Sis20241122

Andrea

22 Novembre 2024

Contents

1 ripasso deadlock

Ci sono 4 approcci per affrontare la questione deadlock: ovvero ignorare il problemama, riconoscere e risolverlo, evitarlo dinamicamente e assicurare che il sistema non possa entrare in deadlock.
Tra gli esempi algoritmi del banchiere, che fa in modo che il sistema rimanga sempre in stati sicuri, evitando di allocare risorse in caso questo possa creare stallo.

2 prevenzione del deadlock

Approccio che previene il deadlock negando almeno una delle quattro **condizioni di Coffman**. Mutua esclusione, hold and wait, attesa circolare e attesa di prerilascio

2.1 Mutua esclusione

Cade ipotesi di mutua esclusione. Si possono usare delle tecniche, tra cui **spooling**, tuttavia introduce della competizione per spazio su disco. Non è facile superare il problema della mutua esclusione su risorse che lo richiedono, generalmente i riesce soltanto a spostare il problema.

Una regola di buona programmazione è allocare risorse per il minor tempo possibile.

2.2 Hold and wait

COndizione che si riferisce al fatto che i processi possono richiedere risorse mentre ne detengono altre. Ed eventualmente essere messi in attesa per le altre risorse.
Come posso negarla?
Ad esempio, posso fare in modo che quando un processo richieda una risorsa non disponibile, rilasci tutte le risorse che detiene. In pratica il processo o detiene tutte le risorse o non le detiene. Processo non può mettersi in attesa finché possiede risorse, **metodo transazionale**. Comporta un basso utilizzo delle risorse e rischio starvation, ha un utilizzo limitato

2.3 Attesa circolare

In una situazione di stallo abbiamo una serie di processi $P_1, P_2, ..., P_n$ tali che ciascun processo sia in attes di una risorsa detenuta dal successivo.
Per risolvere, potrei imporre un ordinamento totale sulle classi di risorse, in modo che il processo richieda le risorse nell'ordine prestabilito. Ricordiamo l'esempio visto all'inizio in cui P1 richiede R1 e detiene R2, P2 richiede R2 e detiene R1.
Fattibile ma difficile da implementare, in quanto non è detto che l'ordine vada bene per tutte le risorse, poi se cambiano le risorse probabilmente devo sostituire l'ordinamento.

2.4 Attesa di prerilascio

Spesso il prerilascio è impedito dalla natura stessa delle risorse, quindi difficilmente applicabile, spesso impossibile.

2.5 Conclusioni

spesso i vari approcci non sono usati in modo esclusivo, ma in base alla categorie di risorse si sceglie approccio. Esempio nel mondo database si sceglie **protocollo in due passi**, in cui prima di iniziare la transazione si chiedono tutte le risorse necessarie, se non ha successo lascia e riprova. Non fattibile in sistemi real-time, in quanto non si può prevedere il tempo di attesa.