Sistemas de Tempo Real

Tempo

Várias definições

Tempo em sistemas (Interpretações)

- Tempo Lógico: definido a partir de relações de precedência entre eventos. Estabelece ordens causais entre conjunto de eventos.
- Tempo Físico: tempo métrico que expressa quantitativamente a distância entre eventos. Estabelece também as ordens totais entre eventos
- Tempo Contínuo: segue a natureza uniforme e contínua do tempo físico e é isomorfo a $\Re$
- Tempo Discreto: simplificação do tempo contínuo e isomorfo a $\aleph$

Farine e et al. Sistemas de Tempo Real. 2000

Tempo

- Tempo Global: Referência temporal única para os componentes do sistema
- Tempo Local: Cada componente do sistema possui sua própria referência temporal

Farine e et al. Sistemas de Tempo Real. 2000

Sistemas de Tempo Real

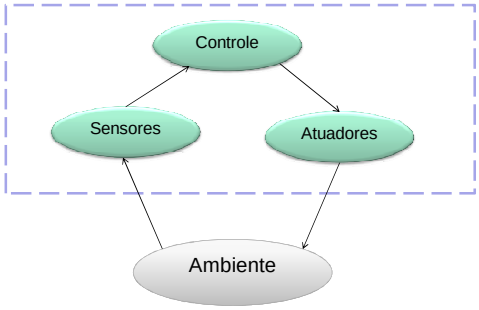
Um sistema que, além de demandar resultados corretos , estes resultados precisam estar disponíveis nos tempos previamente especificados

Sem violação de prazos (*deadline)*

Necessidade de infraestrutura de software especializada (Ex: Sistemas Operacionais de Tempo Real - RTOS)

Sistemas de Tempo Real

Sistemas de Tempo Real



Características

- **Responder a estímulos externos nos tempos especificados**
  - Diferente de computação rápida
- **Previsibilidade**
  - Diferentes execuções do sistema precisam ser similares
- **Dependabilidade**
  - Confiança em um sistema computacional, de tal forma que pode ser assegurada justificadamente a continuidade do serviço que ele disponibiliza
  - Uso de métricas como confiabilidade e disponibilidade
- **Concorrência**
- **Precisão**
  - Resultados precisos
  - Pode ocorrer balanceamento entre imprecisão e precisão caso as restrições temporais não possam ser atendidas

Sistemas de Tempo Real

Alguns sistemas que podem ter restrições temporais:

- Sistemas Embarcados
- Redes de Computadores
- Sistemas Distribuídos
- Sistemas de Banco de Dados
- ...

Classificação

- Sistemas de Tempo Real Não Críticos
  - Violação de prazo = Perdas toleráveis
- Sistemas de Tempo Real Críticos
  - Violação de prazo = Perdas do equipamento ou vidas humanas

Sistemas de Tempo Real Não Crítico

Restrições temporais são flexíveis

Perda de *dead lines* = perda da qualidade do serviço

Usualmente, associação de *soft deadline*

- A não violação do *deadline* permite provê um serviço da melhor

Adoção de distribuições de probabilidade para representação do comportamento (ex: tempo de computação)

Violação de Prazo

Tempo Real Crítico

Tempo Real Não Crítico

Sistemas de Tempo Real

Várias áreas estão envolvidas. Ex: Estatística e Probabilidade, Métodos Formais

Pesquisa

- **Escalonamento de tarefas e recursos**
- Componentização
- Virtualização
- Dependabilidade
- Comunicação e Distribuição

Sistemas de Tempo Real

Várias áreas estão envolvidas. Ex: Estatística e Probabilidade, Métodos Formais

Pesquisa

- **Escalonamento de tarefas e recursos**
- Componentização
- Virtualização
- Dependabilidade
- Comunicação e Distribuição

Foco será nos sistemas críticos

Escalonamento

Abstração computacional via o uso de tarefas

- Unidade concorrente
- Conceito similar a processos

Previsibilidade: as restrições das tarefas precisam ser conhecidas em tempo de projeto

Tipos comumente adotados

- Tarefa Periódicas: Ativação ocorre em intervalos regulares de tempo
- Tarefas Esporádica: Ativação devido à ocorrência de eventos internos/externos

Restrições temporais

Tarefa Periódica: tp = (r,c,d,p)

- r=Tempo de Liberação (Release) – Tempo mais cedo que a tarefa estará pronta pra executar no período
- c=Tempo de computação no pior caso (worst-case computation time - **WCET**)
- d=Tempo de término (deadline) – Intervalo de tempo entre o início do período e o tempo máximo que a tarefa precisará concluir sua execução
- p=Período – Intervalo de tempo para criação de uma nova instância da tarefa

Tarefa esporádica: ts = (c,d,min)

- min=tempo mínimo entre duas requisições consecutivas

Relações entre tarefas

Exclusão Mútua – Somente uma tarefa pode utilizar o recurso compartilhado em um determinado momento

Precedência

- Tarefa i *precede* Tarefa j
- Tarefa j somente poderá executar após a conclusão da Tarefa i

Sistema de Navegação Inercial

Tarefa	Período	Deadline	WCET
Altitude Updater	25	25	4.5
Velocity Updater	400	400	40
Altitude Sender	625	625	100
Navigation Sender	10000	10000	200
Status Display	10000	10000	1000
Runtime BIT	10000	10000	50
Position Updater	12500	12500	250
Unidade de Tempo = 100 microssegundos			

Técnicas de Escalonamento

Dinâmica

- Durante a execução
- Adoção de uma política de escalonamento (usualmente baseada em prioridades)
- Antes da execução, é realizado um teste sobre a possibilidade de escalonamento

Estática

- Antes da execução
- Representada por um algoritmo de busca
- Resultado: uma tabela indicando a ordem de execução de cada instância de tarefa

Considerando relações entre tarefas, o problema é NP-Completo

Abordagens Dinâmicas

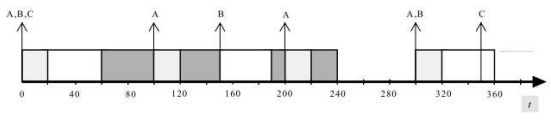
Abordagens representativas

- **Taxa Monotônica (Rate Monotonic)**
  - Prioridade estática
  - Menor período, maior a prioridade
  - Tarefas independentes
  - Deadline = Período
- **Earliest Deadline First**
  - Prioridade dinâmica
  - Maior prioridade, prazo de conclusão mais próximo
  - Tarefas Independentes
  - Deadline = Período
- **Protocolo de Prioridade Teto**
  - Priority Ceiling Protocol
  - Mesmas suposições de rate monotonic
  - Permite exclusão mútua
  - Utilização de semáforos

Taxa Monotônica (RM)

$$U = \sum_{i}^{n} C_i / P_i \leq n\left(2^{1/n} - 1\right) \longrightarrow$$ Condição suficiente

Tarefas Periódicas	*Período* (P_i)	*Tempo de Computação* (C_i)	*Prioridade RM* (p_i)
tarefa A	100	20	1
tarefa B	150	40	2
tarefa C	350	100	3

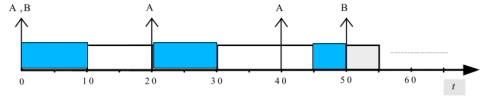


Earliest Deadline First (EDF)

$$U = \sum_{i=1}^{n} C_i / P_i \leq 1 \longrightarrow$$ Condição necessária e Suficiente para EDF

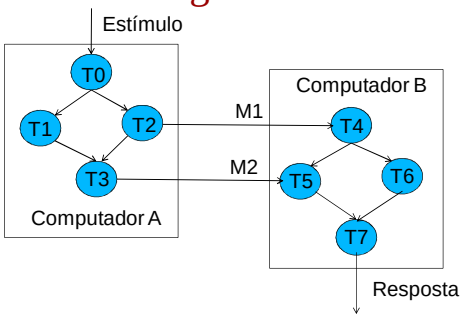
Condição necessária pra qualquer técnica (inclusive estática)

tarefas periódicas	C_i	P_i	D_i
tarefa A	10	20	20
tarefa B	25	50	50

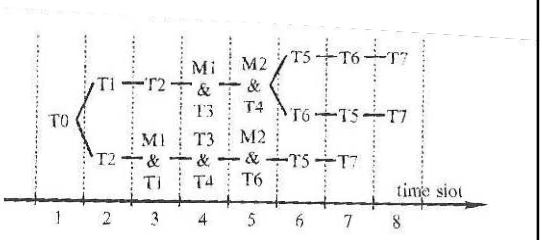


(a) Escalonamento EDF

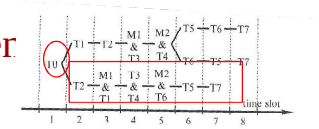
Abordagens Estáticas



Abordagens Estáticas



Abordager



```
void codeT0() {...} void codeT1() {...}
void codeT2() {...} void codeT3() {...}
void sendM1(){...} void sendM2(){...}
struct SchItem sch[7] = {
    {1,START, 0, {(void *) codeT0}},
    {2,START, 2, {(void *) codeT2}},
    {3,START, 1, {(void *) codeT1,  (void *) sendM1}},
    {4,START, 3, {(void *) codeT3}},
    {5,START, 0, {(void *) sendM2}}
}
```

```
void codeT4() {...} void codeT5() {...}
 void codeT6(){...} void codeT7(){...}
struct SchItem sch[4] = {
    {4,START, 0, {(void *) codeT4}},
    {5,START, 6, {(void *) codeT6}},
    {6,START, 6, (void *) codeT5},
    {7,START, 7,(void *) codeT7},

    }
```

Abordagens Estáticas

Construção de uma escala antes da execução (*offline, pre-runtime)*

Uma busca por solução (ex: busca em largura, profundidade,...)

- Utilização de heurísticas
- Complexidade em encontrar solução ótima
- Uma escala viável é satisfatória

Considera um período de escalonamento igual ao mínimo múltiplo comum (MMC) entre os períodos das tarefas

- Várias instâncias de uma tarefa podem ocorrer no MMC
- Release da i-ésima instância $r_{p_i} = r_p + prd_p \times (i-1)$ Deadline da i-ésima instância $d_{p_i} = d_p + prd_p \times (i-1)$

Abordagens Estáticas vs Dinâmicas

Estáticas

- Menos *overhead* em tempo de execução
- Se uma escala existe, então o algoritmo de busca pode encontrá-la
- Menos flexíveis
- Comportamento determinístico
- Grande capacidade em lidar com relações entre tarefas complexas

Dinâmicas

- Teste de escalonamento usualmente simples
- Capacidade de adaptação (Flexibilidade)
- Suposições que podem não ser a realidade de alguns sistemas
- Podem não conseguir gerar uma escala viável em situações que abordagens estáticas conseguem
- *Overhead* maior que as abordagens estáticas

Abordagens Estáticas vs Dinâmicas

Estáticas

- Menos *overhead* em tempo de execução
- Se uma escala existe, então o algoritmo de busca pode encontrá-la
- Menos flexíveis
- Comportamento determinístico
- Grande capaci[illegible]s complexas

Solução: abordagem híbrida

Dinâmicas

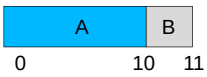
- Teste de escalonamento usualmente simples
- Capacidade de adaptação (Flexibilidade)
- Suposições que podem não ser a realidade de alguns sistemas
- Podem não conseguir gerar uma escala viável em situações que abordagens estáticas conseguem
- *Overhead* maior que as abordagens estáticas

Abordagens Estáticas vs Dinâmicas

Ta = (r=0,c=10,d=12)
Tb = (r=1,c=1,d=2)
Exclusão mútua entre A e B

Abordagens Estáticas vs Dinâmicas

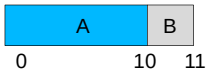
Ta = (r=0,c=10,d=12)
Tb = (r=1,c=1,d=2)
Exclusão mútua entre A e B



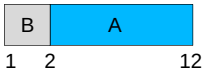
B tem o deadline violado (dinâmico)

Abordagens Estáticas vs Dinâmicas

Ta = (r=0,c=10,d=12)
Tb = (r=1,c=1,d=2)
Exclusão mútua entre A e B



B tem o deadline violado (dinâmico)



Escala viável (estático)

Estimativa do WCET

Métodos para estimar o tempo de computação no pior caso

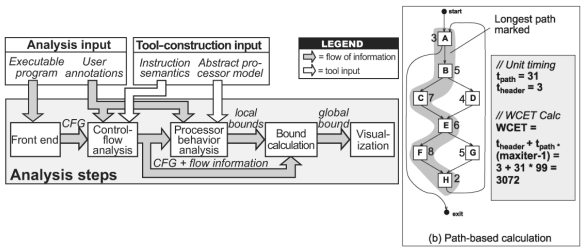
Métodos estáticos

- Analisa os possíveis de fluxos de execuções de um código
- Adota um modelo abstrato de hardware
- Faz estimativas de limites superiores

Metodos baseados em medição

- Usualmente, executa a tarefa ou partes da tarefa em um hardware real
- Considera um conjunto de dados de entrada que geram o pior caso na execução da tarefa
- Faz estimativas baseado no comportamento observado

Métodos Estáticos



Estimativa do WCET

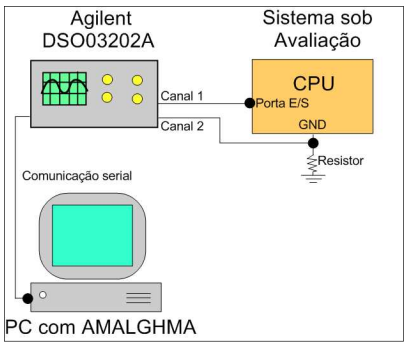
```
for(i = 0; i<= n; i++) {
            ...
}
```

a)

```
if((guard expression) | TRUE) {
        //takes 100ms
            ...
} else {
        //takes 20ms
            ...
}
```

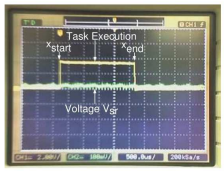
b)

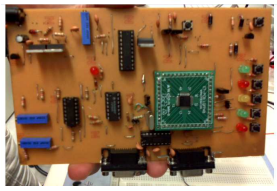
Métodos baseados em Medição



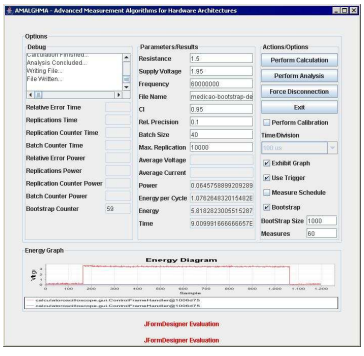
Métodos baseados em Medição

```
while(TRUE) {

      IOPort = IOPort | 0x1;
      codeT1();
      IOPort = IOPort & ~0x1;

      for(i = 0;i < 300;i++); //delay

}
```





Métodos baseados em Medição



Métodos Estáticos x Medição

Ambos métodos são complementares

Métodos estáticos

- Usualmente, fazem uma estimativa do pior caso elevada
- Dificuldades em criar um modelo abstrato do hardware
- Alguns desafios na validação

Métodos baseados em medição

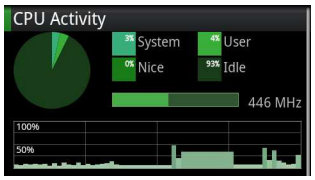
- Usualmente, subestimam o pior caso
  - Ex: Técnicas para melhorar execução de instruções - pipeline, cache, execução de instruções fora de ordem
- Ajudam a refinar o WCET e validar os métodos estáticos

Dynamic Voltage Scaling

Permite reduzir a voltagem de alimentação de alguns componentes (e.g., CPU)

A velocidade também é diminuída bem como o consumo de energia

Sistemas de tempo-real: restrições temporais e de energia usualmente conflitantes



Dynamic Voltage Scaling

Assuma uma CPU com as seguintes voltagens/frequências

1.21V/20MHz 1.39V/30MHz 1.76/50MHz

Escalonar as seguintes tarefas. C é o pior caso em termos de ciclos de execução:

T1 =(r=0,c=150 x 10^6,d=6,p=13)

T2=(r=2,c=50 x 10^6,d=3,p=13)

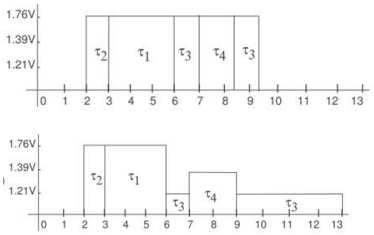
T3=(r=2,c=100 x 10^6,d=13,p=13)

T4=(r=7,c=6 0 x 10^6,d=9,p=13)

T1 precede T3 e T2 precede T4

Exclusão mútua entre T1 e T2

Dynamic Voltage Scaling



Dynamic Voltage Scaling

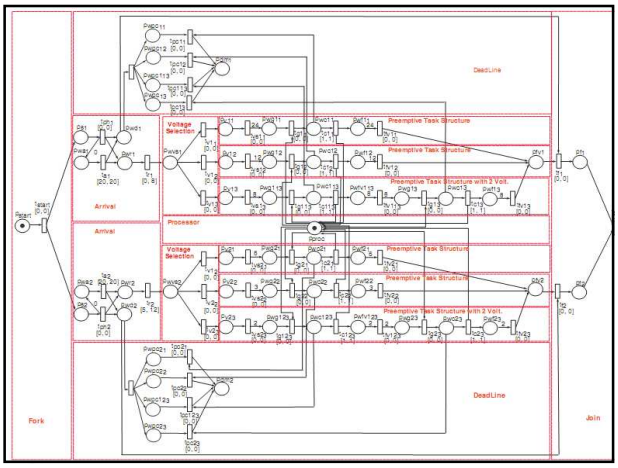
Exemplo de modelagem

CPU: 1V/10MHz e 2V/20MHz

Escalonar as seguintes tarefas. C é o pior caso em termos de ciclos de execução:

T1 =(r=0,c=240 x 10^6,d=20,p=20)

T2=(r=5,c=60 x 10^6,d=15,p=20)



Dynamic Voltage Scaling

```
void codeT1() {...}
void codeT2() {...}
#define SCHEDULE_SIZE 5
struct SchItem sch[SCHEDULE_SIZE] =
{
  {0, INSTANCE, 1, 2V/20MHz,(int *)codeT1},
  {5, INSTANCE, 2, 2V/20MHz,(int *)codeT2},
  {7, VOLT_SWITCH, 2, 1V/10MHz,(int *)codeT2},
  {9, RETURN, 1, 2V/20MHz,(int *)codeT1},
  {12, VOLT_SWITCH, 1, 1V/10MHz, (int *)codeT1},
};
```

Centro de Informática

Modelos Temporizados

Diversos modelos propostos. Alguns representativos (Probabilísticos e Determinístcos):

- Lógicas Temporais (Ex: Linear Time Temporal Logic)
- Autômatos temporizados
- Álgebra de processos temporizadas (ex: Timed CSP)
- Redes de Fila
- Cadeias de Markov
- Redes de Petri Temporizada (Ex: TPN)

Importância dos tempos físicos em sistemas críticos

Foco será nos modelos determinísticos

Modelos Temporizados

Diversos modelos propostos. Alguns representativos (Probabilísticos e Determinístcos):

- Lógicas Temporais (Ex: Linear Time Temporal Logic)
- Autômatos temporizados
- Álgebra de processos temporizadas (ex: Timed CSP)
- Redes de Fila
- Cadeias de Markov
- Redes de Petri Temporizada (Ex: TPN)

Importância dos tempos físicos em sistemas críticos

Foco será nos modelos determinísticos

Avaliação de Sistemas

Modelagem

- Modelos Analíticos
  - Determinísticos
    - Avaliação de pior (melhor) caso
  - Probablísticos
    - Valores médios prováveis
- Simulação
  - Análise Exaustiva

Implementação real

- Medidas obtidas do sistema real
- *Benchmark*
- Protótipos