

Il file system: esercizi

Appello AE-2 del 19/3/2003

Quesito 3. Un *file system* utilizza un vettore di *bit* (*bitmap*), con posizioni numerate da sinistra verso destra, per indicare i blocchi liberi presenti in una data partizione di disco. La formattazione della partizione libera tutti i blocchi tranne il 1°, che viene assegnato alla *directory* radice. In tale sistema, l'assegnazione di blocchi liberi a *file* considera sempre prima i blocchi di indice minore, trascurando ogni considerazione di contiguità. Si mostri il contenuto della parte iniziale di tale vettore dopo ciascuna delle seguenti operazioni:

- Scrittura del *file* A, ampio 6 blocchi
- Scrittura del *file* B, ampio 5 blocchi
- Rimozione del *file* A
- Scrittura del *file* C, ampio 8 blocchi
- Rimozione del *file* B

Si mostri poi l'evoluzione del contenuto di tale vettore a fronte della medesima sequenza di operazioni nel caso in cui il sistema volesse assicurare la massima contiguità dei blocchi assegnati ad ogni *file*.

Soluzione

Soluzione 3 (punti 5). Seguiamo l'evoluzione del contenuto della maschera nel primo caso, concentrandoci sulle sue prime posizioni:

Dopo la formattazione:	1000000000000000 ...
Scrittura del file A:	1111111000000000 ...
Scrittura del file B:	111111111111000 ...
Rimozione del file A:	100000011111000 ...
Scrittura del file C:	111111111111110 ...
Rimozione del file B:	111111100000110 ...

Vediamone ora l'evoluzione nel secondo caso, nel quale prevalgono considerazioni di contiguità dei blocchi assegnati ai file:

Dopo la formattazione:	1000000000000000 ...	
Scrittura del file A:	1111111000000000 ...	allocazione contigua senza frammentazione esterna
Scrittura del file B:	111111111111000 ...	allocazione contigua senza frammentazione esterna
Rimozione del file A:	100000011111000 ...	
Scrittura del file C:	1000000111111111110 ...	allocazione contigua con frammentazione esterna
Rimozione del file B:	10000000000011111110 ...	

Appello AE-2 del 19/3/2003

Quesito 4. Il progettista di un sistema operativo ha deciso di usare nodi indice (*i-node*) per la realizzazione del proprio *file system*, stabilendo che essi abbiano la stessa dimensione di un blocco, fissata a 512 *byte*. Il progettista ha poi deciso che un nodo indice primario contenga 12 campi di indirizzo di blocchi di disco e 2 campi puntatori a nodi indice di primo e secondo livello di indirizzazione rispettivamente. Sapendo che gli indirizzi di blocco sono espressi su 32 *bit*, si vuole allocare un *file* logicamente composto da 10.000 *record* da 80 *byte* ciascuno, imponendo che un *record* non possa essere suddiviso su due blocchi. Calcolare quanti blocchi verranno utilizzati per allocare il *file* dati e quanti per gestire la sua allocazione tramite nodi indice. Determinare inoltre l'occupazione totale in memoria secondaria risultante da tale strategia di allocazione.

Soluzione

Soluzione 4 (punti 5). Richiamiamo i dati del problema:

N_R = numero di *record* che compongono il *file* = 10.000

D_R = dimensione di un *record* = 80 *byte*

D_I = dimensione di un indirizzo = 4 *byte*

D_B = dimensione di un blocco = 512 *byte*

N_{RB} = numero di *record* per blocco = $\text{int}(D_B/D_R) = \text{int}(512/80) = \text{int}(6,4) = 6$

N_{BF} = numero di blocchi occupati dal *file* = $1 + \text{int}(N_R/N_{RB}) = 1 + \text{int}(10000/6) = 1.667$

N_{IB} = numero di indirizzi in un blocco = $D_B/D_I = 512/4 = 128$

N_{ID} = numero di indirizzi in X blocchi a doppia indirazione = X^2

I blocchi da indirizzare sono $N_{BF} = 1.667$

- di questi, 12 possono essere indirizzati direttamente dal nodo indice principale
- dei rimanenti $1.667 - 12 = 1.655$, N_{IB} (cioè 128) sono indirizzabili ad indirazione singola tramite l'indirizzo ad indirazione singola del nodo indice principale
- dei rimanenti $1.655 - 128 = 1.527$, si possono utilizzare blocchi indiretti di 128 con indirazione doppia, come mostrato in figura.

Con le indicazioni riportate in figura possiamo concludere che:

- per allocare il *file* dati sono necessari NBF blocchi, cioè 1.667 blocchi
- per gestire l'allocazione del *file* sono necessari:
 - 1 blocco per il nodo indice principale
 - 1 blocco di indirizzi ad indirazione singola
 - $1 + 12$ blocchi per l'indirazione doppia

per un totale di $1 + 1 + 1 + 12 = 15$ blocchi

- l'occupazione totale in memoria secondaria vale $1.667 + 15 = 1.682$ blocchi da 512 *byte*, per un totale di $1.682 \cdot 512 = 861.184 = \text{byte} = 841 \text{ kB}$

Appello AE-2 del 14/9/2005

Quesito 1 (punti 8). Sia data una memoria secondaria di ampiezza 64 GB, organizzata in blocchi di ampiezza 1 kB. Dopo aver calcolato la dimensione minima di un indice di blocco per tale memoria, sotto il vincolo che essa debba essere un multiplo di 8 (*bit*), si determini la dimensione massima di file ottenibile nel caso pessimo di contiguità nulla sotto le seguenti ipotesi:

1. *file system* di tipo NTFS, con *record* ampi 1 kB, 408 B riservati all'attributo dati nel *record* principale ed 800 B nel *record* di estensione, utilizzando esattamente 2 record;
2. *file system* di tipo Extfs, con *i-node* ampi 1 kB, nodo principale contenente 16 indici di blocco ed 1 indice di I e di II indizione, utilizzando l'intero i-node principale.

Calcolate le dimensioni richieste, si determini per ciascun tipo di *file system*, il rapporto inflattivo determinato dalla sua organizzazione strutturale, ossia l'onere proporzionale dovuto alla memorizzazione delle strutture di rappresentazione rispetto a quella dei dati veri e propri.

Soluzione

Soluzione 1 (punti 8). Essendo la memoria secondaria ampia 64 GB e i blocchi ampi 1 kB, è immediato calcolare che siano necessari $\lceil \frac{64 \text{ GB}}{1 \text{ kB}} \rceil = 64 \text{ M} = 2^5 \times 2^{20} = 2^{25}$ indici, la cui rappresentazione binaria banalmente richiede 26 bit. Stante il vincolo che la dimensione dell'indice debba essere un multiplo di 8 bit, la dimensione dell'indice deve salire a 32 bit (4 B). Vediamo ora quale possa essere la dimensione massima di file ottenibile sotto le ipotesi fissate dal quesito.

File system di tipo NTFS : Dei 408 B riservati all'attributo dati nel record principale, $2 \times 4 = 8$ B saranno riservati alla coppia {base, indice}, mentre i rimanenti $408 - 8 = 400$ B potranno essere utilizzati per indicare le sequenze contigue di caso peggiore (dunque tutte ampie 1 blocco). Poiché ciascuna sequenza richiede una coppia di indici {inizio, fine}, pari a $2 \times 4 = 8$ B, il record principale potrà ospitare $\lfloor \frac{400 \text{ B}}{8 \text{ B}} \rfloor = 50$ sequenze ampie 1 blocco. Il record di estensione dispone invece di 800 B per la memorizzazione di $\lfloor \frac{800 \text{ B}}{8 \text{ B}} \rfloor = 100$ ulteriori sequenze. Ne segue che, sotto le ipotesi del quesito, la dimensione massima di file consentita da NTFS è pari a: $(50 + 100) \text{ blocchi} \times 1 \text{ kB/blocco} = 150 \text{ kB}$, al costo di 2 record, ciascuno ampio 1 kB. Il rapporto inflattivo in questo caso è dunque pari a: $\frac{2 \text{ kB}}{150 \text{ kB}} = 1.33\%$.

file system di tipo Extfs : In questo caso, utilizzando tutti i campi dell'i-node principale, abbiamo a disposizione:

- 16 indici diretti di blocco, al costo di 1 blocco poiché un i-node occupa lo stesso spazio di un blocco;
- 1 indice di I indirezione, il quale punta ad un i-node interamente utilizzato per contenere indici diretti di blocco, che consente di esprimere fino a: $\lfloor \frac{1 \text{ kB}}{4 \text{ B}} \rfloor = \lfloor \frac{2^{10}}{2^2} \rfloor = 2^8 = 256$ indici di blocco, al costo di 1 ulteriore blocco;
- 1 indice di II indirezione, il quale punta ad un i-node speciale, interamente utilizzato per contenere puntatori ad i-node di I livello, che dunque consente di esprimere 256 puntatori a strutture ciascuna contenente fino a 256 indici diretti di blocco, per un totale di $256^2 = (2^8)^2 = 2^{16} = 65.536$ blocchi, al costo di $1 + 256 = 257$ ulteriori blocchi.

Conseguentemente, Extfs consente di rappresentare file di dimensione massima pari a: $(16 + 256 + 65.536) \times 1 \text{ kB} = 65.808 \text{ kB} = 64 \text{ MB} + 272 \text{ kB}$ (438,72 volte maggiore di quanto ottenuto con NTFS) per un rapporto inflattivo pari a: $\frac{(1+1+257) \times 1 \text{ kB}}{65.808 \text{ kB}} = 0,39\%$ (3,4 volte inferiore a quanto ottenuto con NTFS).

La considerazione ovvia da trarre da queste considerazioni è che NTFS) è particolarmente penalizzato dalle condizioni di scarsa o nulla contiguità. (Per contro, ad Extfs la contiguità non giova in alcun modo.)