MAC0438 - Programação Concorrente

Daniel Macêdo Batista

IME - USP, 1 de Março de 2013

Roteiro

Lembretes

Introdução

Entendendo o hardware

Sintaxe

Paralelizando um algoritmo

Lembretes

O que eu estou devendo

- ☐ Até a aula de terça-feira
- ☐ Enunciados dos EPs
- ☐ Apresentar o monitor (Tentar. Não depende de mim)

O que vocês estão devendo

- Até a aula de terça-feira
- Confirmar as datas
- Por enquanto:
  - Provas: 26/04 e 28/06
  - Sub e Rec: 05/07 e 12/07
  - EPs: 01/04, 06/05 e 17/06

Introdução

Por que estudar programação concorrente?

- Imagine diversos carros querendo ir de um ponto A para um ponto B...

Soluções

- Mesma faixa competindo pelo espaço
- Diversas faixas paralelas
- Diversas rotas paralelas (diferentes caminhos)

Na disciplina

- Carros = Processos (ou threads como veremos adiante)
- Mesma faixa competindo pelo espaço = único processador
- Diversas faixas = Múltiplos processadores
- Diversas rotas = Processadores distribuídos
- É necessário ter sincronia e respeitar os sinais

O que é um programa concorrente?

- Um programa que contém dois ou mais processos que trabalham juntos para realizar uma tarefa
- Como os programas conseguem trabalhar juntos? Precisam se comunicar!

Comunicação

- Variáveis compartilhadas
- Troca de mensagens

Sincronização

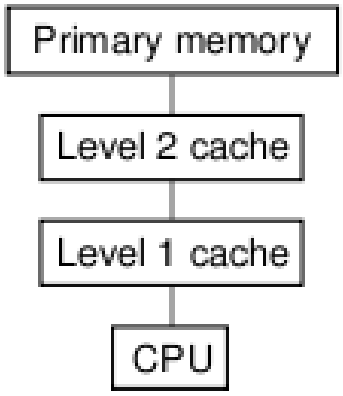
- Necessária para manter a "cooperação"
  - Exclusão mútua: seções críticas não são executadas ao mesmo tempo
  - Sincronização por condição: um processo é atrasado até que uma condição seja verdade
  - Exemplo do produtor/consumidor

Entendendo o hardware

Avanços no hardware

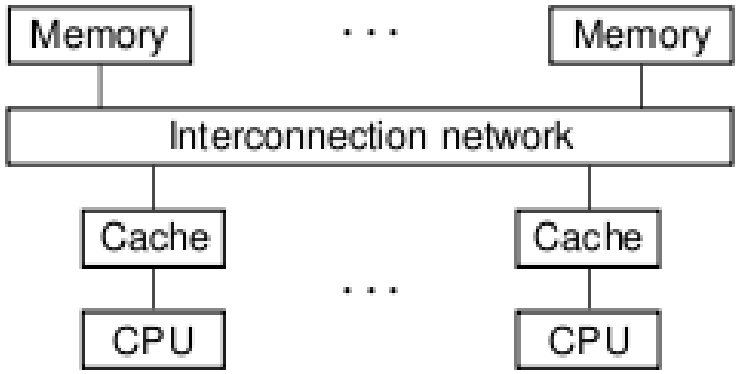
- Surgimento da interrupção
  - Mesmo com um único processador, um processo pode ser interrompido
- Surgimento das máquinas multiprocessadas
  - Como coordenar processos rodando (e modificando variáveis) em processadores diferentes?
- Temos que nos preocupar com os dois casos pois ambos afetam o acesso a variáveis

Processadores e caches



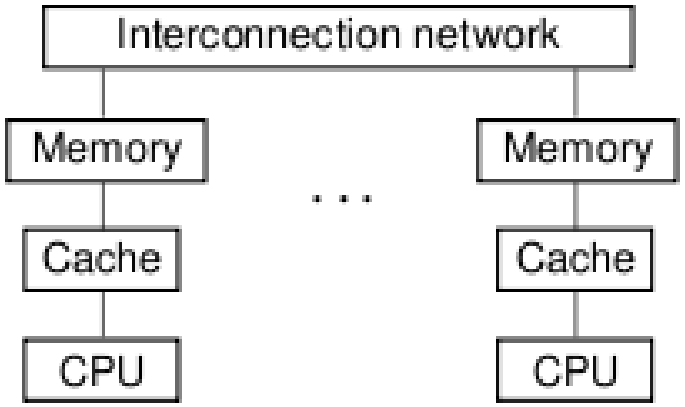
- Por que ter cache?
  - Localidade temporal (loops)
  - Localidade espacial (sequências do código)
  - Linha de cache
- Mas pode não estar no cache (hit e miss)
- E na escrita?

Memória compartilhada



- Cada processador tem seu cache
- Mais comum ter variáveis compartilhadas
  - Problema de consistência de cache (invalidar ou atualizar? "Sniffer"?)
  - Problema de falso compartilhamento (ocupar mais espaço?)

Memória distribuída



- Cada processador tem sua memória e cache (evita problemas de inconsistência)
- Mais comum ter troca de mensagens

Sintaxe

Escrevendo algoritmos concorrentes

- `co`
- `process`

```
for [i=1 to n] {
   write(i);
}
```

o algoritmo acima gera quantas saídas possíveis?

process e co


```
process iteracao[i=1 to n] {
   write(i);
}
comando # Pode ser executado antes dos processos
        # terminarem
```

```
co [i=1 to n] {
   write(i);
}
comando # Soh vai ser executado depois que os
        # processos terminarem
```

os algoritmos acima geram quantas saídas possíveis? n!

co


```
co comando1
   comando2
// comando3
   comando4
oc
   comando5
```

- comando1 e comando2 são um processo
- comando3 e comando4 são outro processo
- comando5 só executa depois que comando1 a comando4 terminarem

Paralelizando um algoritmo

Buscando padrões em um arquivo

- grep no unix
- Como seria o algoritmo de uma versão com um único processo?

Buscando padrões em um arquivo


```
string linha;
leia uma linha de stdin e salve em linha;
while (!EOF) {
   busque por padrao em linha;
   if (padrao em linha)
      write linha;
   leia a proxima linha de stdin e salve em linha;
}
```

- É possível paralelizar?

Quando paralelizar

- Para paralelizar, é necessário que hajam partes independentes
- Duas partes de um programa são independentes se o conjunto de escrita de cada parte é disjunto do conjunto de leitura e do conjunto de escrita da outra parte

Buscando padrões em um arquivo


```
string linha;
leia uma linha de stdin e salve em linha;
while (!EOF) {
   busque por padrao em linha;
   if (padrao em linha)
      write linha;
   leia a proxima linha de stdin e salve em linha;
}
```

- É possível paralelizar?

Buscando padrões em um arquivo

- É possível imprimir a linha antes de buscá-la?
- É possível buscar o padrão e ler a próxima linha?

Buscando padrões em um arquivo em // (Tentativa 1)

- Buscando o padrão e lendo a próxima linha em paralelo

```
string linha;
leia uma linha de stdin e salve em linha;
while (!EOF) {
   co busque por padrao em linha;
      if (padrao em linha)
         write linha;
   // leia a proxima linha de stdin e salve em linha;
   oc;
  }
```

- Está correto? As partes paralelas são independentes?
- Ideias?

Buscando padrões em um arquivo em // (Tentativa 2)

- Utilizando variáveis diferentes

```
string linha1,linha2;
leia uma linha de stdin e salve em linha1;
while (!EOF) {
   co busque por padrao em linha1;
      if (padrao em linha1)
         write linha1;
   // leia a proxima linha de stdin e salve em linha2;
   oc;
}
```

- As partes paralelas são independentes?
- O programa está correto?
- Ideias?

Buscando padrões em um arquivo em // (Tentativa 3)

- Atualizando os conteúdos das variáveis

```
string linha1,linha2;
leia uma linha de stdin e salve em linha1;
while (!EOF) {
   co busque por padrao em linha1;
      if (padrao em linha1)
         write linha1;
   // leia a proxima linha de stdin e salve em linha2;
   oc;
   linha1=linha2;
}
```

- As partes paralelas são independentes?
- O programa está correto?
- Quantos processos serão criados?

Eficiência no exemplo anterior

- `linha1=linha2`. Ideias?
- `co` dentro de um laço. Ideias?