um deadline**
  - Certificação de agência fiscalizadora

Sistema de Tempo Real com Garantia Provada 3/4

- Análise de escalonabilidade 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)

Sistema de Tempo Real com Garantia Provada 4/4

- Muito difícil usar multicore
  - A não ser como um conjunto de monoprocessadores 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 escalonabilidade
  - Processadores projetados especialmente para serem analisados

Sistema de Tempo Real com Garantia Testada 1/4

- Não tolera nenhuma perda de deadline
- A perda de um deadline representa uma falha do sistema
- Requer algum tratamento de exceção forte
  - Tolerância a faltas via replicação ativa
  - Tolerância a faltas passiva via propriedade construtiva (eletro-mecânica)
  - Reinicia
  - Desliga
- **Verificação por teste (que jamais perde um deadline)**

Sistema de Tempo Real com Garantia Testada 2/4

- Não tolera nenhuma perda de deadline
  - A perda de um deadline representa uma falha do sistema
- Mas não é geralmente *safety-critical*
  as vezes até é !!!
- 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)**

Sistema de Tempo Real com Garantia Testada 3/4

- Necessita arquitetura quase determinista
- Código simples, quase determinista
  - Não utiliza algoritmos recursivos, 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

Sistema de Tempo Real com Garantia Testada 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

Sistema de Tempo Real sem Garantia 1/3

- Chamado soft real-time system, best-effort
- Tolera 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

Sistema de Tempo Real sem Garantia 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 (perde deadlines de forma aceitável)**

Sistema de Tempo Real sem Garantia 3/3

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

Sumário

- Verificação da Escalonabilidade
  - Descrição do problema
  - Abordagens para verificar a escalonabilidade
  - Análise de escalonabilidade das tarefas
- Escalonamento de tarefas
- Sistemas de tempo real na perspectiva da engenharia
- **Conclusão**

Conclusão 1/4

- Existe uma vasta teoria de escalonamento **tempo real hard**
  - Muitos milhares de artigos
- Quase toda a teoria (matemática) de escalonamento tempo real visa sistemas de tempo real hard
  - Garantia para os deadlines provada matematicamente
- 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

Conclusão 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
    - Comercialmente improvável

Conclusão 3/4

- No outro extremo ...
- Métodos tradicionais de engenharia de software conseguem lidar com **sistemas de tempo real soft**, 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 mecanimos de sincronização
  - Algoritmos de escalonamento não apropriados

Conclusão 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