Sistemi Operativi 03/05/2012

Esercizi sul file system:

Esercizio 1:
Un file è grande 500KByte, i blocchi per l'allocazione sono di 1KByte ciascuno.
Sappiamo che un puntatore a blocco mi porta via 4Byte. **Gli attributi del file (informazioni necessarie per reperire il contenuto del file su disco) sono già stati caricati in memoria centrale.**

1) Quanti accessi a disco sono necessari per leggere il blocco 260? (nei 3 tipi di allocazione)
2) Quanta memoria è "sprecata"? (nei 3 tipi di allocazione)

Soluzione:
1)

- **Contigua:** 1 Accesso (parto dall'indirizzo del primo e scorro fino al blocco che serve).
- **Concatenata:** 260 accessi
- **Indicizzata:** 3 Accessi (consulto la tabella e risalgo al blocco); sarebbe 2 se la nostra informazione stesse in un unico blocco, ma nel nostro caso, il nostro indice è in un'altra tabella, perché in un blocco possiamo contenere solo i primi 256 indirizzi (1024/4), per cui supponiamo che i blocchi contenenti le tabelle siano "linkati" fra di loro, dovrò consultare in ordine le tabelle fino a trovare il mio indirizzo, che nel nostro caso sarà nel secondo blocco.

2)

- **Contigua:** 0 sprechi per memorizzare il file
- **Concatenata:** 500*4 Byte, nel caso ci sia un solo puntatore per blocco. Lo spreco di memoria avviene solo per memorizzare i puntatori, quindi nel nostro caso ogni blocco ha il suo puntatore.

Se la lista fosse concatenata doppia, ci sarebbero due puntatori ovvero 500*8 Byte sprecati.

- **Indicizzata:** 2KByte utilizzati per memorizzare la tabella.

Esercizio 2:
Prendendo in esame una linked list (lista concatenata), siamo posizionati sul blocco 150 (il file è diviso in almeno 150 blocchi), vogliamo accedere al blocco 99.

1) Quanti accessi al disco?

Soluzione:
1) Spezzo in due casi:
    1. Nel caso in cui la lista sia "simply linked" occorre scorrere dall'inizio il file, per cui occorrono 99 accessi.
    2. Nel caso in cui la lista sia "double linked" mi basta scorrere a ritroso la lista fino al blocco desiderato, questo richiederà 51 accessi.

Esercizio 3:
Il Sistema Operativo può decidere il tipo di allocazione in base a:

(i) Il numero di blocchi occupati da un file;
(ii) Il tipo di accesso al file (modo di muoversi sui record logici: sequenziale/diretto).

I file sono:

A. 1 blocco solo, accesso sequenziale.
B. 100 blocchi, accesso diretto.
C. 1 blocco, accesso diretto.
D. 100 blocchi, accesso sequenziale.

Soluzione:

A. Concatenata o Contigua sono le scelte migliori, nel caso in cui il puntatore della concatenata resti all'interno del blocco. La scelta indicizzata è da scartare poiché servirebbe comunque un blocco per memorizzare la tabella.

B. La scelta concatenata è da scartare a priori con l'accesso diretto, perché ne rallenta molto l'esecuzione (caso peggiore). Nei sistemi operativi moderni, la scelta indicizzata è migliore poiché contribuisce al meglio alla dinamicità del sistema (evito i problemi di frammentazione), se non ho problemi di memoria contigua, anche l'allocazione contigua va comunque bene, evitandomi anche l'accesso alla tabella di indicizzazione.

C. Come per il punto A, la scelta dell'accesso diretto non incide sull'allocazione in questo caso, poiché il blocco su cui è memorizzato il file è unico.

D. In questo caso tutte le scelte possono essere valide, la scelta contigua è meno preferibile per i problemi di frammentazione o derivanti dal trovare spazio contiguo di allocazione.
I vantaggi derivanti dall'indicizzata comporta comunque due accessi al disco, mentre la concatenata doppia comporta una tolleranza ai guasti maggiore.

Esercizio 4 (Free Space Management):
Quando si parla di File System, c'è un discorso di come gestire i blocchi liberi, nel caso dell'allocazione concatenata o indicizzata, questi blocchi possono essere sparsi ovunque sul disco.

- Una soluzione è il **bit vector**, ovvero un vettore di bit della dimensione dei blocchi della memoria.

Salvato in memoria centrale, se il valore è → '1' il blocco è libero (“free”) altrimenti → '0' occupato.
Il vantaggio è che per verificare un'allocazione contigua basta scorrere questo bit vector per trovare n elementi ad '1' consecutivi.

- Un'altra possibilità è avere una **lista** dove ogni elemento libero è memorizzato dal valore di un elemento della lista, in questo caso non è facile verificare i blocchi liberi contigui (perché la lista potrebbe essere disordinata), ma è facile controllare gli elementi per l'allocazione concatenata o indicizzata perché mi occorre un blocco solo per volta.

Per quanto riguarda la fault tolerance è più grave un bit/elemento della lista “settato” in maniera errata piuttosto che uno non presente nel bit/lista, perché oggigiorno i dischi sono grandi quindi perdere un blocco non è tanto grave quanto sovrascrivere un dato al posto sbagliato.

Esercizio 5:

Abbiamo un Hard Disk della capienza di 2^34 Byte, formattato in blocchi da 1024 Byte (2^10 Byte). Assumiamo un'allocazione di tipo indicizzata.

1) Quanti bit mi servono per memorizzare il numero di un blocco?
2) Supponiamo di avere un file A, di dimensioni 400 KByte. Un solo blocco indice, è sufficiente per indirizzare tutti i blocchi che costituiscono questo file?
3) In questa situazione, quanti accessi a disco mi ci vorrebbero per leggere l'ultimo blocco del file?

Soluzione:

1) Abbiamo 2^34/2^10 blocchi, ovvero 2^24, per cui per enumerare questo insieme occorrono 24 bit.
2) Considerando che il mio file occupa 400 blocchi, abbiamo visto che ogni puntatore a blocco è descritto da 24 bit, ovvero 3 Byte, per vedere tutti i blocchi del file, mi ci vorrebbero 1200 Byte, per cui mi occorrerebbe più di blocco per contenere la tabella degli indici, perché in ogni blocco ci sono 1024 Byte, per cui necessiteremmo di esattamente 2 Blocchi.
3) 3 Accessi (2 Tabelle concatenate + 1 Accesso per il blocco che mi interessa).