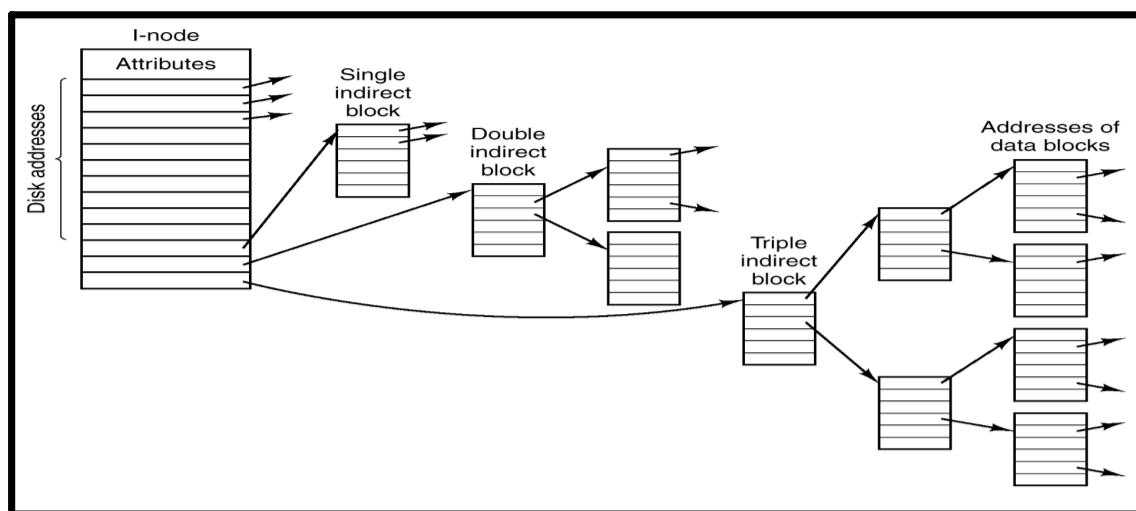


Classica struttura di un i-node

lunedì 17 aprile 2023 14:29

Classica struttura di un i-node



L'i-node non è altro che una tabella: ha una zona dove noi abbiamo gli attributi e poi abbiamo una serie di indicizzatori di blocchi del disco. Nella prima entry sotto "attributes" abbiamo un indice per un BLOCCO del dispositivo, che contiene i dati di quel file.

Poi ci sarà una seconda entry che ci identificherà un altro blocco in cui sono presenti altri byte, ossia altri record, di quel file; e così via...!

I primi 10 elementi puntano esattamente dove sono eventualmente mantenuti i dati del file.

Questo ci permette di dire che noi siamo in grado di indicizzare in maniera diretta, ossia manteniamo all'interno di questo I-NODES le informazioni di dove sono posizionati i blocchi associati al file all'interno del dispositivo di memoria di massa, per dei file che possono avere una taglia pari a 10 x sizeof(Blocco).

Supponiamo che il file sia più grande, quindi non possiamo indicizzarlo con 10 indici diretti: abbiamo altri indici all'interno di un I-NODES? Sì, abbiamo gli ultimi 3.

Il primo dei 3 è un indice indiretto di primo livello: abbiamo l'indicizzazione di un ulteriore blocco del dispositivo all'interno del quale NON ci sono i dati del file, ma ci sono scritti altri indici, quindi riusciamo a reperire un'altra tabella di indici.

Ciascuno di questi indici, può a sua volta puntare ad un BLOCCO che contiene i dati del file.

Supponiamo che non ci basta neanche questa di tabella perché il file cresce ancora.

Il secondo dei 3 indici punta ad un blocco dove noi manteniamo indici che puntano ad altri blocchi dove noi manteniamo ancora indici, che puntano eventualmente a blocchi del file.

Il terzo ed ultimo indice punta ad una tabella di indici che possono puntare ad una tabella di indici che puntano ad una tabella di indici e dopo al blocco su memoria di massa.

Per file piccoli o relativamente piccoli noi abbiamo comunque gli indici diretti, arriviamo rapidamente ad identificare la posizione dei dati associati a questo file su dispositivo di memoria di massa, quindi eventualmente possiamo chiamare delle operazioni di I/O per chiamare questi dati in buffer cache.

Passare attraverso gli indici indiretti e caricare tutti gli altri blocchi in memoria viene effettuato SOLO se il file CRESCE di dimensione: a livello di sistema operativo dobbiamo eseguire più operazioni di I/O per andare a prelevare anche gli indici che ci portano sui dati effettivi che abbiamo a disposizione per quanto riguarda il file.

Se ad esempio consideriamo i blocchi di disco grandi 512 byte, quindi all'interno di ciascuno di questi blocchi possiamo mantenere 512 records del nostro file, e supponiamo che gli indirizzi su disco sono di 4 byte, all'interno di un blocco manteniamo 128 indici.

Parametri (dipendenti dalla versione di file system ed hard drive)

- Blocchi su disco da 512 byte
- Indirizzi su disco 4 byte
- In un blocco: $512/4 = 128$ indirizzi
 - ind. 1-10 10 blocchi dati
 - ind. 11 128 blocchi dati
 - ind. 12 128^2 blocchi dati
 - ind. 13 128^3 blocchi dati

$$\begin{aligned} \text{Maxfile} &= 10 + 128 + 128^2 + 128^3 = \\ &2.113.664 \text{ blocchi} * 512 \text{ bytes} = \\ &1.082.195.968 \approx 1\text{Gbyte.} \end{aligned}$$

Globalmente quanti indirizzi possiamo mantenere?
Utilizzando gli I-NODES e la indicizzazione a livelli multipli.

I primi 10 indici sono diretti, quindi sono 10 blocchi, mentre l'undicesimo indice punta ad un blocco in cui manteniamo indici, quanti ne manteniamo? 128.

Quindi abbiamo altri 128 indici che puntano a blocchi dati.

Se consideriamo il 12esimo indice, esso punta ad un blocco dove noi manteniamo 128 indirizzi, ciascuno di questi 128 indirizzi può puntare ad un altro blocco dove noi manteniamo altri 128, quindi abbiamo 128 al quadrato. E così via...!

Gli attributi di un i-node sono dei metadati per la gestione del contenuto indicizzato da tutti gli indici.