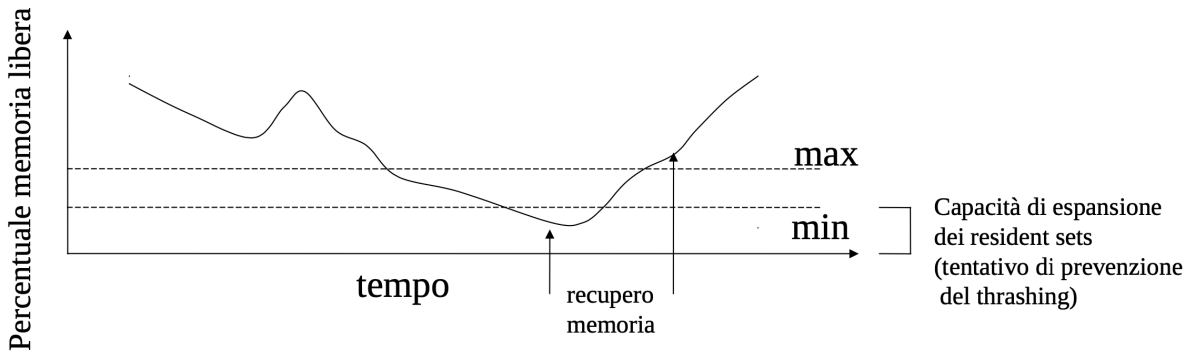
Sostituzione delle pagine in sistemi sistemi Unix

Come funziona tutto questo all'interno di un sistema UNIX?

Funzionamento con algoritmo dell'orologio con metodo di allocazione variabile, e il sistema libera periodicamente (tramite un demone) i frame quando la percentuale di memoria libera scende sotto una soglia minima.
Quindi se andiamo a toccare questo minimo lungo l'evoluzione temporale, noi cerchiamo di liberare la memoria prima che serva a qualcuna di queste applicazioni a eseguire qualche sostituzione. Questo lo facciamo utilizzando un demone, ossia un thread di sistema. Quindi ovviamente liberiamo i frame in modo tale da riottenere la memoria libera al di sopra di una soglia massima.

- algoritmo dell'orologio
- allocazione variabile (numero di frames variabile per processo)
- il sistema libera periodicamente (tramite demone) i frame di memoria quando la percentuale di memoria libera scende sotto una soglia minima
- sono liberati i frames necessari a riportare la memoria libera sopra una soglia massima



Questo lavoro lo fa quel thread e quest'ultimo viene schedulato insieme a tutti, quindi lui ci prova, poi se l'uso della memoria da parte delle applicazioni è talmente massivo che, la memoria che quel thread riesce a liberare non riesce a servire le richieste di memoria da parte delle altre applicazioni, chiaramente cadiamo in trashing. Ma cercare comunque di liberare in anticipo le pagine in modo tale da avere una quantità di memoria disponibile nel momento in cui le applicazioni possano richiedere di materializzare pagine in memoria, è importante per evitare di dover fermare queste applicazioni in uno scenario dove potremmo degradare verso il trashing.

In windows abbiamo una situazione similare.

Sostituzione delle pagine in sistemi Windows

La sostituzione di una pagina avviene nel resident set del processo, quindi abbiamo un ambito locale di sostituzione, però il resident set può essere espanso in caso di page fault qualora la memoria libera sia sopra una certa soglia. Quindi non c'è ancora un uso critico della memoria allora aumentiamo il resident set di un'applicazione che vuole materializzare qualche cosa in più all'interno della RAM.
Se la memoria libera scende sotto una certa soglia il gestore libera i frame utilizzando una politica NOT RECENTLY USED, e i frame che vengono liberati vengono liberati nell'ambito dei resident set di tutti i processi attivi. Quindi andiamo a ridiminuire tutti i resident set di tutti i processi utilizzando comunque l'algoritmo NOT RECENTLY USED per stabilire quali sono le cose che possiamo eliminare dalla memoria.

- la sosituzione di una pagina avviene nel resident set del processo (ambito locale)
- il resident set può essere espanso in caso di page fault qualora la memoria libera sia al di sopra di una data soglia
- se la memoria libera scende sotto una data soglia, il gestore libera i frame seguendo una politica di tipo not-recently-used
- i frame vengono liberati nell'ambito dei resident set di tutti i processi attivi