Fundamentos de Sistemas Concorrentes

Alcides Calsavara

Concorrência na vida real

- Cruzamento de veículos
  - Semáforo (divisão no tempo)
  - Preferencial pelo sentido do veículo
  - Preferencial pelo tipo do veículo
  - Viaduto (replicação)
  - Ordem de chegada (semáforo amarelo piscando)
  - Guarda (divisão no tempo ou por ordem de chegada)

Conceitos fundamentais

- Causas de concorrência:
  - Compartilhamento de recursos
  - Ações logicamente excludentes no tempo
    - Falar ao telefone e dirigir
    - Falar ao telefone e assistir a uma aula
- Modelos de interação:
  - Competição
  - Colaboração

Conceitos fundamentais

- Elementos estruturantes:
  - Região crítica
  - Sincronização
- Unidades de execução:
  - Processo
  - Thread
  - Task (tarefa)

Conceitos fundamentais

- Ambientes de execução:
  - Centralizado (comunicação através de memória compartilhada)
    - Processador único
    - Múltiplos processadores
  - Distribuído (comunicação através de mensagens de rede)
    - Rede (LAN, WAN, MANET, VANET, DTN, …)
    - Cluster (datacenter)
    - Grid

Conceitos fundamentais

- Modos de execução:
  - Paralelismo real
    - múltiplos processadores
  - Paralelismo por escalonamento
    - processador único
- Modos de concorrência:
  - Entre processos (centralizado ou distribuído)
  - Interno a um processo

Eventos

- Relacionamentos entre eventos:
  - Antes
  - Durante
  - Depois
- **Restrições de sincronização**:
  - ***Serialização***: evento A deve ocorrer antes do evento B
  - ***Exclusão mútua***: evento A e evento B não podem ocorrer ao mesmo tempo

Ordenação de eventos

- **Relógio**
  - Basta comparar os tempos de ocorrência dos eventos
  - Nem sempre há um relógio universal (implementação relativamente cara)
  - Nem sempre os relógios possuem precisão fina o suficiente para a aplicação
- **Técnicas de software para garantir restrições de sincronização**

Modelo de execução

- **Paralelo**
  - Múltiplos processadores paralelos
  - Eventos ocorrem em ordem imprevisível
- **Multithreaded**
  - Processador único com threads paralelas
  - Ordem de execução (e consequentes eventos) não é controlada pelo programador

CENÁRIOS EQUIVALENTES PARA FINS DE SINCRONIZAÇÃO

Exclusão mútua através de mensagens

- Exemplo: João e José são supervisores de uma usina nuclear, no mesmo turno de trabalho. Ambos têm a mesma função e poder de intervenção na operação da usina em caso de emergência. Eles trabalham em locais diferentes e não podem almoçar ao mesmo tempo, tal que seja garantido que a usina tem, ao menos, um supervisor ativo o tempo todo.
  - Qual esquema de troca de mensagens (por exemplo, chamadas telefônicas) garante que João e José não almocem ao mesmo tempo, sendo que qualquer um dos dois pode almoçar primeiro?

Serialização por mensagens

Thread 1 (João)

1. Tome café da manhã
2. Trabalhe
3. Almoce
4. Chame José

Thread 2 (José)

1. Tome café da manhã
2. Trabalhe
3. Aguarde uma chamada
4. Almoce

João e José almoçam **sequencialmente**, primeiro João e depois José.

Programas concorrentes

- *Dois eventos são **concorrentes** quando não se consegue dizer através da observação do código fonte qual evento ocorre primeiro.*
- Execução não-determinista
- Depuração difícil: praticamente impossível depurar através de testes
- Exige programação cuidadosa

Variável compartilhada

- Duas ou mais threads podem compartilhar a mesma variável, para operações de leitura e escrita
  - A comunicação entre as threads ocorre através da execução dessas operações
- Situações:

1. Escrita concorrente com leitura
   - Resolvido por serialização: escritor envia mensagem para o leitor para garantir leitura do valor mais recente
2. Escritas concorrentes
3. Atualizações (leitura seguida de escrita) concorrentes

Escritas concorrentes

Thread A

1. x := 1
2. Escreva x

Thread B

1. x := 2

a. Qual será o valor final de x ?
b. Qual valor será escrito?

Caminhos de execução possíveis**:**

1. A1 < A2 < B1
2. B1 < A1 < A2
3. A1 < B1 < A2

Escritas concorrentes

Thread A

1. x := 1
2. Escreva x

Thread B

1. x := 2

a. Qual será o valor final de x ?
b. Qual valor será escrito?

Caminhos de execução possíveis**:**

1. A1 < A2 < B1 ==> 1
2. B1 < A1 < A2
3. A1 < B1 < A2

Escritas concorrentes

Thread A

1. x := 1
2. Escreva x

Thread B

1. x := 2

a. Qual será o valor final de x ?
b. Qual valor será escrito?

Caminhos de execução possíveis**:**

1. A1 < A2 < B1 ==> 1
2. B1 < A1 < A2 ==> 1
3. A1 < B1 < A2

Escritas concorrentes

Thread A

1. x := 1
2. Escreva x

Thread B

1. x := 2

a. Qual será o valor final de x ?
b. Qual valor será escrito?

Caminhos de execução possíveis**:**

1. A1 < A2 < B1 ==> 1
2. B1 < A1 < A2 ==> 1
3. A1 < B1 < A2 ==> 2

Atualizações concorrentes

Thread A

1. contador := contador + 1

Thread B

1. contador := contador + 1

- Onde está a concorrência neste caso?
- Qual o valor final de contador, supondo que o seu valor inicial seja 0 ?
- Existe mais de um caminho de execução?
- A instrução de incremento é **atômica**?

No caso geral, NÃO!

Atualizações concorrentes

Thread A

1. x := contador
2. contador := x + 1

Thread B

1. y := contador
2. contador := y + 1

Código reescrito em formato “mais próximo”
da linguagem de máquina (sem atomicidade no incremento)

Qual o valor final de contador com o
caminho de execução a seguir?

A1 < B1 < A2 < B2

Atualizações concorrentes

Thread A

1. x := contador
2. contador := x + 1

Thread B

1. y := contador
2. contador := y + 1

Código reescrito em formato “mais próximo”
da linguagem de máquina (sem atomicidade no incremento)

Qual o valor final de contador com o
caminho de execução a seguir?

A1 < B1 < A2 < B2 ==> 1

Desafios de projeto

- Sincronização
  - Exclusão mútua
  - Deadlock
  - Livelock
  - Starvation
  - Fairness
- Arquitetura de sistema
  - Desempenho: escalabilidade e tempo real
  - Consistência *versus* disponibilidade (replicação de dados)
  - Granularidade
  - Tolerância a falhas

Abstrações

- Região crítica: pré-condição, invariante, pós-condição
- Semáforo
- Sinalização
- Travas e barreiras
- Rendezvous
- Monitor
- Objeto atômico
- Objeto imutável

Linguagens

- Modelagem e especificação
  - Pseudo-código
  - Diagramas (estados, atividades, sequência)
- Programação
  - Java
  - C++
  - C#
  - Python
  - Ada
  - SR (by Gregory Andrews)