Sistemas Operacionais

Aula 15 – Deadlock – Parte 02

Professor: Wellington Franco

Introdução

- Em ambientes de multiprogramação, vários processos podem competir por uma quantidade finita de recursos.

- Um processo solicita recursos, se os recursos não estão disponíveis naquele momento, o processo entra em estado de espera.

- Em alguns casos, um processo em espera não consegue mudar novamente de estado, porque os recursos que ele solicitou estão reservados para outros processos em espera.

Introdução

- Em ambientes de multiprogramação, vários processos podem competir por uma quantidade finita de recursos.
- Um processo solicita recursos, se os recursos não estão disponíveis naquele momento, o processo entra em estado de espera.
- Em alguns casos, um processo em espera não consegue mudar novamente de estado, porque os recursos que ele solicitou estão reservados para outros processos em espera.
- **DEADLOCK**

Introdução

Algoritmo do banqueiro

- Múltiplas instâncias
- Cada processo deve declarar a quantidade máxima de instâncias de cada tipo de recurso que ele pode precisar
- Quando um processo solicita recursos, o sistema deve determinar se a alocação desses recursos deixará o sistema em um estado seguro
- Se deixar, os recursos são alocados, caso contrário, o processo deve esperar até que algum outro processo libere recursos suficientes.

Algoritmo do banqueiro

- Deve-se utilizar estruturas de dados para implementação do algoritmo.
    - **n**: número de processos
    - **m**: número de tipos de recursos
- **Disponível**
    - Vetor de comprimento **m**.
    - Indica a quantidade de recursos disponíveis.
    - Se **Disponível[j] = k**, existem k instâncias do recurso R_j disponíveis

Algoritmo do banqueiro

- **Max**
  - Matriz n x m.
  - Define a demanda máxima de cada processo.
  - Se **Max[i,j] = k**, então o P_i pode solicitar, no máximo, k instâncias do recurso R_j

- **Alocação**
  - Matriz n x m.
  - Define a quantidade de recursos alocadas correntemente para cada processo.
  - Se **Alocação[i,j] = k** então o processo P_i está atualmente em posse de k instâncias do recursos R_j

Algoritmo do banqueiro

- **Necessidade**
  - Matriz n x m.
  - Indica os recursos remanescentes necessarios para cada processo.
  - Se **Necessidade[i,j] = k**, então o processo P_i pode precisar de mais k instâncias do recurso R_j para poder completar sua tarefa
  - **Necessidade [i,j] = Max[i,j] – Alocação [i,j]**

Algoritmo de segurança

1. Sejam *Trabalho* e *Término* vetores de tamanho m e n, respectivamente. Inicialize $Trabalho = Disponível$ e $Término[i] = false$ para $i = 0, 1, \ldots, n - 1$.
2. Encontre um índice i tal que
   a. $Término[i] == false$
   b. $Necessidade_i \leq Trabalho$

   Se não existir tal i, vá para o passo 4.
3. $Trabalho = Trabalho + Alocação_i$

   $Término[i] = true$

   Vá para o passo 2.
4. Se $Término[i] == true$ para todo i, então o sistema está em um estado seguro.

Algoritmo de solicitação de recursos

- Seja Solicitação$_i$ o vetor de solicitação do processo P_i
- Se **Solicitação[j] = k**, então o processo P_i deseja k instâncias do tipo do recurso R_j.

Algoritmo de solicitação de recursos

1. Se $Solicitação_i \leq Necessidade_i$, vá para o passo 2. Caso contrário, emita uma condição de erro, já que o processo excedeu sua necessidade máxima.
2. Se $Solicitação_i \leq Disponível$, vá para o passo 3. Caso contrário, P_i deve esperar, já que os recursos não estão disponíveis.
3. Faça o sistema simular que alocou os recursos solicitados ao processo P_i modificando o estado como descrito abaixo:

$$Disponível = Disponível - Solicitação_i;$$
$$Alocação_i = Alocação_i + Solicitação_i;$$
$$Necessidade_i = Necessidade_i - Solicitação_i$$

Se o estado de alocação de recursos resultante for seguro, a transação é concluída e o processo P_i recebe seus recursos.

Exemplo de execução do algoritmo do banqueiro

	Alocação	*Max*	*Disponível*
	A B C	*A B C*	*A B C*
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Exemplo de execução do algoritmo do banqueiro

- O estado atual é seguro?

- O que acontece se o sistema receber a seguinte sequência de solicitações?
  - P1 requisita (1,0,2)
  - P4 requisita (3,3,0)
  - P0 requisita (0,2,0)

Exemplo do uso do algoritmo do banqueiro

- 5 processos: P_0 a P_4
- 3 tipos de recursos: A, B e C
    - A: 10 instâncias
    - B: 5 instâncias
    - C: 7 instâncias

Exemplo do uso do algoritmo do banqueiro

Processo	Alocação			Max			Disponível		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	3	3	2
P1	2	0	0	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Exemplo do uso do algoritmo do banqueiro

- Matriz Necessidade?

Exemplo do uso do algoritmo do banqueiro

- O estado atual é seguro para a sequência <P1, P3, P4, P2, P0>?

Exemplo do uso do algoritmo do banqueiro

- O que acontece se o sistema receber a sequência de solicitações?
  P1 requisita (1,0,2) a mais

Exemplo do uso do algoritmo do banqueiro

- O que acontece se o sistema receber a sequência de solicitações?

  P1 requisita (1,0,2) a mais

  Solicitação <= Disponivel

  (1,0,2) <= (3,3,2)

Exemplo do uso do algoritmo do banqueiro

- O que acontece se o sistema receber a sequência de solicitações?
  P1 requisita (1,0,2) a mais

Processo	Alocação			Max			Disponível		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	2	3	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Exemplo do uso do algoritmo do banqueiro

- O estado atual é seguro para a sequência <P1, P3, P4, P0, P2>?

Exemplo do uso do algoritmo do banqueiro

- O que acontece se o sistema receber a sequência de solicitações?

  P4 requisita (3,3,0) a mais

  Solicitação <= Disponível

  (3,3,0) <= (2,3,0)

  Não pode ser atendida

Exemplo do uso do algoritmo do banqueiro

- O que acontece se o sistema receber a sequência de solicitações?

  P0 requisita (0,2,0) a mais

  Solicitação <= Disponivel

  (0,2,0) <= (2,3,0)

Exemplo do uso do algoritmo do banqueiro

- O que acontece se o sistema receber a sequência de solicitações?
  P0 requisita (0,2,0) a mais

Processo	Alocação			Max			Disponível		
	A	B	C	A	B	C	A	B	C
P0	0	3	0	7	5	3	2	1	0
P1	3	0	2	3	2	2			
P2	3	0	2	9	0	2			
P3	2	1	1	2	2	2			
P4	0	0	2	4	3	3			

Exemplo do uso do algoritmo do banqueiro

- O estado atual é seguro para a sequência <P1, P3, P4, P0, P2>?

Exemplo do uso do algoritmo do banqueiro

- O estado atual é seguro para a sequência <P1, P3, P4, P0, P2>?

  O estado resultante é inseguro

Detecção de deadlock

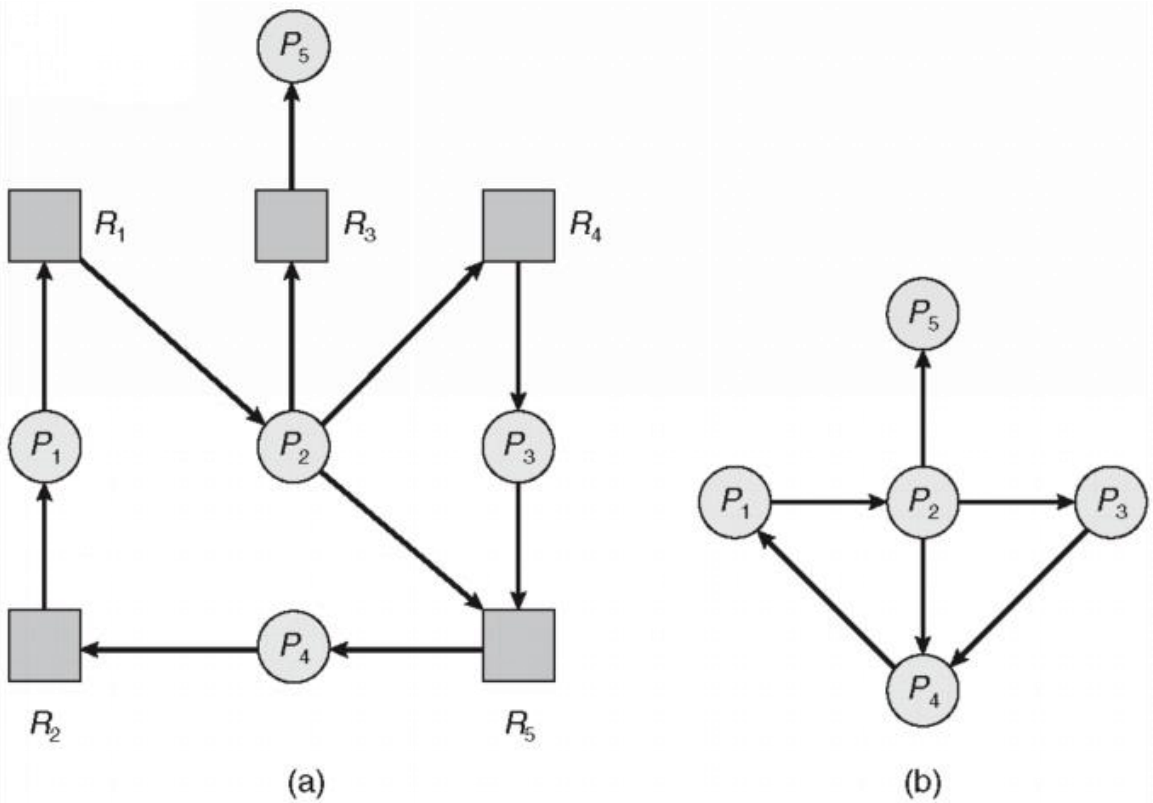
- Permite que o sistema entre em um estado de deadlock
  - Utiliza um algoritmo de detecção para identificar esta situação
  - Utiliza um esquema de recuperação para eliminar o deadlock

Uma única instância de cada tipo de recurso

- Mantêm um grafo de espera (variante do grafo de alocação de recursos)
  - Os nós são processos
  - Pi -> Pj, se Pi está esperando por Pj

- Um algoritmo de detecção de ciclos é executado periodicamente. Se existe um ciclo, existe um deadlock

Uma única instância de cada tipo de recurso



Várias instâncias de um tipo de recurso

- **Disponível**: um vetor de comprimento m indica a quantidade de instâncias disponíveis para cada tipo de recurso

- **Alocação**: uma matriz n x m define o número de instâncias de cada tipo de recurso alocadas para cada processo

- **Solicitação**: uma matriz n x m indicando as requisições realizadas por cada processo. Se **Solicitação[i,j] = k**, o processo Pi está solicitando mais k instâncias do recurso Rj.

Várias instâncias de um tipo de recurso

1. Sejam *Trabalho* e *Término* vetores de tamanho m e n, respectivamente. Inicialize $Trabalho = Disponível$. Para $i = 0, 1, \ldots, n - 1$, se $Alocação_i \neq 0$, então $Término[i] = false$; caso contrário, $Término[i] = true$.
2. Encontre um índice i tal que
   a. $Término[i] == false$
   b. $Solicitação_i \leq Trabalho$

   Se não existir tal i, vá para o passo 4.
3. $Trabalho = Trabalho + Alocação_i$

   $Término[i] = true$

   Vá para o passo 2.
4. Se $Término[i] == false$ para algum i, $0 \leq i < n$, então o sistema está em estado de deadlock. Além disso, se $Término[i] == false$, então o processo P_i está em deadlock.

Exemplo do algoritmo de detecção

- 5 processos: P_0 a P_4
- 3 tipos de recursos: A, B e C
    - A: 7 instâncias
    - B: 2 instâncias
    - C: 6 instâncias

Exemplo do algoritmo de detecção

Processo	Alocação			Solicitação			Disponível		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0	0	0	0
P1	2	0	0	2	0	2			
P2	3	0	3	0	0	0			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	3			

Exemplo do algoritmo de detecção

- Se executarmos o algoritmo para a sequência <P0, P2, P3, P1, P4>, temos ou não o **estado de deadlock**?

Exemplo do algoritmo de detecção

- Se executarmos o algoritmo para a sequência <P0, P2, P3, P1, P4>, temos ou não o **estado de deadlock**?

- Se P2 realizar a solicitação de uma instância do tipo C?

Exemplo do algoritmo de detecção

Processo	Alocação			Solicitação			Disponível		
	A	B	C	A	B	C	A	B	C
P0	0	1	0	0	0	0	0	0	0
P1	2	0	0	2	0	2			
P2	3	0	3	0	0	1			
P3	2	1	1	1	0	0			
P4	0	0	2	0	0	3			

Exemplo do algoritmo de detecção

- Se executarmos o algoritmo para a sequência <P0, P2, P3, P1, P4>, temos ou não o **estado de deadlock**?

Exemplo do algoritmo de detecção

- Se executarmos o algoritmo para a sequência <P0, P2, P3, P1, P4>, temos ou não o **estado de deadlock**?

  Existe um deadlock entre P1, P2, P3 e P4, apenas P0 é liberado

Uso do algoritmo de detecção

- Quando devemos invocar o algoritmo de detecção?
  - Com que frequência um deadlock pode ocorrer?
  - Quantos processos serão afetados pelo deadlock quando ele ocorrer?
- Se ocorrerem deadlocks com frequência, então o algoritmo de detecção deve ser invocado com frequência.
- Os deadlocks só ocorrem quando algum processo faz uma solicitação que não pode ser atendida imediatamente.

Uso do algoritmo de detecção

- Pode-se invocar o algoritmo sempre que uma solicitação for realizada. Assim, é possível identificar o processo que causou o deadlock.
    - Pode causa overhead considerável no tempo de processamento.
    - Se o algoritmo for executado arbitrariamente ele pode encontrar vários ciclos no grafo de recursos e pode não ser possível identificar qual dentre os processos em deadlock “causou” o problema.

Recuperação do deadlock

- Quando o algoritmo detectar que existe o deadlock, há várias alternativas disponíveis de solução:
  - Informar ao operador para que ele manualmente efetue alguma ação
  - Recuperação automática
    - Encerramento de processos
    - Preempção de recursos

Encerramento de processos

- Matar todos os processos em deadlock
- Matar um processo de cada vez até que o ciclo seja desfeito
- Em que ordem matar os processos?
    - Prioridade
    - Tempo de computação e tempo adicional necessário para conclusão
    - Recursos utilizados pelo processo
    - Recursos necessários para que o processo conclua sua execução
    - Quantidade de processos a ser finalizada
    - O processo é interativo ou de lote?

Preempção de recursos

- **Seleção de uma vítima**
  - Associado ao custo
- **Reversão**
  - Voltar para algum estado seguro
- **Inanição**
  - O mesmo processo pode ser sempre escolhido como vítima
  - Incluir o número de vezes que ocorreu a reversão no custo
  - Deve-se garantir que o processo só possa ser selecionado como vítima um número finito (pequeno) de vezes.

Dúvidas??

E-mail: wellington@crateus.ufc.br