Sumário

- O que é o tempo-real.
- Tipos de sistemas de tempo-real.
- Escalonamento.
- Exemplos de sistemas de tempo-real.

Definição

- Tempo de resposta:

  Intervalo de tempo entre o estímulo e a resposta do sistema.

- Sistema de tempo-real:

  Um sistema de tempo real é um sistema em cuja especificação são explicitamente requeridos tempos de resposta pré-determinados.

Tipos de sistemas de tempo-real

- Estrito (hard real-time).

  Imperativo cumprir as metas. Ex: risco de perca de vidas humanas (controlo de avião, de central nuclear, etc.).

- Lato (soft real-time).

  Tempo é importante, o sistema está concebido para cumprir metas embora seja aceitável que por vezes estas não sejam atingidas. Ex: aquisição de dados para controlo da produção de um processo fabril.

Características: facilidades de tempo-real.

- Especificar o instante do início de execução de uma tarefa.
- Especificar o instante em que um determinado resultado deve ser produzido.
- Lidar com cenários aonde não é possível assegurar TODAS as metas.

  Situações de sobrecarga. Quais os serviços mínimos.

- Lidar com situações em que as metas são alteradas dinamicamente.

  Emergências. Mudanças de planos.

Depressa diferente de “a tempo”

- É necessário:

  Elevado desempenho.

- No entanto:

  Mas baixa carga e rapidez não...

  ... oferece **garantias**.

- Previsibilidade:

  É possível prever o comportamento do sistema nos diversos domínios (valor, tempo, etc).

Para oferecer garantias é necessário:

- Determinismo:

  O estado seguinte depende exclusivamente do estado actual e das entradas do sistema.

- Pior-caso:

  Considerar o funcionamento no pior caso (o que pressupõe descobrir qual é o pior caso!).

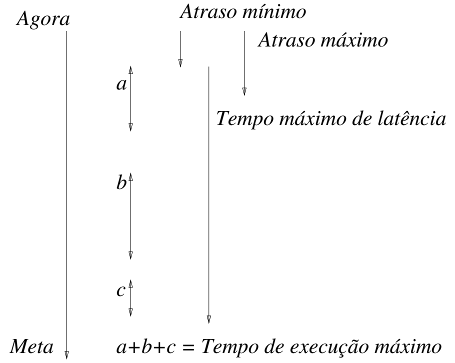
- No entanto:

  Importa minimizar a redundância.

Domínio temporal de um processo

- Meta: *Altura em que a tarefa deve ter terminado a sua execução.*
- Atraso mínimo: *Atraso mínimo após o qual se pode iniciar a tarefa.*
- Atraso máximo: *Atraso máximo antes do qual se deve iniciar a tarefa.*
- Tempo de execução máximo.
- Tempo máximo de latência: *Tempo máximo entre a activação e a terminação da tarefa.*

Domínio temporal (ilustração)



O problema do escalonamento.

- O sistema deve executar um conjunto de tarefas, cada uma com a sua meta temporal.
- Cada tarefa necessita de aceder a recursos partilhados:

  Processador.

  Rede.

  Memória.
- O escalonamento consiste em definir uma atribuição destes recursos às tarefas de modo a que as metas sejam cumpridas.

Escalonadores dinâmicos.

- Toma decisões em tempo de execução com base nas tarefas que existem para executar.
- São flexíveis.
- Dificuldade: calcular o escalonamento pode ser substancialmente demorado.

Escalonadores estáticos.

- Calcula o escalonamento antes da execução.
- Gera uma tabela de despacho em tempo de compilação.
- Necessita de informação completa acerca das tarefas (restrições de exclusão mútua, tempo de execução máximo, metas).
- Baixa sobrecarga em tempo de execução.

Teste de "escalonabilidade"

- Teste para determinar se existe algum escalonamento possível (que cumpra as metas) para um determinado conjunto de tarefas.

  Suficiente.

  Exacto.

  Necessário.
- Escalonador óptimo:

  Se encontra um escalonamento quando um teste exacto indica que este existe.

Escalonador óptimo

- NP-complexo (não polinomial):

  Geralmente intratável computacionalmente.

- Teste de suficiência.

  Mais simples mas por vezes desperdiça recursos.

- Testes de necessidade.

  Podem ser simples mas não chegam.

Tipos de tarefas

- Periódicas:

  São executadas periódicamente, com um período conhecido à partida.

  $$\text{percentagem de utilização} = \sum \text{consumo}_i/\text{período}_i \leq \text{número de processadores}$$

- Esporádicas:

  Deve ser conhecido pelo menos o tempo mínimo entre dois pedidos de execução dessas tarefas.

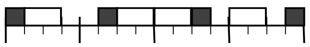
Escalonamento dinâmico óptimo

- Não é possível prever o futuro.
- Define-se óptimo como sendo:

  Um escalonador que encontra uma solução nos casos em que um escalonador com conhecimento do futuro encontraria.

- Quando exitem restrições de exclusão mútua esta tarefa é impossível.

Escalonamento dinâmico (exemplo)



Tarefa a negro deve ser executada imediatamente. Sempre possível quando se consegue prever o momento em que é lançada. Impossível de garantir no caso contrário.

Escalonamento dinâmico: rate monotonic (hipóteses).

H1: Todas as tarefas com metas estritas T_i são periódicas.

H2: Todas as tarefas são independentes (não existem restrições de exclusão mútua).

H3: As metas de T_i são iguais ao seu período.

H4: O tempo máximo de execução é conhecido e constante.

H5: O tempo de troca de contexto é desprezável.

H6: A percentagem de utilização está restringida a:

$$\mu = \sum C_i/P_i \leq n(2^{1/n} - 1) \rightarrow 0.7$$

Escalonamento dinâmico: rate monotonic

- Algoritmo:

  Prioridades fixas com base no período.

  A tarefa com menor período recebe a prioridade mais alta.

  As tarefas são escalonadas por ordem de prioridade.

  Algoritmo óptimo para sistemas com um processador.

Escalonamento dinâmico: earlier deadline first.

- Hipóteses

  H1-H5 do caso anterior.

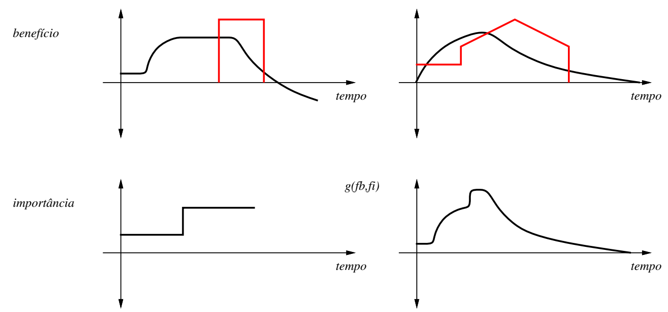
- Algoritmo:

  Prioridades dinâmicas.

  Após cada evento, a tarefa com a meta mais próxima tem maior prioridade.

Escalonamento dinâmico: funções de custo

Usado no sistema Alpha[Jensen].



Escalonamento estático.

- Leitura do estado exterior em momentos pré-determinados.
- Sistema despoletado pelo tempo.

  De x em x unidades de tempo, o despacho é executado.

  A tabela de despacho é gerada em tempo de compilação.

- Procura em árvore:

  O escalonamento é encontrado usando técnicas de procura em árvore.

  O algoritmo requer conhecimento acerca das tarefas, períodos, tempos de execução, restrições de exclusão mútua, etc.

Prioridade

- Indicação da importância relativa dos processos.
- Pode ser utilizada pelo escalonador para definir quais os processos que devem ter acesso ao processador.
- É de difícil utilização.

Dificuldade da gestão das prioridades

- Um processo de alta prioridade pode ficar dependente de um recurso detido por um processo de baixa prioridade.

  Processos de prioridade intermédia podem impedir o processo de baixa prioridade de libertar o recurso e, deste modo, impedir o progresso do processo de alta prioridade.

- A este fenómeno chama-se inversão de prioridade.

Inversão de prioridade

- Uma solução:

  Aumentar a prioridade da tarefa que detém o recurso de modo a esta não ser ultrapassada por tarefas de média prioridade.

  No limite todas têm a mesma prioridade e pode haver interblocagem.

Priority ceiling.

- Define-se tecto de uma região à maior prioridade de uma tarefa que pode aceder a essa região.
- Quando uma tarefa bloqueia outra mais prioritária, aumenta a sua prioridade para o nível da tarefa bloqueada até libertar o recurso.
- Uma tarefa só entra numa nova região se a sua prioridade é maior do que o tecto de todas as regiões em uso (não contando com a própria tarefa).

  Ideia chave: não vai impedir o progresso das tarefas que detém recursos a não ser que seja mais prioritária.

Tolerância a faltas

- Uma meta pode não ser atingida se:

  Os cálculos do pior tempo de execução estão errados.

  As hipóteses feitas ao gerar um escalonamento estavam erradas.

  O escalonador tem um "bug".

  O sistema está a operar fora dos parâmetros previstos.

Recuperação

- Caso uma meta não seja atingida:

  Repõe-se o estado do processo.

  Tenta-se um algoritmo alternativo.

  Ou executa-se um algoritmo correctivo.

MARS

- MAintainable Real-time System

  Projecto de investigação da Universidade de Viena.

- Desenvolver uma arquitectura distribuída para sistemas de tempo-real estrito que permita verificar durante a fase de projecto o comportamento no tempo do sistema.

MARS: Objectivos

- Metas garantidas:

  Dadas hipóteses de faltas e de carga garantir o comportamento temporal (com os recursos necessários).

- Teste:

  A arquitectura está concebida de modo a facilitar o teste do seu comportamento no tempo (não interferência dos mecanismos usados no teste).

MARS: Objectivos (continuação)

- Manutenção:

  Recorrendo a uma concepção modular.

- Tolerância a faltas.
- Desenvolvimento sistemático de sofware:

  Através de metodologias e ferramentas bem definidas.

MARS: Arquitectura despoletada pelo tempo

- Observação do estado despoletada pelo tempo:

  Os objectos controlados são lidos periódicamente, em instantes pré-determinados do tempo.

  Concretiza um controlo de fluxo implícito (dos eventos exteriores assinalados ao sistema).

- Mensagens de estado:

  Mensagens que propagam no sistema o valor de variáveis.

  Propagadas em instantes pré-determinados.

MARS: Entradas saídas

- E/S inteligentes:

  Com a capacidade de armazenar a informação acerca do mundo exterior até ao instante da observação.

  Por vezes, memorizar o instante exacto em que um evento ocorreu.

  Permite obter informação mais pormenorizada do que a oferecida pelo intervalo de observação.

MARS: Estrutura (perspectiva sistema).

- Perspectiva sistema:

  Um conjunto de sub-sistemas autónomos, interligados em rede, designados por agregados que comunicam através de mensagens de estado.

- Agregado:

  Conjunto de Unidades Tolerantes a Faltas, interligadas por um canal de comunicação tempo-real.

MARS: Agregado.

- As FTU também comunicam exclusivamente através de mensagens de estado.
- Protocolo de acesso ao meio: multiplexagem no tempo.
- Tolerância a faltas concretizada ao nível do agregado: um serviço de filiação identifica quais as unidades operacionais.
- Protocolos de comunicação fiável sobre o canal de tempo-real.

MARS: Estrutura (continuação).

- Unidade tolerante a faltas:

  Componentes operando em redundância activa.

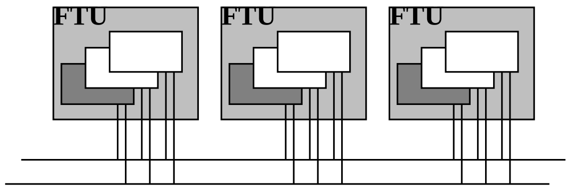
- Componentes:

  Manquinaria e software que respeita o modo de falha silencioso.

- Tarefa:

  No sentido das aulas anteriores.

O sistema MARS



MARS: Operação (execução de tarefas).

- Uma tarefa é despoletada pela leitura de uma mensagem de dados.
- Comunicação entre tarefas por mensagens de estado.
- A tarefa executa-se sem comunicação ou sincronização até ao momento em que as mensagens de estado resultantes são produzidas.
- Deste modo, o tempo de execução de uma tarefa pode ser avaliado em tempo de compilação.
- Ferramentas que ajudam o programador a realizar código que se executa num intervalo bem definido.

MARS: Operação (comunicação)

- No fim da execução de uma tarefa os resultados são difundidos através de mensagens de estado.
- O tempo de execução dos protocolos de comunicação é previsível.
- Um componente de software concretiza um serviço de sincronização e de filiação.

MARS: Operação (escalonamento e modos)

- Escalonamento e sincronização:

  Escalonamento estático: todas as dependências entre tarefas e requisitos de sincronização são conhecidas à partida.

  Em tempo de execução o despacho segue decisões inscritas na tabela de despacho.

- Mudanças de modo:

  Diferentes modos de operação com diferentes escalonamentos.

  Exemplo: modo normal e modo de emergência.

  Mudança de modo activada assim que necessário.

MARS: Tolerância a faltas (modelo de faltas).

- Tolera faltas temporárias ou permanentes da maquinaria.
- Modo de falha silencioso (paragem sem acções erradas).
- Cada FTU está operacional enquanto um dos componentes replicados estiver a funcionar.
- Probabilidades esperadas:

  falha permanente: 10^4 *FITS; falha temporária:* 10^6 *FITS; recuperação de falha temporária: 10s; reparação de falha permanente: 1 dia.*

MARS: Tolerância a faltas (redundância)

- Em cada componente.

  De modo a garantir o modo de falha silenciosa.

  Para detectar falhas transitórias, cada tarefa é executada duas vezes e os resultados comparados.

- Ao nível do aglomerado.

  Protocolo de filiação detecta falha de unidades.

MARS: Tolerância a faltas (determinismo das réplicas).

- Todas as tarefas são determinísticas.
- Uma tarefa não pode ler os relógios locais:

  apenas pode depender do seu tempo de lançamento que é igual em todas as réplicas.

- Quando é necessário realizar uma mudança de modo, o protocolo de filiação garante que todas as réplicas acordam o instante em que esta vai ser realizada.
- Em caso de falhas o componente aborta a sua execução.

MARS: Tolerância a faltas (re-integração de componentes)

- Estado de um componente:

  Parte estática (código, etc).

  Parte dinâmica (deve ser transferida).

- O escalonador prevê pontos de re-integração:

  Em que os componentes difundem a componente dinâmica do seu estado.

  A sobrecarga da re-integração não compromete o desempenho no domínio do tempo.

  Um componente deve esperar pelo ponto de re-integração para receber o estado actualizado.

MARS: Tolerância a faltas (manutenção)

- Falhas de "hardware" são cobertas pelos mecanismos de tolerância a faltas.
- Alterações ao código dependem do tempo de execução.
- Se o tempo máximo de execução não é alterado, não é necessário alterar o escalonamento.
- Caso contrário é necessário definir um novo modo de operação e executar uma mudança de modo.

DELTA-4/XPA: antecedentes

- Delta-4 é uma arquitectura distribuída tolerante a faltas.

  Inicialmente concebida para aplicações genéricas e ambientes abertos.

- Delta-4/XPA (eXtra Performance Architecture).

  Variante da arquitectura para tempo-real.

DELTA-4/XPA: Objectivos

- Parte da observação de que existem dois tipos de problemas: *Carga-limitada e carga-ilimitada.*
- Carga-limitada: *É possível prever qual é a carga máxima do sistema.*
- Carga-ilimitada: *Não é possível definir com precisão qual vai ser o envelope operacional. A maioria dos eventos são esporádicos.*
- Aproximação XPA: tentar atíngir um comportamento de tempo-real estrito para uma dada carga prevista e mudar para um modo do tipo melhor-esforço em sobrecarga.

DELTA-4/XPA: Arquitectura

- Infra-estrutura de comunicação:

  Rede abstracta de tempo-real.

  Concretizada utilizando mecanismos de redundância.

- Serviço de tempo:

  Protocolo de sincronização de relógios (acordo à posteriori).

  Optimizado para redes locais.

DELTA-4/XPA: Arquitectura (continuação).

- Comunicação em grupo:

  Protocolos de filiação e comunicação em grupo, oferecendo garantias temporias.

- Gestão de grupos:

  Protocolos de gestão de replicação.

- Ambiente de suporte:

  Ambiente de execução de aplicações segundo o modelo ODP.

DELTA-4/XPA: Opções de concepção (metas temporais)

- Definir uma política para operação em sobrecarga.
- Usar o conceito de precedência: definem-se diversas classes de precedência.
- Dentro de cada classe de precedência usa-se um escalonador do tipo "meta mais próxima primeiro".
- Projecção das classes de precedência em classes de prioridades na rede de tempo-real.

DELTA-4/XPA: suporte à computação

- Um conjunto de componentes que reage a eventos externos e que comunica através de RPCs.
- Replicação é transparente e suportada por diversos protocolos de gestão da replicação (activa, passiva e semi-activa).

DELTA-4/XPA: preêmpção e determinismo das réplicas

- O código das réplicas é preparado para testar periódicamente mensagens urgentes (em pontos bem definidos do código).
- Quando uma mensagem urgente chega, o processo que vai receber essa mensagem é escalonado.
- O processo reconhece a existência da mensagem e coordena com as restantes réplicas o instante "lógico" em que deve processar a mensagem.
- Geralmente: modelo líder-seguidor (um processo decide e avisa os restantes)

Resumo

- Garantir o comportamento no tempo.
- Especificação de metas e escalonamento.
- Tolerância a faltas.
- Sistemas Delta-4 e MARS.