

Programação Concorrente e Paralela

Noemi Rodriguez

2013



- princípios e técnicas de programação concorrente
 - multiprocessadores
 - memória compartilhada
 - troca de mensagens

obs: diferentes níveis de abstração!

que princípios e técnicas são esses?

- notações para controle de fluxo
 - criação de processos e threads, ou outras formas de expressão de paralelismo
- abstrações de comunicação
- sincronização em sistemas de memória compartilhada
 - ... mas problemas também existem sem memória compartilhada...
- propriedades: *safety* e *liveness*
- avaliação de desempenho
- balanceamento de carga



- M. Herlihy, N. Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2008.
- G. Andrews. *Foundations of Multithreaded, Parallel, and Distributed Programming*. Addison-Wesley, 2000.
- M. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill, 2003.
- P. Pacheco. *An Introduction to Parallel Programming*. McGraw-Hill, 2011.
- artigos técnicos.

- linhas de controle independentes ativas simultaneamente
 - o que é “atividade”?
 - um processador X múltiplos processadores
 - estudo nasceu com sistemas operacionais
 - importância hoje: popularidade de sistemas multiprocessadores
- disputa por recursos!

- aplicações que conceitualmente poderiam ser executadas de forma sequencial
- execução realmente simultânea
- paralelismo para melhoria de desempenho (ou viabilidade de execução)
 - lei de Amdahl

paralelismo e concorrência

- em ambos os casos necessidade de *coordenação* entre atividades
- cooperação x competição

- SISD
 - single instruction single data
- SIMD
 - single instruction multiple data
 - extensão para SPMD
- MISD
 - multiple instruction single data
- MIMD
 - multiple instruction multiple data

- processos, threads preemptivos, threads cooperativos
 - pilha de execução
 - dados globais?
 - espaços de endereçamento protegidos?
 - preempção?

Fluxos Independentes



globais
arquivos abertos
...

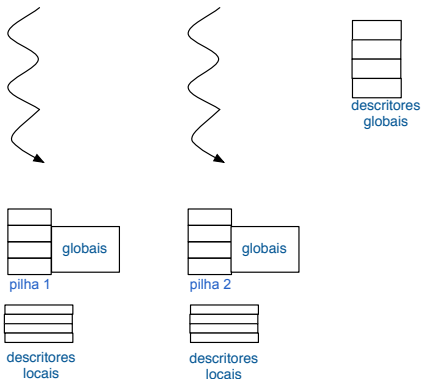


pilha 1



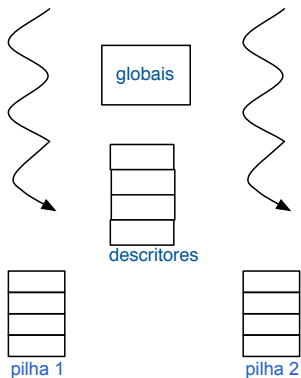
pilha 2

- pilhas independentes?
- acesso a globais?



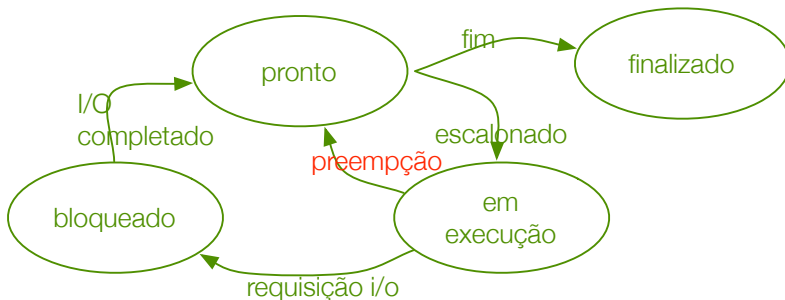
- pilhas independentes
- globais independentes
- descritores podem ser compartilhados
- preempção

Threads Preemptivas

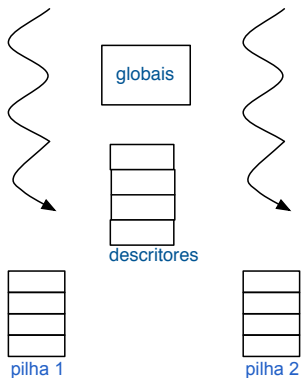


- pilhas independentes
- globais compartilhadas
- descritores compartilhados
- **PREEMPÇÃO**

Preempção



Threads Cooperativas



- pilhas independentes
- globais compartilhadas
- descritores compartilhados
- **SEM PREEMPÇÃO**

Concorrência: modelo mais comum

threads preemptivos com memória compartilhada

Exemplo: Busca de primos com memória compartilhada

- imprimir primos entre 1 e 10^{10}
- máquina de 10 processadores
- um thread por processador



Busca de Primos com Memória Compartilhada

- idéia 1: cada thread testa um intervalo pré-definido
- cada thread executa:

```
void primePrint {  
    int i = ThreadID.get(); // IDs in {0..9}  
    for (j = i*109+1, j<(i+1)*109; j++) {  
        if (isPrime(j))  
            print(j);  
    }  
}
```

- nem todos os intervalos têm números iguais de primos
- números grandes: maior dificuldade computacional
- desbalanceamento



Busca de Primos com Memória Compartilhada

- idéia 2: contador compartilhado
- cada thread executa:

```
int counter = new Counter(1);

void primePrint {
    long j = 0;
    while (j < 1010) {
        j = counter.getAndIncrement();
        if (isPrime(j))
            print(j);
    }
}
```



Contador Compartilhado

- o objeto `counter` é único para todos os threads

```
public class Counter {  
    private long value;  
  
    public long getAndIncrement() {  
        return value++;  
    }  
}
```

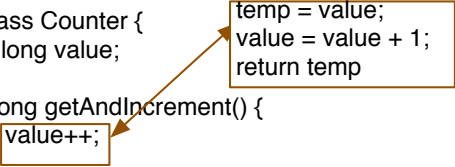


Contador Compartilhado

- o objeto `counter` é único para todos os threads
- e o problema é que podemos ter:

```
public class Counter {  
    private long value;  
  
    public long getAndIncrement() {  
        return value++;  
    }  
}
```

```
temp = value;  
value = value + 1;  
return temp
```



Memória Compartilhada e Preempção

- muitos entrelaçamentos são possíveis...

thread 1:

```
temp = value
```

```
temp = temp + 1
```

```
value = temp
```

thread 2:

```
temp = value
```

```
temp = temp + 1
```

```
value = temp
```

que mundo é esse em que value++ não tem o efeito que esperamos...?



- e mais do que isso...
- não basta pensar nos entrelaçamentos possíveis!

You don't know Jack about shared variables or memory models. Hans-J. Boehm and Sarita V. Adve. 2012. *Commun. ACM* 55, 2 (February 2012), 48-54.

Memória Compartilhada e Preempção

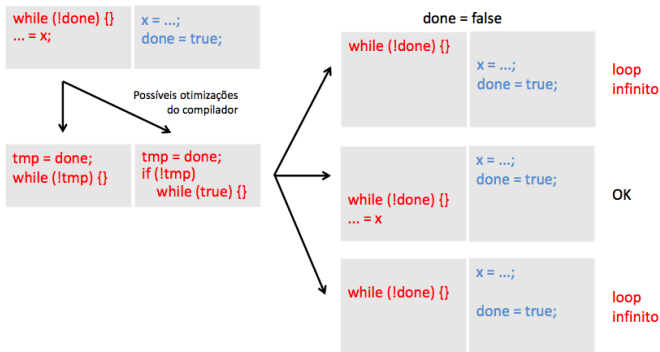
- imagine que o programa usa valores maiores do que a palavra do hardware
- `x++` traduzido para algo como:

```
tmp_hi = x_hi;  
tmp_lo = x_lo;  
(tmp_hi, tmp_lo)++;  
x_hi = tmp_hi;  
x_lo = tmp_lo;
```



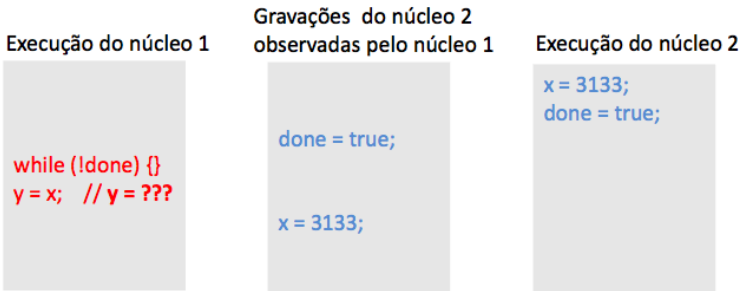
Memória Compartilhada e Preempção

- otimizações do compilador podem causar efeitos inesperados



Memória Compartilhada e Preempção

- caches podem ser atualiados em ordens diferentes



Concorrência: programador sempre terá que lidar com dificuldades de programação?

mecanismos e técnicas

- mecanismos para lidar com memória compartilhada
 - semáforos e locks
 - monitores
 - mecanismos não bloqueantes
 - memória transacional
- eliminar memória compartilhada
 - troca de mensagens e outras abstrações
- utilizar multithreading cooperativo
- soluções mais recentes
 - propriedades garantidas pelo compilador
 - D, DPJ, ...



Paralelismo: questões

- como paralelizar a solução de um problema?
 - objetivo maior normalmente é obter menor tempo de execução
 - desenho da solução paralela normalmente é dependente da plataforma de execução
- que plataformas facilitam o desenvolvimento e depuração?
- como medir o tempo de execução e comparar soluções?
- tolerância a falhas: como garantir que o trabalho feito até o momento da falha não seja perdido?



Inicialmente: Exclusão Mútua

- como criar mecanismos que garantam acesso exclusivo aos dados em ambientes de memória compartilhada “livremente” ...?
- slides livro Herlihy
- propriedades de *safety* e *liveness*