Programação Distribuída

Parte II

Paradigmas para
interação entre processos

1. Comunicação entre processos

- Introdução
  - Tecnologias: troca de mensagens (MPI), RPC, RMI, CORBA, Web Services, etc …
  - Problemas: podem ser divididos em classes
  - Resolução: podem seguir modelos de algoritmos
  - Paralelo x Distribuído
    - Paralelo: ambiente dedicado → menor ocorrência (!) de falhas de nós, atrasos e perdas de mensagens são raros
    - Distribuído: heterogeneidade (nós, SOs, rede), falhas, tempo de comunicação muito maior, atrasos e erros na entrega de mensagens

1. Comunicação entre processos

- Paradigmas de base
  - Mais genéricos: adaptáveis a vários problemas
  - Por serem genéricos são superados por paradigmas mais específicos adaptados a determinados problemas.
  - São eles:
    - Cliente / Servidor
    - Produtor / Consumidor
    - Interacting Peers

2. Cliente / Servidor

- Modelo mais difundido
- Cliente: requisita serviço e espera resposta
- Servidor: espera requisições, processa e responde
- Comunicação bi-direcional
- Tipicamente: um servidor, múltiplos clientes
- Servidor pode ser concorrente: threads
- Aplicações: sistemas de arquivos, banco de dados, gerentes de memória, escalonadores de disco e problemas clássicos (e.g., jantar dos filósofos)

3. Produtor / Consumidor

- Produtor: calcula e gera resultados
- Consumidor: lê resultados e calcula
- Mais interessante: seqüência de produtores / consumidores (alguns processos realizando as duas funções)
- Exemplo: problema do buffer limitado
  - Ciclos de produção e consumo do buffer
  - Novas inserções podem depender de retiradas do buffer
  - Vários consumidores e vários produtores para um único buffer → semáforos e monitores para controlar acesso

4. Interacting peers

- Todos processos executam o mesmo algoritmo
- Dados estão distribuídos: cada processo tem o seu
- Comunicam-se com outros processos para processar suas partes do trabalho
- Comunicação
  - pode ser somente com processos vizinhos (por exemplo)
  - com todos os processos ativos (simétrico)
  - Outros padrões: árvore, anel, centralizado.
- Exemplo: programas SPMD

6. Paradigmas específicos

- São combinações dos anteriores
- Atendem problemas mais especifícos
- São eles:
  - Manager / workers
  - Heartbeat algorithms
  - Pipeline algorithms

  Computação paralela
  - Probes and Echoes
  - Broadcast algorithms
  - Token-passing algorithms
  - Replicated server processes

  Computação distribuída

7. Paradigmas Comp. Paralela

- Manager / workers
  - Gerente/trabalhadores, mestre/escravo, etc …
  - Versão para memória distribuída do Bag-of-tasks
  - Gerente implementa o saco de trabalho, atribui tarefas, coleta resultados e detecta término.
  - Processos tabahadores compartilham "saco" de tarefas independentes.
  - Pode haver retro-alimentação de dados com os novos resultados gerados
  - Fácil de variar o número de trabalhadores e fácil de equilibrar a carga (tarefa do gerente)

7. Paradigmas Comp. Paralela

- Heartbeat algorithms
  - Fases paralelas
  - Útil para aplicações iterativas onde é possível processar partes do conjunto de dados em paralelo
  - Dados da nova iteração dependem dos resultados da iteração anterior
  - Forte sincronização (todos enviam e recebem)
  - Tipicamente, cada processo :
    - expande – envio das informações
    - contrai – coleta novos dados
    - processa e volta re-inicia o procedimento
  - Usos: algoritmos para processamento de imagens (region growing) e o Jogo da Vida (cellular automata)

7. Paradigmas Comp. Paralela

- Pipeline algorithms
  - Seqüência de produtores / consumidores
  - Saída de um processo é usada com entrada de outro
  - Processos trabalham sobre diferentes dados em paralelo
  - Tipicamente, três variações :
    - Aberto: fonte e destino não são especificados
    - Fechado: fonte e destino são direcionados a um coordenador
    - Circular: Destino redirecionado para fonte
  - Usos: multiplicação de matrizes

8. Paradigmas Comp. Distribuída

- Probe/Echo Algorithms
  - Adaptados para buscas em árvores e grafos
  - Em computação distribuída, muitas estruturas podem ser representadas por um grafo onde :
    - os processos são nós;
    - as arestas são os canais de comunicação.
  - Aplicações: Web searchers, banco de dados, jogos distribuídos, sistemas especialistas
  - Probes: mensagem enviada ao sucessor
  - Echoes: resposta subsequente
  - Probes são enviados em paralelo para todos os sucessores
  - Implementação: Descobrindo a topologia de uma rede

8. Paradigmas Comp. Distribuída

- Broadcast Algorithms
  - Redes menores (LANs): todos nós podem estar diretamente interligados → primitiva especial → broadcast
  - Corresponde ao envio concorrente da mesma mensagem a todos os nós (que a recebem através de uma primitiva receive normal).
  - Envio ordenado, recebimento não necessariamente
  - Técnica usada para disseminar informação através de uma rede de pequeno porte
  - Aplicações: útil para diversos problemas de sincronização distribuída (e.g., semáforos distribuídos)
  - Normalmente necessitam de ordenação total dos eventos de comunicação → Relógios lógicos (Lamport)
  - Implementação: Semáforos distribuídos

8. Paradigmas Comp. Distribuída

- Token-passing Algorithms
  - Baseados na utilização de uma mensagem com funcionalidade especial → token.
  - Configuração (sequencia) de processos conhecida (e.g., anel)
  - Usada para:
    - Obter algum tipo de permissão por consenso
    - Juntar informação global do estado de processos
  - Aplicações: garantia de acesso a Seção Critica, detecção de terminação de computação, detecção de deadlocks.
  - Implementação: Exclusão mútua distribuída

8. Paradigmas Comp. Distribuída

- Replicated Servers (Servidores replicados)
  - Servidor: gerenciador de algum recurso
  - Podem ser replicados:
    - Quando há várias instâncias do mesmo recurso (cada uma gerenciada por um servidor)
    - Por razões de desempenho: dar a impressão de existir um só recurso quando existem vários(e.g., arquivos replicados).
    - Tolerância a falhas
  - Pode ser necessário:
    - Manutenção de coerência global
    - Algoritmos de eleição
  - Implementação: Jantar dos filósofos distribuído