1- Descrivere l'allocazione dinamica della memoria di un sistema di archiviazione di massa. Identificare le tecniche più utilizzate, descrivendone i comportamenti nei casi specifici, e compararle dal punto di vista dell'efficacia e dell'efficienza. Nello specifico delle tecniche sopra introdotte, discutere il problema della frammentazione ed identificare le soluzioni più comunemente adottate.

L'allocazione dinamica della memoria in un sistema di archiviazione di massa è un processo attraverso il quale lo spazio di archiviazione viene gestito e distribuito tra i vari file e le applicazioni che necessitano di memorizzare dati. Questo processo deve essere efficiente per garantire che le operazioni di lettura e scrittura siano rapide e che lo spazio disponibile sia utilizzato in modo ottimale.
Le tecniche più utilizzate sono: •Best fit. •First fit. •Worst fit. •Rotating first fit / next fit.

La tecnica del best fit è un metodo utilizzato principalmente in problemi di allocazione della memoria o nel contesto di problemi di ottimizzazione e approssimazione.
il best fit è un algoritmo utilizzato per allocare memoria a blocchi di dimensioni variabili. L'idea principale è di trovare il blocco di memoria libero che si adatta meglio alla richiesta di memoria. Questo algoritmo opera in tre fasi:
- Ricerca del Blocco Adeguato**:** Quando un processo richiede una certa quantità di memoria, l'algoritmo scorre l'elenco dei blocchi di memoria liberi e trova quello che ha la dimensione più piccola tra quelli che sono almeno grandi quanto la quantità richiesta.
- Allocazione**:** Una volta trovato il blocco che si adatta meglio, una porzione di quel blocco viene assegnata al processo. Se rimane spazio inutilizzato nel blocco (più grande della quantità minima richiesta per la gestione della memoria), questo spazio residuo viene mantenuto come un nuovo blocco libero.
- Gestione dei Blocchi Liberi**:** L'elenco dei blocchi liberi viene aggiornato, mantenendo traccia di quali blocchi sono stati allocati e quali sono ancora disponibili.

Ha il vantaggio di minimizzare lo spreco di memoria interna, poiché cerca di trovare il blocco più vicino alla dimensione richiesta, tuttavia questa tecnica può portare a frammentazione esterna, con molti piccoli blocchi di memoria liberi che non possono essere utilizzati per richieste future di memoria più grandi.
Dunque la sua efficacia è quella di ridurre la frammentazione interna selezionando il blocco più piccolo adeguato. Tuttavia, può aumentare la frammentazione esterna nel lungo termine.
La sua efficienza sta nel tempo di allocazione, è più lento del first fit, dato che deve esaminare tutti i blocchi liberi. Un altro suo punto di efficienza è nel tempo di deallocazione, questo algoritmo può richiedere più tempo per mantenere l'elenco ordinato.

La tecnica first fit è un algoritmo utilizzato per l'allocazione dinamica della memoria in cui si cerca il primo blocco di memoria libero che è abbastanza grande da soddisfare la richiesta.
Quando un processo richiede un blocco di memoria di una certa dimensione, l'algoritmo first fit scorre l'elenco dei blocchi di memoria liberi e seleziona il primo blocco che è sufficientemente grande per soddisfare la richiesta. Una volta trovato il blocco appropriato, parte di questo

blocco viene allocata al processo, e se rimane spazio libero, questo spazio viene mantenuto come un nuovo blocco libero.
Questo algoritmo lavora in 4 passaggi:

- Richiesta di Memoria**:** Un processo richiede un blocco di memoria di dimensione.
- Ricerca**:** L'algoritmo scorre l'elenco dei blocchi di memoria liberi dall'inizio e trova il primo blocco che ha una dimensione maggiore o uguale ad n
- Allocazione**:**

● Se il blocco trovato ha una dimensione esattamente uguale a n, l'intero blocco viene allocato al processo.
● Se il blocco trovato è più grande di n, viene allocata una porzione di n unità al processo, e il resto del blocco viene mantenuto come blocco libero.

- Aggiornamento dell'Elenco**:** L'elenco dei blocchi liberi viene aggiornato per riflettere la nuova allocazione.

Il vantaggio di questa tecnica è la sua semplicità, dato che è facile da implementare e comprendere. Inoltre è veloce nell'allocazione, poiché si ferma non appena trova un blocco adeguato.
Tuttavia possiede due svantaggi: quello della frammentazione esterna, dato che questa tecnica può lasciare molti piccoli blocchi di memoria non utilizzati (frammenti) che non possono essere facilmente riutilizzati per richieste future.
Inoltre l'efficienza dell'algoritmo può diminuire nel tempo, man mano che i blocchi di memoria si frammentano e diventa più difficile trovare un blocco sufficientemente grande vicino all'inizio dell'elenco.
La sua efficacia è la sua tendenza a causare frammentazione esterna, poiché allocazioni successive possono lasciare piccoli buchi di memoria inutilizzabili.

La tecnica worst fit è un algoritmo utilizzato per l'allocazione dinamica della memoria. Contrariamente alla tecnica first fit, che cerca il primo blocco sufficientemente grande, la tecnica worst fit cerca il blocco più grande tra quelli disponibili per soddisfare una richiesta di memoria. L'idea alla base di questa tecnica è di lasciare blocchi di memoria liberi più grandi, riducendo così la frammentazione esterna.
Quando un processo richiede un blocco di memoria di una certa dimensione, l'algoritmo worst fit scorre l'elenco dei blocchi di memoria liberi e seleziona il blocco più grande disponibile. Una volta trovato il blocco più grande, parte di questo blocco viene allocata al processo, e il resto rimane come un nuovo blocco libero.
Anche questo algoritmo lavora in 4 passaggi:

- Richiesta di Memoria**:** Un processo richiede un blocco di memoria di dimensione n.
- Ricerca**:** L'algoritmo scorre l'elenco dei blocchi di memoria liberi e trova il blocco con la dimensione maggiore.
- Allocazione**:**

● Se il blocco trovato ha una dimensione esattamente uguale a n, l'intero blocco viene allocato al processo.
● Se il blocco trovato è più grande di n, viene allocata una porzione di n unità al processo, e il resto del blocco viene mantenuto come blocco libero.

- Aggiornamento dell'Elenco**:** L'elenco dei blocchi liberi viene aggiornato per riflettere la nuova allocazione.

Il suoi vantaggi sono:

- Riduzione della Frammentazione Esterna**:** Allocando memoria dai blocchi più grandi, si tende a lasciare blocchi liberi più grandi, che sono più facili da riutilizzare per future richieste di memoria.
- Migliore Utilizzo dei Grandi Blocchi**:** Gli spazi di memoria più grandi vengono utilizzati in modo più efficiente.

Mentre gli svantaggi sono:

- Tempo di Ricerca: La ricerca del blocco più grande può richiedere tempo, soprattutto se l'elenco dei blocchi liberi è lungo.
- Frammentazione Interna: Allocando da blocchi molto grandi, si può lasciare una quantità significativa di memoria inutilizzata all'interno del blocco allocato.

La sua efficacia sta nella riduzione della frammentazione esterna lasciando grandi blocchi liberi. Tuttavia, può causare frammentazione interna. Mentre la sua efficienza sta nel tempo di allocazione, che può essere lento e nel tempo di deallocazione, è simile al best fit, può essere inefficiente se l'elenco è mantenuto ordinato.

La tecnica next fit è un algoritmo di allocazione dinamica della memoria che è una variante del first fit. A differenza del first fit, che inizia sempre la ricerca dall'inizio dell'elenco dei blocchi liberi, il next fit riprende la ricerca dal punto in cui si era fermato l'ultima volta. Questo approccio può contribuire a distribuire le allocazioni in modo più uniforme attraverso l'intero spazio di memoria.
Quando un processo richiede un blocco di memoria, l'algoritmo next fit cerca il primo blocco libero che è sufficientemente grande per soddisfare la richiesta, ma la ricerca riprende dal punto in cui si era fermata l'ultima volta, non dall'inizio della lista dei blocchi liberi.

Questo algoritmo avviene in 5 passaggi:
- Richiesta di Memoria: Un processo richiede un blocco di memoria di dimensione n.
- Ricerca: L'algoritmo inizia a cercare un blocco libero dalla posizione successiva all'ultima allocazione effettuata.
- Allocazione:
- Se trova un blocco di dimensione esattamente uguale a n, l'intero blocco viene allocato al processo.
- Se trova un blocco più grande di n, viene allocata una porzione di n unità al processo, e il resto del blocco rimane libero.
- Aggiornamento della Posizione: La posizione da cui riprendere la ricerca per la prossima allocazione viene aggiornata alla posizione subito successiva al blocco appena allocato.

- Elenco dei Blocchi**:** L'elenco dei blocchi liberi viene aggiornato per riflettere la nuova allocazione.

L’efficacia del next fit sta nella frammentazione, simile al first fit, ma con un comportamento più prevedibile nel tempo.
Mentre la sua efficienza si divide in tempo di allocazione, solitamente veloce, continua da dove si era interrotta l’ultima ricerca e tempo di deallocazione, questo è molto rapido, ma può richiedere la fusione di blocchi.

In conclusione:

- First Fit e Next Fit: sono buoni per velocità di allocazione e semplicità, ma possono soffrire di frammentazione esterna.
- Best Fit e Worst Fit: possono migliorare l'utilizzo della memoria, ma a costo di tempi di allocazione più lunghi e possibile frammentazione.

La scelta della tecnica dipende dalle specifiche esigenze dell'applicazione e dal comportamento previsto del carico di lavoro.

Il problema della frammentazione si verifica quando la memoria disponibile è divisa in blocchi più piccoli, ma la dimensione totale dei blocchi liberi non è sufficiente per soddisfare una richiesta di memoria, anche se la memoria totale libera potrebbe essere sufficiente. Ci sono due tipi principali di frammentazione: frammentazione esterna e frammentazione interna.

1. Frammentazione Esterna: Si verifica quando ci sono molti piccoli blocchi di memoria libera sparsi in tutto lo spazio di memoria, ma nessuno di essi è abbastanza grande da soddisfare una particolare richiesta di allocazione. Questo può portare a una situazione in cui, pur avendo abbastanza memoria libera complessiva, non è possibile allocare un blocco di memoria di una certa dimensione a causa della dispersione dei blocchi liberi.
2. Frammentazione Interna: Si verifica quando un blocco allocato di memoria è leggermente più grande della quantità richiesta effettiva. Questo si traduce in una porzione di memoria all'interno del blocco che non viene utilizzata effettivamente dal processo. La frammentazione interna si verifica spesso quando i blocchi di memoria sono allocati in unità fisse più grandi rispetto alle richieste effettive dei processi.

Le tecniche di allocazione della memoria come Best Fit, First Fit, Worst Fit e Next Fit mirano a gestire la frammentazione della memoria in sistemi di archiviazione.

Nella tecnica del Best Fit, quando un nuovo file deve essere memorizzato, viene cercato lo spazio libero più piccolo che sia sufficiente per contenere il file. Questo metodo tende a ridurre la frammentazione interna, poiché cerca di trovare il blocco più adatto alle dimensioni del file. Tuttavia, può portare a frammentazione esterna poiché possono essere lasciati spazi liberi troppo piccoli tra i blocchi assegnati. La soluzione è la compattazione: periodicamente, il sistema può eseguire operazioni di compattazione per riorganizzare i blocchi di memoria e ridurre la frammentazione esterna.

Con il metodo del First Fit, il sistema assegna il primo spazio libero che soddisfa le dimensioni richieste del file. Questo approccio è semplice e veloce, ma può portare a frammentazione esterna, poiché potrebbero essere lasciati spazi liberi non utilizzati tra i blocchi assegnati.

Soluzione alla Frammentazione → Garbage Collection**:** Un processo periodico di pulizia della memoria può recuperare spazi liberi non utilizzati e ridurre la frammentazione esterna.

Nel Worst Fit, il sistema assegna lo spazio libero più grande disponibile che soddisfi le dimensioni del file. Questo può aiutare a ridurre la frammentazione esterna poiché i blocchi assegnati sono generalmente più grandi, ma può aumentare la frammentazione interna.

La soluzione è allocazione a Blocchi di Dimensioni Variabili: Questa tecnica può essere utilizzata per adattare meglio le dimensioni dei blocchi ai requisiti dei file e ridurre la frammentazione interna.

Next Fit è simile a First Fit, ma inizia a cercare lo spazio libero dal punto in cui è stato lasciato l'ultimo blocco assegnato anziché dall'inizio della memoria. Questo può ridurre leggermente la frammentazione esterna rispetto al First Fit, ma può ancora comportare lo spreco di spazio libero tra i blocchi.

Soluzione alla frammentazione → Tecniche di Garbage Collection, dove la pulizia periodica può recuperare spazi liberi non utilizzati e ridurre la frammentazione esterna.

2- Introdurre il concetto di Job-Shop Scheduling per i processi, con particolare riguardo al problema della semantica della concorrenza per processi senza controllo dell’interleaving. Per il più comune algoritmo di gestione della concorrenza con tecniche di scope control, descrivere il metodo di gestione dello stato dell'assegnazione mediante semafori e discutere gli effetti sul throughput e sul waiting time del time sharing.

Il Job-Shop Scheduling è un problema classico nella teoria della schedulazione e dell'ottimizzazione operativa. In questo contesto, si tratta di assegnare un insieme di lavori a un insieme di risorse (macchine) in modo tale da ottimizzare un certo obiettivo, come minimizzare il tempo totale di completamento, il ritardo totale, o altre metriche di performance.

Ogni lavoro è composto da una sequenza di operazioni che devono essere eseguite in un ordine specifico su determinate macchine. Ogni macchina può eseguire una sola operazione alla volta, e ogni operazione ha una durata specifica.

In ambito di sistemi operativi e programmazione concorrente, il problema della concorrenza riguarda la gestione di più processi o thread che vengono eseguiti simultaneamente, condividendo risorse comuni. Questo è particolarmente critico in sistemi multi-threaded o in ambiente di Job-Shop Scheduling dove più processi competono per risorse comuni.

Il problema della semantica della concorrenza senza controllo dell'interleaving si presenta quando l'ordine di esecuzione delle operazioni non è deterministico e non è controllato. Questo può portare a vari problemi come:

1. Race Conditions (Condizioni di Gara):
    - Si verificano quando due o più processi accedono a una risorsa condivisa (come una variabile) contemporaneamente, e l'output dipende dall'ordine specifico di esecuzione delle operazioni.
2. Deadlock (Stallo):
    - Accade quando due o più processi si bloccano aspettando l'uno l'azione dell'altro, senza che nessuno dei due possa procedere.
3. Livelock (Vita Inutile):
    - Simile al deadlock, ma invece di bloccarsi, i processi continuano a cambiare stato senza fare progressi.
4. Starvation (Inanizione):
    - Alcuni processi potrebbero non ottenere mai le risorse necessarie per procedere perché altre operazioni continuano a prevaricare.

I semafori sono uno degli strumenti più comuni per la gestione della concorrenza e del controllo dell'accesso alle risorse condivise in un sistema operativo. I semafori permettono di sincronizzare l'esecuzione dei processi, evitando problemi come race conditions, deadlock e starvation.

Lo scope control con semafori implica l'uso di semafori per delimitare le sezioni critiche di codice che accedono a risorse condivise. Ogni volta che un processo entra in una sezione critica, acquisisce il semaforo (wait) e lo rilascia (signal) quando esce dalla sezione critica.

Il throughput rappresenta il numero di processi completati in un'unità di tempo. I semafori possono avere effetti positivi e negativi sul throughput:

- Positivi:
    - Sincronizzando l'accesso alle risorse, i semafori evitano race conditions, che possono portare a errori e crash di sistema, migliorando la stabilità e il throughput.
- Negativi:
    - L'acquisizione e il rilascio di semafori introducono un overhead. In un sistema molto concorrente, questo overhead può ridurre il throughput se non gestito correttamente.
    - Se non utilizzati correttamente, i semafori possono portare a deadlock, dove i processi rimangono bloccati, riducendo drasticamente il throughput.

Il waiting time è il tempo medio che un processo trascorre in attesa di accedere a una risorsa.

- Positivi:
    - Se usati con strategie adeguate i semafori possono evitare la starvation, assicurando che ogni processo ottenga le risorse necessarie in un tempo ragionevole.

- Negativi:
    - Se molteplici processi competono per lo stesso semaforo, il waiting time può aumentare. Questo è particolarmente vero per sezioni critiche lunghe o se le risorse sono limitate.

Il time sharing è una tecnica di gestione della CPU che consente a più processi di condividere la CPU, passando da un processo all'altro in modo rapido. L'uso di semafori nel contesto del time sharing ha alcuni effetti specifici:

1. Efficienza del Contesto di Switching:
    - I semafori possono migliorare la gestione del contesto di switching, riducendo i tempi morti dovuti alla sincronizzazione e migliorando l'uso della CPU.
2. Preemption:
    - Nei sistemi time-sharing, la preemption deve gestire correttamente i semafori per evitare di lasciare sezioni critiche incomplete, il che potrebbe portare a inconsistenze.

3- Analizzare le modalità di soluzione del problema del coordinamento di processi concorrenti in presenza di un sistema reader-writer a due processi. Mostrare la tecnica di eliminazione dei side effects dell'interleaving mediante semafori e provare con pseudocodice specifico come provocare il coordinamento mediante priorità, o mediante altre tecniche di controllo, se conosciute. Non è necessario implementare codice corrispondente.

Il problema dei Reader-Writer è un classico problema di sincronizzazione in sistemi operativi, dove ci sono processi che leggono e processi che scrivono su una risorsa condivisa (come un file o un database). La sfida principale è garantire che i lettori possano leggere simultaneamente senza interferenze, ma gli scrittori necessitano di accesso esclusivo per evitare inconsistenze.

Esistono due principali varianti del problema Reader-Writer, ognuna con la propria politica di priorità:

Priorità ai Lettori, in questa soluzione, i lettori hanno la precedenza sugli scrittori. Gli scrittori possono scrivere solo quando non ci sono lettori attivi.

Priorità agli Scrittori, in questa soluzione, gli scrittori hanno la precedenza sui lettori. I lettori possono leggere solo quando non ci sono scrittori in attesa.
La soluzione al problema reader-writer a due processi utilizza semafori per garantire l'accesso coordinato e sicuro alla risorsa condivisa, assicurando che i lettori possano leggere contemporaneamente senza interferenze e che gli scrittori possano scrivere esclusivamente. Questa implementazione bilancia efficacemente throughput e waiting time, rendendola adatta a scenari con un solo lettore e uno scrittore.

L'eliminazione dei side effects dell'interleaving nei sistemi concorrenti si ottiene sincronizzando l'accesso alle risorse condivise in modo che le operazioni critiche vengano eseguite in modo atomico. Utilizziamo i semafori per garantire che solo un processo alla volta possa eseguire

operazioni critiche su una risorsa condivisa. Supponiamo di voler implementare una priorità per gli scrittori, garantendo che se uno scrittore vuole accedere alla risorsa, i lettori devono attendere. Utilizziamo semafori per implementare questa priorità.

Utilizzando semaphore wrt, garantiamo che solo un processo alla volta (lettore o scrittore) possa accedere alla risorsa condivisa in modo esclusivo.

Il semaforo queue assicura che quando un scrittore è in attesa, nessun nuovo lettore può accedere alla risorsa fino a quando lo scrittore non ha terminato.
Il contatore read_count e il semaforo mutex coordinano i lettori, garantendo che possano accedere alla risorsa simultaneamente ma bloccando gli scrittori quando almeno un lettore è attivo.