Paginazione su domanda (seconda parte)

G. Lettieri

15 Maggio 2018

Ci proponiamo ora di eliminare tutte le semplificazioni che abbiamo introdotto a livello hardware nella MMU, studiando come gestire l'albero di traduzione su più livelli e la presenza del TLB.

1 Tabella su più livelli

Dal momento che tutte le entrate delle tabelle, di ogni livello, possiedono il bit P, possiamo caricare dallo swap, su domanda, anche le tabelle. In questo modo le tabelle fungono anche da struttura dati che ci permette di trovare tutte le pagine dentro lo swap: è sufficiente conoscere il numero del blocco della tabella di livello 4. Questa conterrà inizialmente i numeri di blocco delle tabelle di livello 3 e così via fino alle pagine. All'avvio del processo possiamo caricare la sola tabella di livello 4 e cedere il controllo al processo.

Quando la routine di page fault va in esecuzione può leggere da `cr2` l'indirizzo che ha causato il fault, sia V, ma non sa in che punto la traduzione è stata interrotta. Per scoprirlo ripercorre il tragitto della MMU, ripetendo in software le sue azioni, fino a trovare il descrittore che contiene il bit P a zero. Il compito della routine è ora quello di caricare in memoria la pagina che contiene V e *tutte le eventuali tabelle mancanti* nel percorso di traduzione di V dal livello 4 al livello 1. Al termine della routine il processore rieseguirà l'istruzione che aveva causato il fault e questa volta la MMU riuscirà a completare la traduzione.

In pratica la routine di page fault della vera MMU deve fare le stesse azioni di quella di MMU_1, ma deve ripeterle più volte, una volta per ogni entità mancante nel percorso di traduzione. Vedremo il codice completo quando studieremo il nucleo del sistema.

Ricordiamoci che abbiamo deciso di usare la memoria M_2 per contenere sia le pagine che le tabelle. Questo comporta che dobbiamo modificare i descrittori di frame, in quanto i frame possono contenere sia tabelle (di vari livelli), sia pagine virtuali. Il descrittore di frame conterrà i seguenti campi:

- **livello**, con valori che vanno da -1 a 4: -1 indica che il frame è libero, 0 che contiene una pagina, 1–4 che contiene una tabella del corrispondente livello; i campi rimanenti sono significativi solo se il frame non è libero;

- **residente**, che può valere **true** o **false**: se true, indica che l'entità contenuta non può essere rimpiazzata;
- **indirizzo virtuale**: permette di risalire al descrittore che contiene il bit P per l'entità contenuta;
- **blocco**: numero del blocco nell'area di swap da cui è stata caricata l'entità contenuta;
- **contatore**: statistiche degli accessi all'entità contenuta.

Per aggiornare i contatori la routine di page fault deve consultare (e poi azzerare) i bit A di tutte le pagine e tabelle caricate. Tali bit sono contenuti a loro volta nelle tabelle e queste sono sparpagliate in M_2, ma è possibile ritrovarle tutte scorrendo i descrittori di frame e considerando solo quelli in cui il campo livello è maggiore di 0. Anche in questo caso vedremo il codice completo quando studieremo il nucleo.

Al momento di scegliere una vittima per il rimpiazzamento, dobbiamo stare attenti a non scegliere mai una tabella che abbia ancora qualche entrata con P=1, altrimenti renderemmo irraggiungibili tutte le entità a cui sta puntando (e quelle a cui queste puntano, etc.). Notiamo che il contatore delle statistiche di una tabella sarà sempre maggiore o uguale dei contatori di tutte le entità a cui essa punta (ogni volta che la MMU mette a 1 un bit A in un descrittore, lo mette a 1 anche nella catena di descrittori incontrati fino a quel punto). Se scegliamo come vittima sempre l'entità che ha il contatore minimo abbiamo dunque un solo caso critico: una tabella che ha il valore del contatore uguale ad almeno una delle entità a cui essa punta (questo può accadere se la tabella punta ad una sola entità). Per risolvere anche questo caso è sufficiente che, tra due contatori che hanno lo stesso valore, si scelga sempre quello il cui corrispondente campo livello è minore.

Una volta scelta una vittima, la routine di page fault deve renderla non presente. Come per la MMU_1, abbiamo il problema di dover risalire dal numero di frame al descrittore dell'entità in essa contenuta (in modo da porre P=0). Per le pagine virtuali (livello 0) il campo "indirizzo virtuale" continua ad avere lo stesso significato. Per le tabelle (livelli 1 e 4) abbiamo bisogno del numero di pagina virtuale di una qualunque delle pagine virtuali la cui traduzione passa dalla tabella in questione: è sufficiente che la routine di page fault, nel momento in cui carica una tabella, memorizzi nel campo "indirizzo virtuale" del corrispondente descrittore di frame l'indirizzo che ha causato il page fault corrente.

Il campo "blocco" si utilizza in modo simile a quanto visto per MMU_1. Anche qui conviene usare i descrittori che hanno P=0 per memorizzare il blocco di swap che contiene l'entità assente. Ora la cosa è particolarmente conveniente, in quanto la routine di page fault deve già accedere ai vari descrittori per scoprire quali di essi ha il bit P a 0: nel momento in cui lo trova ha anche subito a disposizione il blocco da cui caricare l'entità non presente. Una volta caricata l'entità il blocco deve essere copiato nel corrispondente descrittore di frame. Si

noti cosa accade alle tabelle: quando sono caricate sono vuote (tutti i bit P sono 0) e tutti i descrittori contengono i blocchi delle entità puntate nell'area di swap. Mentre si trovano in memoria fisica saranno modificate varie volte, per rendere presenti o assenti le entità di livello inferiore, ma quando sono selezionate come vittime sono sicuramente ritornate vuote, e ora tutti i descrittori puntano nuovamente agli stessi blocchi a cui puntavano quando la tabella era stata caricata. Quindi, sotto le ipotesi di funzionamento fin qui esaminate, non è mai necessario ricopiare nell'area di swap una tabella vittima.

2 Il TLB

Notiamo ora cosa comporta il fatto che, se il descrittore che sta cercando si trova già nel TLB, la MMU non percorre l'albero delle tabelle:

1. che succede quando la routine di page fault rimpiazza una pagina vittima?
2. la MMU non aggiornerà i bit A nei descrittori dopo la prima volta che ha usato un descrittore, almeno fino a quando quel descrittore resta nel TLB;
3. supponiamo che un descrittore venga caricato a causa di una operazione di lettura e abbia in quel momento il bit D a zero; poi il processore esegue una operazione di scrittura all'interno della stessa pagina e la MMU trova la traduzione nel TLB. Che accade al bit D nel descrittore in memoria?

Nel caso 1 la routine di page fault deve invalidare (usando l'istruzione `invlpg`) la traduzione della pagina vittima dopo aver posto il bit P a zero nel suo descrittore di livello 1. Se non lo facesse, e il TLB avesse una copia del descrittore caricata precedentemente, la MMU continuerebbe a tradurre l'indirizzo virtuale della pagina rimossa ottenendo l'indirizzo del frame che, dopo il rimpiazzamento, contiene ora una pagina virtuale completamente diversa.

Consideriamo il caso 2. Il problema, in questo caso, è che il mancato aggiornamento dei bit A per tutti i descrittori che si trovano nel TLB falsa le statistiche sugli accessi calcolate dalla routine di page fault. Quel che è peggio, proprio le pagine il cui descrittore si trova nel TLB, e che sono quindi probabilmente quelle usate più di recente, vedrebbero i loro contatori non aggiornati. Per rimediare a questo problema si richiede un intervento della routine di page fault: dopo aver letto e azzerato tutti i bit A, deve *invalidare* l'intero TLB. In questo modo gli accessi successivi costringeranno la MMU a percorrere la catena e rimettere a 1 i bit A delle pagine a cui il programma sta accedendo. Per invalidare tutto il TLB basta eseguire la coppia di istruzioni `movq %cr4, %rax; movq %rax, %cr3`. La seconda, infatti, invalida il TLB anche se il contenuto di `cr3` non cambia.

Consideriamo infine il caso 3. Il problema, in questo caso, è che la routine di page fault deve sapere se ci sono state scritture su una pagina perché, se la pagina è scelta come vittima, deve sapere se è necessario ricopiarla nell'area di swap. La routine di page fault ricava questa informazione dal bit D nei descrittori che si trovano in memoria (ricordiamo che non ha modo di sapere

cosa ci sia dentro il TLB), e dunque il valore di tale bit deve essere corretto. Questo problema è risolto dalla MMU, che agisce nel seguente modo:

1. quando il descrittore cercato non è nel TLB si comporta come già detto (percorre tutta la catena e poi carica il descrittore trovato nel TLB); questa operazione porta anche il valore corrente del bit D nel TLB;
2. se invece il descrittore cercato è nel TLB e
   (a) l'accesso è in scrittura e il bit D vale 0, si comporta come se il descrittore *non* fosse nel TLB;
   (b) negli altri casi (lettura, oppure scrittura con D=1) usa il descrittore nel TLB.

Il caso interessante è il 2.(a). Se in questo caso la MMU si limitasse ad usare l'informazione nel TLB senza andare ad aggiornare il descrittore in memoria, la routine di page fault potrebbe non sapere mai che c'è stata una scrittura su quella pagina. Invece la MMU percorre tutta la catena e aggiorna il bit D. Alla fine ricopia anche il descrittore nel TLB, e questa volta avrà il bit D=1: in questo modo (caso 2.(b)) questa operazione viene eseguita soltanto per la prima scrittura sulla pagina.