Sistemas de Tempo Real

Msc.Marcelo de Paiva Guimarães

Doutorando da Universidade de São Paulo

Laboratório de Sistemas Integráveis

Escola Politécnica da Universidade de São Paulo

{paiva@lsi.usp.br}

Fonte: http://www.lcmi.ufsc.br/gtr/livro/principal.htm

Sistemas de Tempo Real

- Sistemas computacionais de tempo real
  - Submetidos a requisitos de natureza temporal
  - Resultados devem estar corretos (lógica e temporalmente)
  - " **Fazer o trabalho usando o tempo disponível**"
- Sistemas em geral

  "**Fazer o trabalho usando o tempo necessário**"

Exemplos de STR

- Telecomunicações
  - Estabelecimento de conexões, videoconferência
- Aeroespacial
  - Automação em aeronaves, sondas espaciais
- Defesa
  - Radar
- Entretenimento
  - Vídeo games, vídeo sob demanda

Concepções Erradas

- Tempo real significa execução rápida
- Computadores mais rápidos vão resolver todos os problemas
- STR são pequenos, escritos em *assembly*
- Não existem problemas específicos da área de tempo real

Conceitos Básicos

- Tarefa (task)
  - Segmento de código cuja execução possui atributo temporal próprio
  - Exemplo: método em OO, subrotina, trecho de um programa
- Deadline
  - Instante máximo desejado para a conclusão de uma tarefa
- Tempo real crítico (*hard real-time*)
  - Falha temporal pode resultar em **consequência castratróficas**
  - Necessário garantir requisitos temporais em projeto
  - Exemplo: usina nuclear, indústria petroquímica, mísseis
- Tempo real não crítico (*soft real-time*)
  - Requisito temporal descreve apenas **comportamento desejado**
  - Exemplo: multimídia

Conceitos Básicos

- Carga de tarefas (*task load*)
  - Descrição de quais tarefas deverão ser executadas
  - Estática: Limitada e conhecida em projeto
  - Dinâmica: Conhecida somente ao longo da execução
- Previsibilidade (*predictability*)
  - Capacidade de afirmar algo sobre o comportamento futuro do sistema
  - Determinista: Garante que todos os requisitos temporais serão cumpridos
  - Probabilista: Fornece uma probabilidade para o seu cumprimento

Suporte para Sistemas de Tempo Real

- Sistemas são construídas a partir dos serviços oferecidos por um **sistema operacional (SO)**
- O atendimento dos requisitos temporais depende não somente do código da aplicação, mas também da **colaboração do sistema operacional**
- No sentido de permitir **previsibilidade** ou pelo menos um **desempenho satisfatório**

Suporte para Sistemas de Tempo Real

- Muitas vezes os requisitos temporais da aplicação são tão rigorosos que o **SO é** substituído por um **simples núcleo de tempo real**
- Não inclui serviços como
  - Sistema de arquivos ou
  - Gerência sofisticada de memória
- Núcleos de tempo real oferecem uma funcionalidade mínima
- Mas são capazes de apresentar **excelente comportamento temporal**

Suporte para Sistemas de Tempo Real

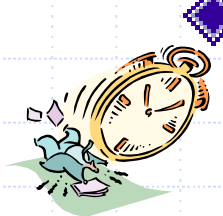
- **SO convencionais** são construídos com o objetivo de apresentar um **bom comportamento médio**
- **Distribuem os recursos** do sistema de forma **justa** entre as tarefas e os usuários
- **Não existe** uma preocupação com **previsibilidade** temporal
- Mecanismo como
  - Memória virtual
  - Fatias de tempo do processador
- Melhoram o desempenho médio do sistema
- Mas tornaram **mais difícil de fazer afirmações** sobre os tempos de uma tarefa em particular

Sistema Operacional de Tempo Real (SOTR)

- Aplicações com restrições de tempo real
  - **Menos interessadas** em uma **distribuição uniforme** dos recursos
  - **Mais interessadas** em **atender** requisitos tais como **períodos de ativação** e ***deadlines***
- Sistema operacional de tempo real
  - **Atenção** é dedicada ao **comportamento temporal**
  - Serviços são definidos não somente em termos **funcionais** mas também **termos temporais**

Aspectos Funcionais de um Sistema Operacional de Tempo Real

- Como qualquer SO, o SOTR procura **tornar a utilização** do computador
  - Mais eficiente
  - Mais conveniente
- Facilidades provindas de um SO de propósito geral são bem vindas em um SOTR
- Aplicações de tempo real são usualmente organizadas na forma de várias *threads* ou tarefas concorrentes
- Logo, um requisito básico de um SOTR é oferecer suporte para tarefas e *threads*



Tarefas e *Threads*

- Tarefas ou processos são abstrações que incluem
  - Um espaço de endereçamento próprio (possivelmente compartilhado)
  - Um conjunto de arquivos abertos
  - Um conjunto de direitos de acesso
  - Um contexto de execução formado pelos registradores do processador
  - Vários outros atributos
- *Threads* são tarefas leves
  - Únicos atributos são associados com o contexto de execução
- Chaveamento entre duas *threads* de uma mesma tarefa é muito mais rápida que o chaveamento entre duas tarefas
- Qualquer SO provê tarefas
  - é bom ter *threads* também

Comunicação entre Tarefas e entre *Threads*

- Uma **aplicação de tempo real é** tipicamente **um programa** concorrente
  - Formado por tarefas e/ou *threads*
  - Que se comunicam e se sincronizam
- Existem duas grandes classes de soluções para programação concorrente
  - Troca de mensagens
  - Variáveis compartilhadas
- A correta programação da comunicação e sincronização das tarefas
  - Garante o seu comportamento funcional
  - Mas não o seu comportamento temporal

Instalação de Tratadores de Dispositivos

- Sistemas de tempo real lidam com periféricos especiais, diferentes tipos de sensores e atuadores
  - Automação industrial
  - Controle de equipamentos em laboratório
- Projetista da aplicação deve ser capaz de
  - Desenvolver os seus próprios tratadores de dispositivos (*device drivers*)
  - Incorpora-los ao sistema operacional
- Muitas vezes a aplicação e o periférico estão fortemente integrados
  - Código da aplicação confunde-se com o código do tratador do dispositivo
  - Acontece no contexto dos sistemas embutidos (*embeded systems*)
  - SOTR deve permitir a aplicação instalar os seus próprios tratadores de interrupções

Temporizadores

- Aplicações precisam realizar operações que manipulam tempo
  - Ler a hora com o propósito de atualizar um histórico
  - Realizar determinada ação a cada X unidades de tempo
  - Realizar uma ação depois de cada Y unidades de tempo a partir de agora
  - Realizar determinada ação a partir do instante absoluto de tempo Z
- SOTR deve oferecer um conjunto de serviços que atenda estas necessidades
- Tipicamente o sistema possui pelo menos um temporizador (*timer*) implementado em *hardware*
  - Gera interrupções com uma dada freqüência
  - SOTR utiliza este temporizador para criar temporizadores lógicos

Aspectos Temporais

- Aplicações de SOTR **compartilham os mesmos recursos** do *hardware*
- Comportamento temporal do SOTR afeta o comportamento temporal da aplicação
- Por exemplo
  - Rotina do SO que trata as interrupções do *timer*
- O projetista da aplicação pode **ignorar** completamente **a função** desta rotina
- Mas **não pode ignorar** o seu **efeito temporal**
  - A interferência que ela causa na execução da aplicação
- Solicitar um serviço ao SO através de chamada de sistema significa
  - Processador **será ocupado** pelo código do SO
  - Capacidade da aplicação atender seus *deadlines* **passa a depender** da **capacidade do SO** em **fornecer o serviço** solicitado em um tempo que não inviabilize aqueles *deadlines*

Limitações do SO de propósito geral (SOPG)

- Diversas **técnicas populares** em SOPG são especialmente **problemáticas** quando as aplicações possuem requisitos temporais
- Mecanismo de memória virtual é capaz de gerar grandes atrasos
- Mecanismos tradicionais usados em sistemas de arquivos, fazem o tempo para acessar um arquivo variar muito
  - Ordenar a fila do disco para diminuir o tempo médio de acesso
- Aplicações de tempo real procuram minimizar o efeito negativo
- **Desativa o mecanismo sempre que possível**
- **Usa** o mecanismo apenas em **tarefas sem requisitos** temporais rigorosos
  - Acesso a disco feito por tarefas sem requisitos temporais

Limitações do SO de propósito geral

- Todos os SO desenvolvidos ou adaptados para tempo real mostram grande preocupação com a divisão do tempo do processador entre as tarefas
- Entretanto, o **processador é apenas um recurso** do sistema
- Memória, periféricos, controladores também deveriam ser escalonados visando atender os requisitos temporais da aplicação
- Muitos sistemas ignoram isto
  - Tratam os demais recursos da mesma maneira empregada por um SO de propósito geral

Limitações do SO de propósito geral

- Tipicamente qualquer SO dispõe de escalonamento baseado em prioridades
- Entretanto, a maioria dos SOPG geral **inclui mecanismos** que **reduzem** automaticamente a **prioridade** na medida que a tarefa consome tempo de processador
- Mecanismo utilizado para
  - Favorecer as tarefas com ciclos de execução menor
  - Diminuir o tempo médio de resposta do sistema
- Em sistemas de tempo real
  - A **justa distribuição** de recursos entre as tarefas é **menos importante** do que o atendimento dos requisitos temporais

Limitações do SO de propósito geral - Métricas

- Fornecedores de SOTR costumam divulgar métricas
  - Para mostrar como o sistema suporta aplicações de tempo real
- Estas métricas refletem a prática da construção de aplicações TR
  - Ligadas à desempenho
- Uma métrica muito utilizada é
  - O **tempo para chaveamento** entre duas tarefas
- Este tempo inclui
  - Salvar os registradores da tarefa que está executando
  - Carregar os registradores com os valores da nova tarefa
- Não inclui o tempo necessário para decidir qual tarefa vai executar
  - Depende do algoritmo de escalonamento

Limitações do SO de propósito geral - Métricas

- Outra métrica é a **latência**
  - Até o **inicio do tratador de uma interrupção** do *hardware*
- Eventos **importantes** e **urgentes** no sistema serão **sinalizados por interrupções**
- **Importante iniciar rapidamente** o tratamento destas interrupções
- Na análise do escalonabilidade
  - Tratador de interrupções corresponde a tarefa com a prioridade mais alta
  - Gera interferência sobre as demais tarefas
- Tempo definido pela forma como o *kernel* foi programado

Teoria de Escalonamento e Sistema Operacional

- Abordagem de escalonamento é determinada pela natureza da aplicação
  - Crítica ou não, carga estática ou não...
- Questão fundamental para quem vai usar um SOTR é
  - Determinar sua capacidade de suportar a abordagem de escalonamento
    - Desde que os atrasos e bloqueios do SOTR sejam conhecidos
- Maior obstáculo à aplicação da teoria de escolonamento é a dificuldade em determinar os tempos máximos de execução
- Dependem de vários fatores como
  - Fluxo de controle
  - Arquitetura do computador (*cache*, *pipeline*, etc)
  - Velocidade de *barramento* e processador
- Existem ferramentas experimentais nesta área, ainda não existem ferramentas com qualidade suficiente

Teoria de Escalonamento e Sistema Operacional

- Existem alguns caminhos para contornar este problema
- Em tempo de projeto, quando o código ainda não existe, é possível estimar os tempos de execução das rotinas
- Usar estas estimativas com os dados de entrada
  - Os resultados são estimativas
- Permite detectar durante o projeto problemas futuros com respeito aos tempos de resposta das tarefas
- **Detectar** a necessidade de **alterações antes** de iniciar a programação **muito melhor** do que fazer alterações depois que tudo já estiver programado

Teoria de Escalonamento e Sistema Operacional

- Uma vez que a aplicação esteja programada é possível analisar se as **estimativas** usadas em tempo de projeto foram **adequadas**
- Embora existam ferramentas que fazem isto automaticamente
  - São ainda projetos acadêmicos,
  - Sem a qualidade necessária para utilização em projetos
- Uma alternativa é medir os tempos de execução
- Não existe a **garantia** de que o **pior caso apareça** nas medições
- Uma **margem de segurança** pode ser associada

Teoria de Escalonamento e Sistema Operacional

- Grande obstáculo à aplicação da teoria de escalonamento é obter os atrasos e bloqueios associados com o SOTR
  - Em função de chamadas de sistema
  - Interrupções de hardware
  - Acesso a periféricos, ...
- Análise de escalonabilidade requer detalhes do SOTR
  - Maioria das vezes não são disponibilizados pelo fornecedor
  - Este quadro deverá mudar lentamente
- Se aplicação é do tipo soft *real-time*
  - Projetista escolhe um SO com boas propriedades
  - Escalonamento baseado em prioridade preemptivas
  - *Kernel* que executa com interrupções habilitadas
  - Chaveamento de contexto rápido
  - Baixa latência de interrupção

Tipos de Suporte para Tempo Real

- A diversidade de aplicações gera uma diversidade de necessidades
- Resulta em um leque de soluções com respeito aos suportes
- Com diferentes tamanhos e funcionalidades
- Podemos classificar os suportes de tempo real em dois tipos
  - Núcleo de tempo real (NTR)
  - Sistemas operacionais de tempo real (SOTR)
- NTR consiste de um pequeno *kernel*
  - Com funcionalidade mínima
  - Mas excelente comportamento temporal
  - Indicada para, por exemplo, o controlador de uma máquina industrial

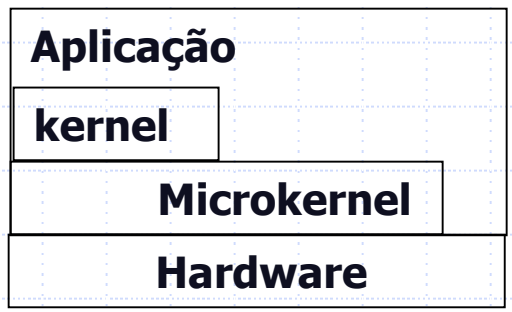
Tipos de Suporte para Tempo Real

- SOTR é um SO completo
  - Funcionalidade típica de propósito geral
  - Mas cujo *kernel* foi adaptado para melhorar o comportamento temporal
  - Qualidade temporal do kernel adaptdo varia de sistema para sistema
    - Alguns são completamente reescritos para tempo real
    - Outros recebem apenas algumas poucas otimizações

	Funcionalidade	
	mínima	completa
Previsibilidade maior	Núcleo de Tempo Real	Futuro
Previsibilidade menor	Qualquer Núcleo Simples	SO Adaptado

Microkernel

- Sistemas podem ser organizados em camadas
- Subindo na estrutura de camadas
  - Os serviços tornam-se mais sofisticados
  - O comportamento temporal menos previsível
- Aplicação tem a sua disposição uma gama completa de serviços
- Quando os requisitos temporais da aplicação aumentam
  - Pode acessar diretamente o microkernel
  - E até mesmo o hardware
- Apropriado para sistemas onde
  - Apenas uma aplicação é executada



Escolha de um suporte de Tempo Real

- Difícil comparar diferentes SOTR
  - Diferentes abordagens de escalonamento
  - Desenvolvedores de SOTR publicam métricas diferentes
  - Desenvolvedores de SOTR não publicam métricas diferentes
  - Detalhes internos sobre o *kernel* não estão normalmente disponíveis
  - Métricas fornecidas foram obtidas em plataformas diferentes
  - Conjunto de ferramentas para desenvolvimento que é suportado varia
  - Ferramentas para monitoração de depuração das aplicações vairam
  - Linguagens de programação suportadas em cada SOTR são diferentes
  - Conjunto de periféricos suportados por cada SOTR varia
  - Conjunto de plataformas de *hardware* suportados varia
  - Cada SOTR possui um esquema para a incorporação de *device-drivers*
  - Possuem diferentes níveis de conformidade com os padrões
  - Política de licenciamento e custo associado variam

POSIX em Tempo Real

- Posix é um padrão para SO
  - Baseado no Unix
  - Criado pela IEEE (*Institute of Electrical and Eletronic Engineers*)
- Posix define as interfaces do SO
  - Mas não sua implementação
  - Posix API (*Application Programming Interface*)
- SOTR possuem uma API proprietária
  - Aplicação fica amarrada aos conceitos e às primitivas do sistema em questão
- Usando um SOTR que é compatível com Posix,
  - Aplicação fica amarrada aos conceitos e às primitivas do Posix
- Muitos SOTR atualmente já suportam a API do Posix

Linux para Tempo Real

- Linux é um SO com fonte aberto,
  - Inclui multiprogramação, memória virtual, bibliotecas compartilhadas, protocolos de rede TCP/IP, etc
- Linux convencional segue o estilo de um *kernel* Unix tradicional
  - *Kernel* monolítico, não é baseado em *microkernel*
  - **Não é apropriado** para a maioria das aplicações de tempo real
- *Kernel* do Linux possui um recurso que facilita sua adaptação
  - Aceita "módulos carregáveis em tempo de execução"
  - Podem ser incluídos e excluídos do *kernel* sob demanda

Conclusões

- Área de SOTR é muito dinâmica
- Novos sistemas ou novas versões dos sistemas existentes são apresentadas a todo momento
- Mensagem central:
  - O comportamento temporal da aplicação de tempo real depende tanto da aplicação quanto do sistema operacional
  - Desta forma, a seleção do SOTR a ser usado depende fundamentalmente dos requisitos temporais da aplicação em questão
- Não existe um SOTR melhor ou pior para todas as aplicações
- A diversidade de aplicações de tempo real existente gera uma equivalente diversidade de SOTR