

Deadlock

Indice

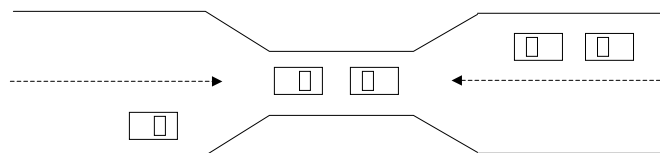
- Modello
- Caratterizzazione dei deadlock
- Metodi per gestire i deadlock
- Prevenire i deadlock
- Evitare che accadano i deadlock
- Rilevare la presenza di deadlock
- Recovery da una situazione di deadlock

Il problema del deadlock

- Un insieme di processi bloccati che hanno acquisito una risorsa ed attendono di acquisirne un'altra tenuta da un altro processo dello stesso insieme.
- Esempio 1:
 - Il sistema ha due unità nastro.
 - P1 e P2 hanno acquisito un nastro ciascuno e hanno bisogno di un secondo nastro.
- Esempio 2:
 - semafori A and B, inizializzati a 1

P0	P1
wait (A);	wait(B);
wait (B);	wait(A);

Esempio dell'attraversamento di un ponte

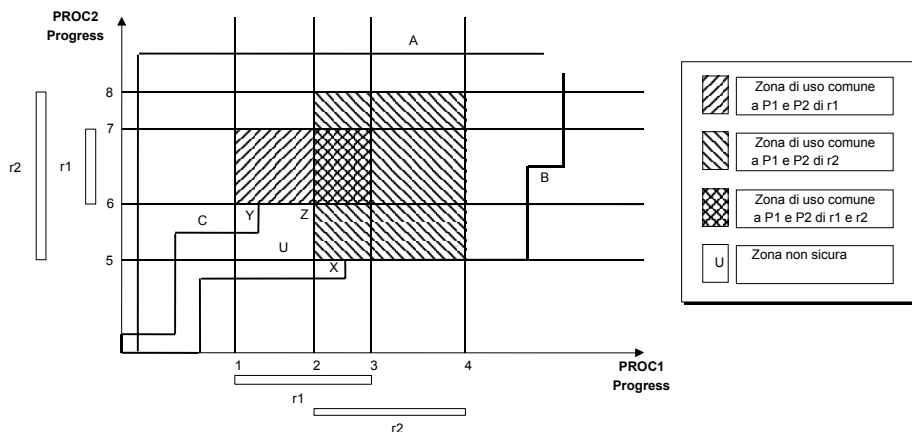


- Il traffico è a senso unico
- Si può considerare una metà del ponte come una risorsa
- Se avviene il deadlock, può essere risolto se una macchina torna indietro (rilascia la risorsa ed effettua un rollback)
 - Eventualmente deve tornare indietro più di una macchina

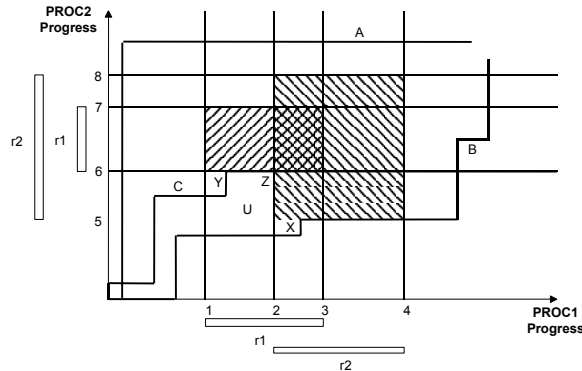
Starvation

- Si ha starvation quando uno più processi, non in blocco critico, attendono indefinitamente prima di usare una risorsa
- Dipendente dalla politica di assegnazione della risorsa
- Si può evitare con una politica FIFO

Progresso congiunto di due processi



Metodi di gestione del deadlock



- Prevenzione: proibire l'esistenza di uno stato non sicuro
- Avoidance : proibire l'entrata in uno stato non sicuro
- Recovery: proibire la residenza in uno stato non sicuro

Modello

- Tipi di risorse R_1, R_2, \dots, R_m
 - Cicli di CPU, spazio di memoria, periferiche
- Ogni risorsa di tipo R_i ha W_i istanze
- Ogni processo utilizza una risorsa con il protocollo:
 - richiesta
 - uso
 - rilascio

Caratterizzazione del deadlock

- Il deadlock può avvenire se si verificano contemporaneamente 4 condizioni:
 - Mutua esclusione: solo un processo alla volta può usare una risorsa.
 - Hold and wait: un processo che mantiene almeno una risorsa attende di acquisire altre risorse mantenute da altri processi.
 - No preemption: una risorsa può essere rilasciata volontariamente solo dal processo che la utilizza, dopo che ha completato il suo task.
 - Attesa circolare: esiste un insieme $\{P_1, \dots, P_n\}$ di processi in attesa di una risorsa:
 - P_1 attende una risorsa tenuta da P_2 , ..., P_{n-1} attende una risorsa tenuta da P_n e, P_n attende una risorsa tenuta da P_1 .

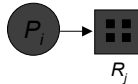
Grafo di allocazione delle risorse

- Grafo di allocazione delle risorse: un insieme di vertici V e un insieme di archi E
- V è partizionato in due tipi:
 - $P = \{P_1, P_2, \dots, P_n\}$, l'insieme dei processi nel sistema.
 - $R = \{R_1, R_2, \dots, R_m\}$, l'insieme dei tipi di risorse nel sistema.
- Arco di richiesta $P_i \rightarrow R_j$
- Arco di assegnazione $R_j \rightarrow P_i$

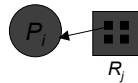
Grafo di allocazione delle risorse

- Processo 
- Tipo di risorsa con 4 istanze 

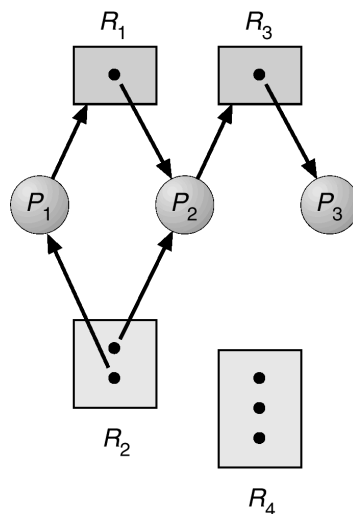
- P_i chiede un'istanza di R_j



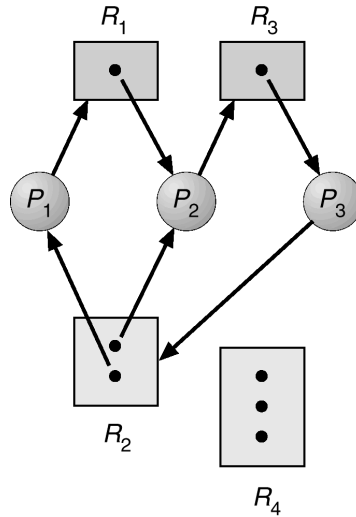
- P_i possiede un'istanza di R_j



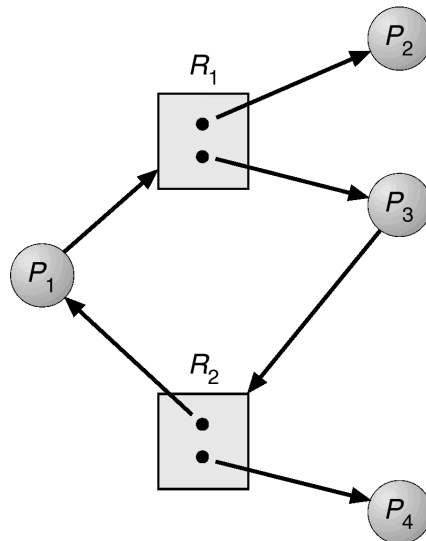
Esempio di grafo di allocazione delle risorse



Esempio di grafo di allocazione delle risorse



Grafo di allocazione con un ciclo ma senza deadlock



Cicli e deadlock

- Se un grafo non contiene cicli \Rightarrow
 - non c'è deadlock.
- Se un grafo contiene un ciclo \Rightarrow
 - se esiste solo un'istanza per tipo di risorsa, allora c'è sicuramente deadlock.
 - se esiste più di un'istanza per tipo di risorsa, allora c'è la possibilità del deadlock.

Metodi per gestire i deadlock

- Assicurare che il sistema non entri mai in uno stato di deadlock.
- Permettere che il sistema entri in uno stato di deadlock e poi effettuare un'operazione di recovery.
- Ignorare il problema e prevedere che la probabilità di un deadlock nel sistema sia bassissima; metodo utilizzato da molti sistemi, UNIX compreso.

Prevenire il deadlock

- Vincolare le richieste che possono essere effettuate
- “Violare” le condizioni necessarie per il deadlock.
- Mutua esclusione – non è richiesta per risorse che possono essere condivise.

Prevenzione del deadlock

- Hold and Wait – deve garantire che quando un processo chiede una risorsa, esso non mantenga altre risorse.
 - Request All First (RAF): i processi devono acquisire tutte le risorse di cui necessitano prima di iniziare l'attività di elaborazione;
 - Release Before Request (RBR): ai processi è consentito richiedere risorse solamente se non ne hanno già acquisite in precedenza. Prima di ogni nuova richiesta devono essere rilasciate le risorse già utilizzate.
- Svantaggi:
 - Scarso utilizzo delle risorse; possibilità di starvation.

Prevenzione del deadlock

- No Preemption
 - Se un processo che mantiene alcune risorse ne chiede un'altra che non può essere allocata immediatamente, allora è costretto a rilasciare tutte le risorse mantenute (preemption).
 - Le risorse liberate sono aggiunte alla lista delle risorse che il processo attende di acquisire.
 - Il processo sarà svegliato solo quando potrà riacquisire tutte le sue vecchie risorse e quelle nuove che richiede.

Prevenzione del deadlock

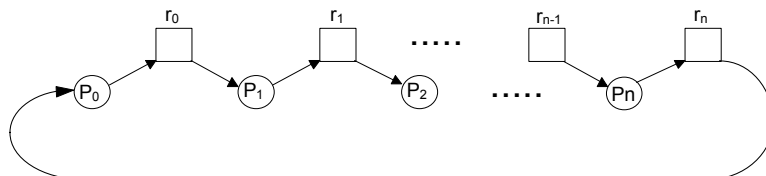
- Attesa Circolare – Hierarchical Resource Usage (HRU).
 - Impone una relazione di ordinamento totale tra i vari tipi di risorse, associando a ciascuno di essi un numero intero, ad esempio:
 - dischi = 1
 - lettori di nastri = 5
 - stampanti = 12
- forza ogni processo a richiedere le risorse con un ordine crescente di enumerazione.

HRU

- Un processo può richiedere una istanza della risorsa r_k solo se $F(r_k) > F(r_i)$, dove r_i è il tipo di risorsa richiesto prima di r_k .
- In caso contrario il processo deve rilasciare tutte le risorse di tipo r_h per cui $F(r_h) \leq F(r_k)$.

Hierarchical Resource Usage

- Per assurdo, supponiamo che vi sia un insieme di processi $P = \{ p_0, p_1, \dots, p_n \}$ in attesa circolare \Rightarrow
 $F(r_k) < F(r_{k+1}), \forall k = 0 \dots n-1$ cioè
 $F(r_0) < F(r_1) < \dots < F(r_n) < F(r_0)$
 $F(r_0) < F(r_0)$
che è assurdo.
- Dunque non può sussistere una condizione di attesa circolare



Evitare il deadlock

- Richiede che il sistema abbia alcune informazioni disponibili a priori.
- Il modello più semplice ed utile richiede che ciascun processo dichiari il massimo numero di risorse di ciascun tipo di cui ha bisogno.
- Algoritmo del banchiere esamina dinamicamente lo stato del grafo di allocazione delle risorse per assicurare che non si crei un ciclo, e quindi attesa circolare.
- Lo stato dell'allocazione delle risorse è definito dal numero di risorse disponibili ed allocate, e dal massimo numero di richieste dei processi.

Algoritmo per i grafi di allocazione

- Un arco di intenzione di richiesta $P_i \rightarrow R_j$ indica che il processo P_j può richiedere la risorsa R_j ; è rappresentato da una linea tratteggiata.
- Quando un processo richiede una risorsa, un arco di intenzione di richiesta diventa un arco di assegnazione.
- Quando un processo rilascia una risorsa, l'arco di assegnazione ritorna ad essere un arco di intenzione di richiesta
- Le risorse devono essere richieste a priori al sistema

Algoritmo del banchiere: stato

Cassa	2
P	4 (4)
Q	2 (1)
R	2 (7)

Risorse acquisite (ancora richieste)

Algoritmo del banchiere: stato sicuro

- Quando un processo chiede una risorsa disponibile, il sistema deve decidere se l'allocazione immediata lascerebbe il sistema in uno stato non sicuro.
- Il sistema è in uno stato sicuro se esiste una sequenza sicura di allocazione delle risorse per tutti i processi.
- Se il sistema è in uno stato sicuro \Rightarrow non c'è deadlock.
- Se il sistema non è in uno stato sicuro \Rightarrow possibilità di deadlock.
- Si assicura che il sistema non entrerà mai in uno stato non sicuro.

Algoritmo del banchiere

- Ogni processo deve dichiarare a priori il massimo uso di risorse
- Quando richiede una risorsa può essere bloccato.
- Quando ottiene le risorse che gli servono, garantisce che le restituirà in un tempo finito.
- Complessità $O(m \times n^2)$, ove m è il numero di risorse e n il numero di processi).

Esempio di stato sicuro

Cassa	2	Cassa	4	Cassa	8	Cassa	10
P	4 (4)	P	4 (4)	P	- (-)	P	- (-)
Q	2 (1)	Q	- (-)	Q	- (-)	Q	- (-)
R	2 (7)	R	2 (7)	R	2 (7)	R	- (-)

Esempio di stato non sicuro

Cassa	2	Cassa	1	Cassa	3
P	4 (4)	P	4 (4)	P	4 (4)
Q	2 (1)	Q	2 (1)	Q	- (-)
R	2 (7)	R	3 (6)	R	3 (6)

Rilevazione del deadlock

- Si permette al sistema di entrare in uno stato di deadlock.
- Ad intervalli di tempo fisso, o quando viene assegnata una risorsa, il sistema esegue un algoritmo di rilevazione del deadlock.
- Se esiste un ciclo si applica uno schema di recovery.

Recovery da deadlock: Preemption delle risorse

- Selezione di una vittima
 - minimizzando il costo.
- Rollback
 - far ritornare un processo ad uno stato sicuro
- Starvation
 - è possibile che lo stesso processo sia scelto come vittima più volte e quindi incorra in numerosi rollback.