PROGRAMAÇÃO DISTRIBUÍDA
Aula 02

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

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.

Modelo - Hardware

- Conjunto de máquinas autônomas, heterogêneas, com características de processamento diferentes e recursos próprios.
- Comunicação através de um meio de comunicação com características próprias de desempenho e confiabilidade.

Modelos - Programas

- Composto de um número arbitrário de processos fisicamente distribuídos e executando de forma paralela, interagindo através do envio de mensagens para troca de dados e sincronização.
- Características:
  - Controle descentralizado;
  - Impossibilidade de determinação de um estado global do sistema;

Problemas Específicos Sincronização

- Sincronizar é necessário para:
  - manter estados de processos consistentes (replicado- complementar);
  - determinar uma ordem de ocorrência de eventos;
  - atingir um consenso.
- A dificuldade de sincronização é devida a:
  - tempo de comunicação imprevisível;
  - falhas independentes dos nós e na comunicação;
  - falta de sincronismo de relógios.

- Técnicas para sincronização:
  - transações atômicas;
  - induzir uma ordem no sistema:
    - relógios lógicos;
    - *vector timestamps.*

Problemas Específicos
Tolerância a Falhas

- Algoritmos distribuídos deveriam levar em conta a possibilidade de falhas e garantir um funcionamento correto apesar das mesmas.

Tolerância a falhas envolve:

- **detecção de falhas**
  - – informação adicional;
  - – dados redundantes;
  - – *timeouts*.
- **recuperação do estado**
  - – *forward recovery*
    - reexecuta operação ou tenta operação alternativa.
  - – *backward recovery*
    - periodicamente salva estados (*checkpoints* síncronos ou assíncronos).

Problemas Específicos Compartilhamento

- Compartilhamento de recursos (unidades de disco, fitas, impressão, ...);
- Serviços;
- Compartilhamento de dados(BD completo, registros, ...);
- Controle
  - seqüenciador;
  - mestre-escravo.

Objetivos do compartilhamento:

- reduzir o custo total;
- garantir um bom desempenho do sistema:
  - redução do tempo de espera;
  - aumento na taxa de tarefas por unidade de tempo (*throughput*).
- evitar congestionamentos;
- evitar *deadlocks*;
- evitar *starvation;*
- garantir justiça;
- evitar *livelock*

Diferenciação de algoritmos

- Quanto ao mecanismo de comunicação.
  - Envio ponto-a-ponto
    - síncrona
    - assíncrona
  - Envio *broadcast*
  - Chamada remota de procedimentos (RPC)

- Quanto ao modelo de execução (*timing model*)
  - síncrono: computação e comunicação prosseguem em *lockstep*. (arquitetura paralela com relógio único)
  - parcialmente síncrono: limite na velocidade relativa dos nós; limite no tempo máximo de comunicação.
  - assíncronos: velocidade de execução e ordem de execução e comunicação arbitrários.

- Quanto ao modelo de falha
  - Ausência total de falhas.
  - Falhas tipo omissão temporária.
  - Falhas quaisquer.
- Quanto à topologia
  - número conhecido de processos e topologia de interconexão fixa.
  - número desconhecido de processos mas topologia não muda.
  - conjunto dinâmico de processos e topologia desconhecida.

Modelos de Processamento Distribuído

1. Grau de sincronismo de processamento e comunicação.
2. Tipos de falhas nos processos.
3. Tipos de falhas de comunicação.
4. Topologia de interconexão dos processos.
5. Processamento determinístico vs. randômico.

Modelo de Execução

- Síncrono
- Parcialmente síncrono
- Assíncrono:
  - não existe limite de tempo para troca de mensagens.
  - não há limite para defasagem entre relógios.
  - não há limite para execução de cada instrução.

Falhas nos processos

1. *Failstop*: processo pára sua execução e isto é detectável.
2. *Crash*: processo pára sua execução.
3. *Send omission*: *crash* ou mensagens não são enviadas temporariamente.
4. *Receive omission*: *crash* ou mensagens não são recebidas temporariamente.
5. *General omissions*: *send omission* ou *receive omission*.
6. Falha maliciosa.

Falhas de comunicação

- Partição: conexão entre processos é quebrada.
- Omissões: perdas isoladas de mensagens.
- Arbitrária: pode haver corrupção/adição de mensagens.

Topologia de Interconexão

- Dependendo do controle a ser adotado existe alguma topologia lógica mais adequada: grafo direcionado.
- Topologias usuais:
  - grafo completo
  - estrela
  - anel
  - árvore
  - grafo qualquer

Determinístico vs. Randômico

- Determinístico: processa altera seu estado dependendo apenas do estado anterior do conjunto de mensagens recebidas.
- Randômico: existe uma transação espontânea de estado, que é randômica.

O Modelo Síncrono

- Parte estática:

  O programa é um grafo direcionado $G=(V,E)$, $n=|V|$ é o número de nós. E é o conjunto de ligações entre os nós.

  Para cada nó i:

  $OUT\ N_i = \{j \in V / \text{existe } (i,j) \text{ em } G\}$

  $IN\ N_i = \{j \in V / \text{existe } (j,i) \text{ em } G\}$

  distância(i,j) = número de arestas do menor caminho dirigido de i para j.

  $diâmetro(G) = max_{i,\, j \in G}\{distância(i,j)\}$

- Parte dinâmica:

  M conjunto de mensagens

  $M^+ = M \cup \{null\}$

  A cada nó i temos uma máquina de estados definida pela seguinte tupla:

  $P_i = (S_i, I_i, msg_i, trans_i)$, onde

  S_i = conjunto de estados do processo

  I_i = conjunto de estados iniciais

  $msg_i = S_i \times OUTN_I \rightarrow M \cup \{null\}$

  $VM_i = M^+ \times M^+ \times ... \; M^+ \times M^+$

  $trans_i = S_i \times VM_I \rightarrow S_i$

- Associado a cada aresta $(i,j) \in G$ tenho um canal (*link*) de comunicação tal que:

  1. $L[i,j]$ pode conter um elemento de M^+

  2. P_i pode adicionar uma mensagem, P_j pode retirar a mensagem.

A Execução

- Para todo P_i, ela começa em algum $s \in P_i$.
- A partir daí, todo processo P_i executa de forma sincronizada com os demais.
- Em uma rodada:
  1. Aplicar a função msg_i ao estado corrente e escrita das mensagens nos respectivos canais.
  2. Aplicar a função $trans_i$ aoestado corrente e ao vetor VM_i para obter o novo estado e remoção das mensagens do canal correspondente.

Término do Algoritmo

- É necessário definir-se estados de parada H_i:

  a) $msg_i(h,j)=null$, $\forall\ h \in H_i, j \in OUTN_i$

  b) $trans_i(h,vm)=h'$, onde $h,h' \in H_i, vm \in VM_i$