Sumário

- O que é a memória partilhada.
- Coerência forte.
- Concretização.

Objectivo

- Tornar um conjunto de memórias físicamente não partilhadas.

  Apenas interligadas por um meio de comunicação.
- Numa "única" memória partilhada.
- Se possível, mantendo a mesma aparência de uma única memória física.

Diferenças para o modelo cliente-servidor.

- Por vezes um servidor fornece serviços que só estão acessíveis num dado nó:

  Acede a maquinaria específica.
- O algoritmo não é conhecido pelo cliente.
- Segurança: acesso à memória só pode ser feito por um determinado programa.
- No modelo de memória partilhada, confia-se nos nós que acedem à memória.

Modelos de coerência

- O que esperamos intutivamente de uma memória?
- Esperamos um certo comportamento, por exemplo:

  Se escrevemos um valor numa variável e de seguida lemos essa variável (sem que ninguém tenha re-escrito), esperamos ler esse valor.
- Um modelo de coerência de memória define que comportamento podemos esperar de um sistema de memória.

  Em termos de operações de leitura e escrita.

Coerência atómica

- Modelo intuitivo de uma memória centralizada.
- Coerência atómica:

  Uma leitura retorna sempre o último valor escrito.

  Parte do princípio que existe uma referência de tempo global.

Memória sequencial

- O resultado de qualquer execução é equivalente ao de uma execução em que:

  a) todas as operações são executadas por uma ordem sequencial (arbitrária)

  b) as operações de cada processo aparecem nesta sequência pela ordem em que foram especificadas.
- Só se distingue da coerência atómica se houver acesso ao tempo real.

Exemplo

```
 P1               P2
a := 1;          b := 1;
print (b);       print (a);
```

- Existem diversas ordens sequênciais para executar estas acções:

  (1) $W_1(a)1\ R_1(b)0\ W_2(b)1\ R_2(a)1$

  (2) $W_1(a)1\ W_2(b)1\ R_1(b)1\ R_2(a)1$

  (3) $W_1(a)1\ W_2(b)1\ R_2(a)1\ R_1(b)1$

  (4) $W_2(b)1\ W_1(a)1\ R_1(b)1\ R_2(a)1$

  (5) $W_2(b)1\ W_1(a)1\ R_2(a)1\ R_1(b)1$

  (6) $W_2(b)1\ R_2(a)0\ W_1(a)1\ R_1(b)1$

História válida

- Considere-se a seguinte história:

$$H_1 : W_1(a)1\ R_1(b)0$$

$$H_2 : W_2(b)1\ R_2(a)1$$

Trata-se de uma história válida porque produz os mesmos resultados do que (1).

História inválida

- Considere-se a seguinte história:

$$H_1 : W_1(a)1\ R_1(b)0$$

$$H_2 : W_2(b)1\ R_2(a)0$$

Trata-se de uma história inválida não existe nenhuma sequência que produza estes resultados.

Requesitos para concretização

- Para concretizar uma memória partilhada é necessário:

  Interceptar o acesso à memória.

  Em cada acesso, executar um algoritmo de gestão da memória que garanta o modelo de coerência.

- Para já vamos supor que se consegue interceptar o acesso à memória.

Concretização usando RPC

- A memória reside fisicamente num único nó.
- Sempre que um acesso é interceptado, este é transformado num RPC para esse nó.
- Desvantagens:

  Esse nó fica sobrecarregado.

  Os outros nós detectam um atraso no acesso à memória.

Exemplo de concretização (2)

- Uma maneira (um pouco) mais eficiente de concretizar uma memória distribuída.

  Existe apenas uma cópia da memória partilhada.

  Em cada momento, a cópia encontra-se na memória física de um dado processo designado por o dono da cópia.

  Sempre que alguém quer ler ou escrever é pedida (automáticamente) a transferência do estado para a sua memória local.

  A memória virtual é transferida de memória em memória de modo transparente.

Optimização

- No algoritmo anterior, a cada acesso vindo de um nó diferente, o estado de toda a memória tem de ser transferido.
- Destingem-se operações de escrita e leitura.

  Podem haver várias cópias válidas no sistema, desde que se esteja apenas a ler a memória (não há alteração de estado).

- Quando se escreve:

  Actualizam-se todas as cópias válidas.

  Ou invalidam-se as cópias desactualizadas.

Invalidação/ actualização

- Invalidação:

  Invalidar todas as cópias da página.

  Mudar o modo de acesso da página para escrita.

  Obter uma cópia da página.

  Escrever na página.

Invalidação/ actualização

- Actualização em sistemas paginados:

  Escrever em todas as cópias em cada acesso.

  Impraticável, por ser muito dispendioso.

  Pode ser interessante, se o acesso à memória estiver limitado por instruções de sincronização.

Fragmentação da memória

- Considere-se uma zona de memória com duas posições de memória x e y, ambas com um valor inicial 0.

  Considere-se que o processo P1 executa uma operação de escrita em x, $W_1(x)1$.

  Considere-se que posteriormente o processo P2 executa uma operação de leitura em y, $R_2(y)0$.

- Será que o processo $P2$ necessita de obter uma nova cópia da memória?

  Não porque estão a aceder a variáveis diferentes.

Fragmentação da memória

- Do ponto de vista de evitar transferências desnecessárias, os mecanismos de gestão de memória deveriam considerar cada variável individualmente.
- Depende do modo como é feita a intercepção no acesso à memória.
- Pode haver conjunto de variáveis que são acedidas em conjunto.

  Neste caso é mais eficaz transferir tudo de uma vez (em vez de cada variável, uma a uma).

Exemplo de concretização (4)

- Uma maneira de concretizar uma memória distribuída estruturada em páginas.

  Mantém-se uma lista de nós que têm uma cópia actualizada de cada página.

  Quando se pretende ler, pede-se uma cópia a um destes nós.

  Quando se quer escrever, invalidam-se as restante cópias.

Exemplo de concretização (4)

- Problemas a resolver:

  É necessária manter o mapa de nós que possuem cópias para todas as páginas. Esta informação pode ser mais volumosa que a própria página se a página for muito pequena.

  É necessário sincronizar acessos concorrentes.

Técnicas de particionamento da memória

- Espaço linear, dividido em páginas de igual tamanho:

  Intercepção feita por hardware.

- Conjunto de variáveis ou conjunto de objectos.

  Intercepção pode ser feita por software desde que para aceder à variavél seja necessário chamar uma função.

  Por vezes o acesso está limitado por um par de funções (que podem estar associadas a primitivas de sincronização).

 mente

Diferentes técnicas (YVY [Li & Hudak])

- É necessário manter algures e para cada página:

  O modo de acesso (escrita/ leitura).

  A lista de nós que têm cópias da página.

  Uma lista de processos que pretendem aceder à página.

- Esta informação pode estar sempre no mesmo nó:

  Solução centralizada.

- Ou estar distribuída por vários nós.

Distribuição das tabelas de página

- Centralizada:

  Um nó fica muito sobrecarregado.

- Distribuída fixa:

  A informação respeitante a uma dad página reside sempre no mesmo nó, mas cada nó guarda informação de páginas diferentes.

- Distribuída dinâmica:

  Todos os nós sabem que foi o último nó a ter acesso para escrita.

  Esse nó sabe a quem deu cópias para leitura.

Granularidade e falsa partilha

- Quando menor for a unidade de gestão de memória:

  Menos se verifíca a falsa-partilha.

  Menor limitação à concorrência do sistema.

  Maior o custo relativo da gestão de memória.

- O desempenho depende da localidade do código:

  Varia de aplicação para aplicação.

Resumo

- Memória atómica.
- Concretização.
- Desempenho.