PROGRAMAÇÃO DISTRIBUÍDA
Aula 01

Prof. Henrique Mongelli
mongelli@dct.ufms.br
345-7460

Cenário-Exemplo

Vários carros desejam ir de um ponto A a um ponto *B*. Eles podem competir por espaços em uma mesma estrada ou acabam seguindo uns aos outros ou competindo por posições. Ou poderiam andar em vias paralelas, desta forma chegando quase que ao mesmo tempo sem invadir a via do outro. Ou poderiam trafegar em rotas diferentes usando estradas separadas.

Computação Concorrente

- Existem múltiplas tarefas a serem feitas. (carros em movimento)
- Cada tarefa pode ser executada:
- uma de cada vez em um único processador (uma única estrada);
- em paralelo em múltiplos processadores (pistas em uma estrada); ou,
- em processadores distribuídos (estradas separadas).

Características

- Um programa concorrente contém dois ou mais processos que trabalham juntos para executar uma tarefa.
- Cada processo é um programa seqüencial.
- Programa seqüencial ⇨ único *thread* de controle.
- Programa concorrente ⇨ múltiplos *threads* de controle.

Comunicação

- Os processos em um programa concorrente trabalham juntos comunicando-se entre si.
- A comunicação pode ser feita através de:
  - variáveis compartilhadas
  - troca de mensagens
- Independente da forma de comunicação, os processos precisam sincronizar-se:
  - exclusão mútua (seções críticas)
  - sincronização condicional

Breve Histórico

- Apareceu na década de 60 dentro do contexto de Sistemas Operacionais.
- A motivação foi a criação de unidades de *hardware* denominadas canais ou dispositivos de controle.
- Estes dispositivos funcionam independente de um processador de controle e podem fazer operações de E/S concorrentemente com a execução de um programa.

- Um canal comunica-se com o processador central através de uma interrupção.
- Com a introdução dos canais, partes de um programa poderiam funcionar de forma imprevisível.
- Logo após o aparecimento dos canais, foram desenvolvidos as máquinas multiprocessadas.
- Estas máquinas permitem que aplicações diferentes sejam executadas em processadores diferentes ao mesmo tempo.

- Permite também que uma aplicação possa ser executada mais rapidamente se puder ser reescrita de forma a utilizar múltiplos processadores.
- Como sincronizar as atividades de processos concorrentes?
- Como utilizar múltiplos processadores para que uma aplicação seja executada mais rapidamente?

Algoritmos Distribuídos

- Algoritmos que foram desenvolvidos para serem executadas em muitos processadores “distribuídos” em uma grande área geográfica.
- Atualmente, o termo cobre algoritmos que são executadas em redes locais e em multiprocessadores de memória compartilhada.

Aplicações

- processamento de informações distribuídas;
- computação científica;
- controle de processos de tempo real

Tipos de Algoritmos Distribuídos

- Método de comunicação entre processos: memória compartilhada, mensagens ponto-a-ponto, difusão de mensagens(*broadcast*) e chamadas remotas a procedimentos (RPC).
- Modelo de Execução (*Timing* model): completamente síncronos, completamente assíncronos, parcialmente síncronos.

- Modelo de falha: *hardware* completamente confiável ou pode-se admitir alguma falha. Na presença de falha: o processo pode parar com ou sem aviso; pode falhar brevemente; ou pode apresentar falhas graves e o sistema funcionar de forma arbitrária. Também podem ocorrer falhas na comunicação: perda ou duplicação de mensagens.

- Problemas abordados: alocação de recursos, comunicação, consenso entre processadores distribuídos, controle de concorrência em bancos de dados, detecção de *deadlock*, instantâneos globais (*global snapshots*), sincronização, e implementação de vários tipos de objetos.

Características

- Apresentam um alto grau de incerteza e mais independência de atividades, entre elas:
  - número de processadores desconhecido;
  - topologia de rede desconhecida;
  - entradas independentes em diferentes locais;
  - vários programas sendo executados de uma só vez, começando em tempos diferentes, e operando a velocidades diferentes.

– não-determinismo dos processadores;
– tempos de envio de mensagens incertos;
– desconhecimento da ordenação das mensagens;
– falhas de processadores e comunicação

Motivação

- permitir iteração entre usuários;
- compartilhamento de recursos/dados e serviços;
- em potencial: desempenho melhor, disponibilidade maior.

Software tem que estar distribuído, controle deve estar distribuído.

Requer nova forma de programação/projeto de algoritmos que levam em conta:

- paralelismo e não-determinismo;
- controle descentralizado;
- atomicidade de ações;
- tomada de decisão baseada em informação (parcial) sobre os dados do sistema;
- otimização do número de mensagens a serem trocadas;
- levar em conta a possibilidade de falhas.