

# Sistemi operativi anno 2021-2022 registrazione: 2° semestre

2022-03-03

slide 01-arch-hw.pdf

Cravatta di oggi: "Ponti Giapponesi di Monet" qual'è il messaggio? È sempre impressionista, ma il messaggio è più nel PONTE oggi. Bisogna costruire ponti. Muri, tombe o altre cose distruggono il vivere civile, i ponti uniscono. Non a caso i Popi venivano spesso chiamati "pontefici" → che creavano ponti. OK, torniamo a Sistemi Operativi. Avete dubbi o domande? Riassunto delle puntate precedenti:

Stiamo iniziando a guardare come costruire Sistemi Operativi, mentre prima abbiamo visto come usarli, cosa sono. E stiamo guardando di capire alcuni particolari relativi all'architettura all'hardware. Nello schema che abbiamo visto un paio di lucidi prima, abbiamo visto che il sistema operativo è quello stato che crea un'astrazione sull'hardware. Quindi se volete costruire un Sistema Operativo dovete sapere come funziona l'hardware, qual'è il linguaggio parlato dall'hardware. Abbiamo visto dall'inizio di questo paio di lucidi: "cosa sono gli interrupt?", "Perché gli interrupt sono così importanti?", e che diventano veramente il motore che fa funzionare i sistemi operativi. Andiamo poi a vedere quelle che sono le risorse che il sistema operativo deve gestire.

Era stata arrivata qua (slide 23 01-arch-hw.pdf) se ricorda bene. Una risorsa molto importante, fondamentale è la RAM (Random Access Memory) per come sono costruiti i sistemi

oggi. La RAM è la memoria veloce, direttamente accessibile al processore volatile. Non sempre i sistemi erano fatti così, il primo sistema che ho usato nella mia vita aveva una memoria ai nuclei di ferrite, e quindi quella che era la memoria di lavoro era non volatile, era permanente. Mentre invece nei sistemi comuni oggi si tende a usare la RAM da RAM assieme ai registri è l'unico spazio di memorizzazione acceduto direttamente dal processore. Per accedere a memoria secondaria e terziaria ed altre risorse, non lo si fa direttamente parlando con il bus, ma bisogna tramite il bus dare dei comandi a dei controller che accedono al device. La RAM nei sistemi moderni è chiamata RAM statica, viene costruita con praticamente dei condensatori che si caricano, deve essere rinfrescato il contenuto perché tende a scemare con il tempo, ma tutta questa attività è invisibile al Kernel perché viene fatta direttamente dai moduli della RAM stessa. Quindi per quanto riguarda l'uso della RAM si accede direttamente al bus mettendo l'indirizzo a cui si vuole leggere e scrivere e si da il comando LOAD / STORE. Nei sistemi moderni, però, l'indirizzamento della RAM non viene attuato in maniera diretta ma tramite un'unità fisica, un componente fisico chiamato Memory Management Unit (MMU). Mi raccomando per le abbreviazioni e i concetti simili: Esiste una cosa chiamata Memory Mapper, che è una componente del sistema operativo e del software. Esiste una cosa chiamata Memory Management Unit che è una componente hardware ed è il componente che interfaccia la memoria.

Se volete il Memory Manager è il componente del sistema operativo che pilota, programma, configura la MMU. ok.  
Esistono ancora nei sistemi delle ROM (Read Only Memory), pensate a epram o cmos ram, alimentate, cioè memorie normalmente utilizzate in sola lettura. Tipicamente queste memorie vengono utilizzate per funzionalità molto basiliari, per esempio per fare il boot della macchina. Lo avete un po' visto anche quando avete visto un po' di architettura di pmp (microvmps). (pop 24)

Esistono alcuni dispositivi che sono MEMORY MAPPED, che si riescono ad accedere come se fossero memoria, ad esempio il video grafico dei Personal Computer. Ad alcuni indirizzi di bus corrispondono delle aree che non sono vere aree di memoria, ma dove il sistema può scrivere dei dati e questi dati vengono letti in tempo reale. È una memoria a doppio accesso: il bus scrive da un lato e dall'altro vengono letti da un componente, la scheda grafica, che prende questi valori scritti come valori normalmente come luminosità RGB (Red Green Blue, i tre colori) per accendere i vari pixel sullo schermo. Lo schermo grafico viene acceduto in questo modo, la gestione è semplice e lineare perché in questa modalità uno sa calcolare l'indirizzo di quel pixel, scrivere un valore e il valore diventa una luminosità di un pixel. Ma dall'altra parte necessita di sincronizzazioni di accesso. Essendoci due componenti che contemporaneamente accedono a questa memoria a doppio accesso, quello

che succede è che ci vogliono meccanismi di sincronizzazione  
talvolta fatti in maniera hardware, perché ad esempio  
il video non faccia FLICKERING (Visionario Oxford...  
Sostanzialmente, lo sfarfallio dell'immagine sullo schermo  
di un televisore o di un computer.). Quindi spesso c'è  
una doppia memoria, viene scritto il nuovo frame poi  
si dà l'impulso per fare in modo che il nuovo  
frame diventi quello visibile dall'utente. (pag 25)

DISCHI! Nonostante l'avvento, nonostante la presenza di  
dischi allo stato Solido (Solid State Drive  $\Rightarrow$  SSD) si usano  
tantissimo ancora i dischi rotazionali; i dischi convenzionali,  
quelli che abbiamo visto alla prima lezione di questo  
semestre direttamente aperti. I dischi sono dispositivi che memorizzano  
l'informazione in forma magnetica e quindi mantengono l'informazione  
in maniera non volatile, significa che anche se togliete l'alimentazione  
potete ritrovare i dati scritti. L'accesso è diretto (random i.e. non  
sequenziale). Per capire questa affermazione bisogna vedere che  
esistono anche dei dispositivi magnetici ad accesso sequenziale,  
che erano i nastri, i tape (erano ad accesso sequenziale).

L'indirizzamento su disco è dato in termini di 3 parametri:  
**cilindro, settore e testina**. Per cambiare cilindro, pensa lo  
abbiate visto anche ad architettura, quindi non sta a lungo in testa,  
c'è il pettine delle testine e quindi se voi non muovete il  
pettine delle testine il disco gira e quindi tutto ciò che è  
memorizzato con quella posizione delle testine è accedibile, vuoi  
aspettando che durante una rotazione transiti il settore sotto la

testina, vuoi decidendo di usare una delle testine fra quelle disponibili. Se invece uno vuole spostare il pettine delle testine, quindi cambiare cilindro si chiama cilindro perché ovviamente se tenete le testine ferme il disco gira tutti i dati saranno a una certa distanza dal punto di rotazione, quindi se lo pensate geometricamente è un cilindro. Per cambiare cilindro ci vuole un tempo meccanico di spostamento della testina. Per cambiare settore ci vuole un tempo meccanico per aspettare che quel dato che vogliamo leggere o scrivere passi sotto la testina. Per cambiare testina non ci vuole tempo meccanico basta tempo elettronico. Per intenderci per cambiare cilindro occorrono dei tempi dell'ordine di millisecondi e dipende anche da quanti cilindri dovrete saltare, per cambiare settore il tempo si può anche calcolare, perché quando comprate un disco il disco fa i parametri di lavoro a il numero di giri al secondo. Se avete un disco a basso costo oppure un disco che fa 5400 giri al minuto RPM (revolutions per minute) e se prendete un disco molto costoso ne fa 15000, non è che possiate avere un disco che fa 200000 giri al minuto, decollerelde il computer :D no, non ce la si fa fisicamente a costruirlo. Quindi se pensate 5400 giri... facciamo 600 per fare i conti pari se poi fosse 6000 giri al minuto al secondo sono 100, quindi vuol dire 10 millisecondi, quindi anche quello non è un tempo banale. ok?

Dal punto di vista del sistema operativo questo cosa significa? Perché non è indolare avere questa gestione. Significa che le operazioni di SEEK devono essere limitate e bisogna fare in modo

(pag 26)

che siano l'orari. Quindi quello che succede è che, visto che i vari processi chiederanno in concorrenza di voler leggere e scrivere dati su disco occorrerà avere anche in questo caso dei meccanismi di scheduling di ordinamento delle richieste che ottimizzino il percorso della testina su disco. OK? E' altra cosa, che incide più sull'implementazione del file-system rispetto alla gestione fisica del disco è quello di fare in modo che dati che vengono acceduti spesso, in sequenza e vicini nel tempo siano vicini anche su disco. Se voi utilizzerete dei file-system, e lo vedremo, poco efficienti, costruiti male, per esempio FAT (File Allocation Table), a forza di usarlo quel file system spargerà i "pezzi dei file" in posti casuali del disco. OK? È quindi farà in modo che il sistema man mano perda sempre di più di prestazioni. OK? Infatti sono quei file system per i quali è prevista deframmentazione. Non so se avete mai fatto defrag. ma potrete scegliere altri file system che non avevano questo tipo di problema. Mi vengono in mente 2 cose che vi faccio vedere per compiere il ritmo.

su google cerca "cybernare pezzo nel cyberspazio"  
clicca su "Trappola nel Cyberspazio - Roberto di Cosmo"  
link: <https://www.dicosmo.org/Piège/cyberespace/trappola.html>  
Roberto di Cosmo, nonostante il nome è italianoissimo, ma vive e lavora in Francia, insegnava direi a Paris Diderot, (leggi il capitolo intitolato "armadi a cassetti e la paga dei casselli") e racconta: "Ebbene durante uno dei miei viaggi mi sono ritrovato a fianco di un gentilissimo signore, giovane e dinamico

funzionario d'impresa, che si apprestava ad eseguire sulla sua macchina il famigerato programma DeFrug. Questo programma mostra una bella matrice riempita di piccoli quadrati di tanti colori che si muovono in tutte le direzioni mentre il disco lavora intensamente e rumorosamente.

Non ho potuto resistere alla tentazione (...): dopo essermi complimentato per il mio bel portatile, gli ho chiesto, fingendo la più grande ignoranza, [di un famosissimo professore di informatica] cosa fosse quel bellissimo programma che io non avevo sul mio portatile. Con un'aria di superiorità mista a compassione (...) mi ha risposto che si trattava di uno strumento essenziale che bisogna lanciare di tanto in tanto per "fare andare la macchina più veloce". Ha proseguito ripetendomi a memoria gli argomenti che si trovano nei manuali Windows: più si utilizza il disco, più questo si "frammenta", e più il disco è frammentato più la macchina è lenta.

A questo punto ho tirato fuori il mio computer portatile, che non utilizza Windows, ma GNU/Linux (...) e gli ho detto con un'aria un po' stupita, che tutto quello che mi aveva detto mi sorprendeva enormemente: sul mio portatile il disco è molto poco frammentato e più si utilizza meno si frammenta.

[...] Questo disco è come un gigantesco armadio a cassetti: ogni cassetto ha la stessa capienza e ciascun disco contiene alcuni milioni di cassetti [meno di quelli di oggi]. Se i dati che vi interessano sono sistemati in cassetti contigui, vi si può accedere più velocemente [...] .

Immaginiamo un ministero che conserva i suoi dossier in un enorme armadio con milioni di cassetti [ ... ].

Se avete un segretario [ lo metto al maschile perché lo trovo più coerente ... ] con due candidati dalle abitudini molto diverse uno che quando trova un dossier si limita a ruotare i cassetti e quando trova uno nuovo lo separa in piccoli fascicoli e li mette a caso nel primo cassetto vuoto che trova. Un altro che conserva sulla sua scrivania una lista di cassetti vuoti contigui e aggiorna la lista tutte le volte che la pratica viene chiusa. Questo lo trovo una maniera molto simpatica di raccontare la frammentazione.

L'altra cosa che mi volevo far vedere (vediamo se lo trovo) (cerca su google "Lost We Remember: Cold Boot Attacks on Encryption Keys" link: [https://www.usenix.org/legacy/event/sec08/tech/full\\_papers/haldeman.pdf](https://www.usenix.org/legacy/event/sec08/tech/full_papers/haldeman.pdf) oppure <https://cipsite.s3.amazonaws.com/xp-content/uploads/2019/01/23195456/haldeman.pdf>)

"Voi pensate che la RAM sia volatile, in realtà qui hanno fatto vedere un attacco su chiavi crittografiche fatte sulla RAM in che modo? Prendendo la RAM in funzione e facendo queste azioni velocissime, e poi neanche troppo veloci, tempo qualche secondo:

- Spegnete brutalmente la macchina senza che il sistema possa fare shutdown.
- Togliete la RAM e consigliatela

è ancora possibile leggere quello che c'è scritto sulla RAM.

Interessante... ecco (pag 6, Figura 4 dell' articolo)

Quello è l'immagine nella RAM e vedete il contenuto dopo  
• 5 secondi • 30 secondi • 60 secondi e • 5 minuti.

Quindi congelato, entro 5 secondi avete ancora tantissime informazioni  
utili per vedere (ovviamente era un'immagine ma l'hanno fatto  
vedere su chiavi critografiche)..., in fondo chiavi critografiche quando  
voi scrivete la password, è vero che nel sistema viene salvata in  
maniera critografata e così via, ma per fare l'input della  
password quella password per un po' passa in RAM, in chiaro."  
Comunque per vostra informazione non penso vi sia utile per fare  
Sistemi Operativi a prova di congelamento della RAM.

Quando si dice che la RAM è volatile è proprio perché  
è fatta con i condensatori, quindi i condensatori si scaricano  
pian piano, non c'è più il meccanismo di refresh perché non  
c'è più corrente, ma il contenuto può essere recuperato.

OK. (pag 27)

SSD. Ho aggiunto questo lucido perché anche SSD ha degli effetti  
collaterali sui sistemi operativi. Per copiare cosa dobbiamo fare con i sistemi  
operativi dobbiamo capire quali sono le caratteristiche alle quali dobbiamo stare attenti:  
di queste apparecchiature. Il numero di cicli di scrittura è limitato,  
alto ma limitato, quindi occorre per quanto possibile spargere  
uniformemente sulla gamma di indirizzi, cioè nello spazio  
memorizzabile le operazioni. Quindi è bene riciclare lo spazio  
riutilizzato in maniera da... Non importa come nei dischi  
cercarlo vicino, ma occorre fare in modo di riciclare e  
riutilizzare ora ciò che abbiamo utilizzato in tempo più remoto,

in maniera tale da sporgere l'accesso su tutti gli indirizzi.

Si leggono a blocchi. Si scrivono a blocchi (numerosi blocchi)

Il meccanismo di scrittura dell'SSD prevede che la scrittura non venga fatta per un blocco solo ma per un gruppo di blocchi. OK? Queste cose sono normalmente abbastanza mascherate dai controller, ma meglio si sanno queste cose, per esempio è bene fare in modo che cose che vengono aggiornate contestualmente siano a indirizzi limitrofi. Per il numero di cicli di scrittura limitato vedremo che ci sono degli effetti collaterali anche non solo nella gestione fisica del dispositivo ma anche sul file system, per esempio un caso tipico d'uso è quello di evitare di fare aggiornamenti non necessari, grosse cache aiutano, ma tra l'altro nei sistemi UNIX, per esempio, è possibile, quando si monta un file system, dichiarare se si vuole avere la data di ultimo accesso al file o no.

In realtà la data di ultimo accesso ad un file, accesso non modifica, se togliete la data di modifica saltano un sacco di cose per esempio make non funziona più. Mentre invece la data di ultima lettura è rocambole utile, quindi di solito se avete dei dischi SSD si fa in modo che non venga aggiornata, c'è il parametro di mount che si chiama `noatime` `no access time`, che fa in modo che non ci sia questa informazione aggiornata. Pensate "Quanti file leggete?" "Quanti file scrivete?" "Quante volte leggete un file?", pensate `/etc/passwd` tutte le volte che c'è un autenticazione da qualche parte o si vuole vedere qual'è il nome corrispondente a un utente c'è un accesso a quel file.

Quanto più velocemente degrada un disco allo stato solido (SSD) se chiedete questo aggiornamento? Perché ovviamente i dati di controllo continuano a essere aggiornati ad ogni accesso, e quasi si tenta di fare cache degli inode usati però... (pag 28)

Quindi possiamo pensare di avere una gerarchia di memoria, abbiamo tante memorie, abbiamo registri, RAM, dischi e anche unità offline.

Andrew Tanenbaum ha detto

"Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway"

e il prof lo ha tradotto con:

"Non sottovalutate mai il bandwidth  
di un vagone ferroviario carico di nastri"

google traduttore lo ha tradotto con:

"Non sottovalutare mai la larghezza di banda di  
una station wagon piena di nastri che sfreccia  
lungo l'autostrada".

In realtà il bandwidth può essere notevolissimo, è difficile accederlo... E cosa c'entra con questo? Di solito le memorie vengono chiamate memoria primaria, secondaria, terziaria. E prima ancora della memoria primaria ci sono i registri. La memoria primaria è quella direttamente accessibile dal processore, quindi normalmente la RAM. La memoria secondaria è la memoria accessibile tramite un controller dal processore quindi i dischi rigidi e i SSD. La memoria terziaria è memoria offline che può essere collegata in necessità, la memoria terziaria normalmente ha bisogno di un intervento umano o

simili per farlo.

bot: che cosa intende per cicli di scrittura?

Per la SSD, quando dovete aggiornare il dato che avete memorizzato sulla SSD dovete dare un comando di scrittura e quindi quello è un ciclo di scrittura, il ciclo di scrittura fisico.

Come dicevo prima normalmente per farne il meno possibile si tengono dei buffer, quindi se un dato un piccolo file viene acceduto, modificato più volte in realtà non ci fa alcuna scrittura fisica, se non in fondo se il file non viene anche cancellato. OK? Quindi l'idea è quante scritture fate, lo chiamo ciclo di scrittura e non scrittura perché è diversa la scrittura che appare a voi quando scrivete un programma fate una write e la scrittura fisica che viene fatta poi veramente sulla SSD. Come le chiavette USB, che infatti sono fatte più o meno della stessa pasta, se mai provate a usarla pesantemente in lettura e scrittura e la staccate senza fare l'operazione di chiusura di umount su quella chiavetta ci può essere di tutto, perché tenderà a fare meno scritture possibili fisiche e molte cose quando l'andate a scollegare pensavate di averle fatte ma erano ancora nel buffer.

Allora abbiamo una gerarchia di memoria (pag. 29, figura)

Partendo dai registri guardiamo una riga sì e una riga no.

registri, memoria principale che è la RAM; memoria secondaria Dischi SSD; memoria terziaria. Poi c'è una linea di demarcazione tra la memoria volatile e quella non volatile, ma oltre gli elementi dispari nel mezzo ci sono gli elementi per cui sono le CACHES. L'ultima parte l'ho un po' inventata per

domandarmi che cos'è oggi l'equivalente delle memorie terziarie.  
In fondo oggi come memorie terziarie quello che si usa più normalmente sono memorie allo stato solido removibili, mentre una volta c'erano nastri, CD rom, DVD rom che hanno perduto sempre più utilizzo perché risultano più costosi, più scomodi di unità allo stato solido. Parliamo del concetto di cache importantissima.

Che cos'è uno CACHE?

Una cache è una porzione di memoria più veloce che viene usata per mantenere temporaneamente i dati più utili di una memoria più lenta. Facendo così statisticamente potrò capitare frequentemente di poter ritrovare il dato senza andare ad accedere alla memoria più lenta, ma trovandolo già disponibile nella memoria più veloce. OK? Questo è un concetto astratto che potete applicare a tutti i livelli quindi per esempio nelle CPU nei processori ci sono delle cache del contenuto della memoria, se guardate quando acquistate un processore o acquistate una macchina con dentro un processore, nelle caratteristiche tecniche del processore c'è scritto quanta cache c'è. OK? Questa cache che vedete là su (in figura n. 23) tra i registri e la memoria centrale è costruita con le tecnologie dei registri quindi molto veloce e questa cache è gestita direttamente dal processore quindi normalmente è trasparente ai sistemi operativi. Quindi il sistema operativo / il programma applicativo non si accorge della cache se non dal punto di vista prestazionale, quando si accede alla memoria centrale automaticamente la cache fa il suo mestiere e tenta di mantenere i dati più recentemente usati. Tutti questi meccanismi si basano su un principio di... viene spesso detto in informatica di località, se volette è l'equivalente informativo dell'inerzia. Ci si aspetta che dati limitrofi

in memoria, logicamente limitrofi in memoria e dati recentemente utilizzati vengano utilizzati nel prossimo futuro, che non è una regola generale. È possibile costruire dei programmi per fare in modo che la cache lavori peggio possibile, però statisticamente le cose vanno bene. Non è totalmente indolare fare una cache, non è così banale, infatti è stato un trappismo ingegneristico non banale, perché per esempio se avete un sistema multiprocessore, ognuno con la sua cache, occorrono dei meccanismi, che i processori implementano, per invalidare la cache degli altri processori se uno modifica un dato nella cache, perché se no risulta disallineato... Una delle garanzie che deve dare una cache è che sia trasparente, che non si veda, che nessun programma abbia risultati diversi se eseguito in un sistema con la cache o senza.

E quindi siccome la semantica di accesso, l'astrazione nell'accesso alla memoria centrale è una semantica immediata, cioè mi aspetto che se un programma scrive a quella locazione un dato dal momento in cui è stato fatto l'operazione tutti gli altri possono leggere lo stesso dato, occorre che la cache garantisca questa cosa, e per farlo deve invalidare i contenuti delle cache degli altri processori.

Come c'è una cache tra la memoria centrale e i registri, si può fare una cache dei contenuti dei dischi nella RAM. Ancora una volta pensiamo la RAM ha velocità un milione di volte superiore a quella del disco. Se andate a comprare della RAM, le caratteristiche tecniche dicono tempi di accesso 8, 10, 10 è già scarsissima, 8 nanosecondi: ( $1 \text{ nanosecondo} = 1 \text{ ns} = 1 \cdot 10^{-9} \text{ s}$ ) NANO  $\Rightarrow$  miliardesimi, milli ( $10^{-3}$ ) micro ( $10^{-6}$ ), nano ( $10^{-9}$ ), pico ( $10^{-12}$ ), fento ( $10^{-15}$ ), atto ( $10^{-18}$ ). Prima allora visto i dischi millisecondi sono millesimi:

quindi: 2 miliardi diviso 1000 togliete 3 zeri 1 milione.

Ordini di grandezza, ma pensate che così il rapporto di un milione. Pensate che cosa succede in 1 secondo e che cosa succede in un milione di secondi OK?

Allora se si riescono a tenere i dati più utili nel prossimo futuro, si riescono a tenere in RAM e non su disco, il sistema avrà un aumento di prestazioni spaventoso. OK?

Ecco questa cache, in realtà, nonostante il concetto sia lo stesso, questa deve venire implementata dal sistema operativo, non è una cache hardware, è una cache software. ovviamente si può pensare di fare la stessa cosa ad alto livello, per esempio se una delle unità in cloud, che vanno benissimo però hanno una latenza notevole, perché quando andate ad accedere alla rete, non so che latenza abbiate voi nelle vostre connessioni di rete, ma se vi va bene sono almeno 10-15 millisecondi di "RAM trick?" giusto per mandare il dato lì e tornare. Quindi può essere conveniente creare delle cache locali in RAM o anche su dischi locali del materiale che c'è in giro.

Addirittura vedremo ci sono dei file system sperimentali, tipo Andrew File System, che sono file system i cui dati sono sparsi per il pianeta, potete avere server che vi formano un unico file system sparsi, e lì addirittura c'è la cache dell'intero file, e lì hanno dovuto cambiare la semantica d'accesso perché se no era inusabile.

Comunque l'importante è che vi sia chiaro il concetto di CACHE.

(pag 30) Ok, questo più o meno l'ho detto.

(pag 31) Problemi da considerare. Tutte le volte che abbiamo una cache c'è l'algoritmo di replacement.

Dato la caratteristica della memoria, quella più veloce e più accessibile è in quantità minore ed è più costosa e man mano si va a scemare. Quella disponibile a tempi più elevati è normalmente più lenta, enormemente meno costosa e disponibile in quantità molto più ampia. Quindi, concetto chiaro, la cache ha una dimensione standard più piccola dell'unità che andiamo a ottimizzare. All'inizio è facile, tutto ciò a cui accediamo lo mettiamo anche nella cache, ma poi a un certo punto la cache si riempie e occorrerà capire "chi butta giù dalla torre?" quale dato presumibilmente non sarà utile nell'immediato futuro e quindi si potrà pensare di scaricarlo dalla cache, e questi sono i così detti ALGORITMI di REPLACEMENT, (di rimpiazzo, per cambiare). OK? E a questo punto vedremo meccanismi per varie funzionalità del Sistema Operativo. Occorre creare un CURISTICO che abbia senso, non c'è "la" soluzione. Allora per esempio uno può pensare di cercare la pagina accessuta meno recentemente (la pagina ... l'elemento di cache che è lì da più tempo e che non è stato accedito, quindi si presuppone che non sia utile) oppure si può pensare di guardare la FREQUENZA di accesso (quell'elemento della cache è stato accedito frequentemente quindi nel futuro si pensa che...)

vedete è un "processo" di NERZIA, si tenta di vedere qual'è il comportamento nel prossimo futuro analizzando il comportamento nel passato prossimo, alliamo parlarne delle previsioni del tempo, il concetto è lo stesso. Quindi l'algoritmo di replacement decide qual'è l'elemento da eliminare nella cache quando bisogna aggiungerne un altro, ovviamente l'elemento

che deve essere eliminato deve essere aggiornato nella memoria reale se sono avvenute delle operazioni di modifica.

Come si sceglie l'algoritmo di replacement?

C'è un trade-off fra quello più statisticamente performante e le strutture dati che bisogna mantenere per poter fare funzionare quell'dato. Che ne so, se uno vuole tenere traccia mano mano di quelli meno meno recentemente utilizzati deve tenere una struttura dati da aggiornare ad ogni accesso per fare una lista per vedere quale è il più utilizzato e quello meno, se uno deve tenere la frequenza deve tenere per ogni elemento un contatore che ad ogni accesso venga incrementato.

L'altro problema è la **coerenza**, siccome la cache è una copia, è una copia in una memoria più veloce, quindi quel dato che logicamente è lo stesso dato, è presente in più replicazioni in punti diversi della gerarchia di memoria.

Dove essere coerente. Come dicevo prima, deve fare in modo che la cache alla fine sia trasparente, esserci o non esserci il risultato non cambia se non il tempo per produrre il risultato.

(pag 32) Ancora l'hardware. I meccanismi di protezione.

I processi in esecuzione su un sistema operativo non devono interferire tra loro, devono poter comunicare e ... . Quindi una cosa sono i servizi forniti dal Sistema Operativo per operazioni ordinarie, decite, una cosa è se accidentalmente o volontariamente un processo può riuscire a perturbare/disturbare/rovinare.

l'esecuzione di un altro processo o addirittura del sistema operativo stesso. Come si fa ad evitare che ci siano queste interazioni non volute? Sicuramente non è possibile farlo totalmente via software, occorre una mano dell'hardware. Studente: Questo perché potrebbe fallire via software, quindi allora bisogna di un "ultima spiaggia"?

Occorre creare una barriera "cosa può fare il processo e cosa no?" se un processo potesse direttamente accedere all'unità di I/O potrebbe rovinare il lavoro degli altri, se un processo potesse accedere direttamente alla memoria degli altri potrebbe rovinarne il funzionamento. Esiste del codice che deve poter accedere alla memoria degli altri, il Kernel, quindi quello che dobbiamo chiedere al sistema è che si dia il modo di riconoscere gli umani dagli dei, di riconoscere i processi che possono fare poche cose, i processi che possono essere cattivi, possono essere ... sono quelli da controllare e che non devono produrre danni e i controlleri, cioè la parte di codice che è affidabile e controlla il resto delle cose. Quindi il punto chiave che la cosiddetta "modalità protetta", quando hanno inventato i Personal Computer, con il 6502, penso ad Apple 2, oppure con i 8086, 8088 Intel, penso ai PC IBM, questi erano processori che non avevano modalità protetta. Cos'è la modalità protetta? Il processore ha un bit (pag 33) (o simili, prendiamo 1 bit, poi vi spiego che in Intel c'è una cosa leggermente diversa) comunque ha 1 bit, se quel bit è modalità Kernel il codice che è in esclusione può fare tutto quello che vuole e

ha pieno accesso al sistema, se invece il bit è spento in modalità user, in modalità user il codice, il processore è configurato in maniera tale che ci sono solo le operazioni NON privilegiate in funzione. Quando un processo è in modalità Kernel può volontariamente fare tutto, tra le varie operazioni, può anche dire "adesso vado in modalità user". Un processo in modalità user può fare solo quello che gli è concesso e l'unica cosa che può succedere è che arrivi un interrupt o una trap e nella gestione dell'interrupt o della trap, non per volontà diretta del processo, o meglio non con codice sotto controllo del processo ... il processo può chiedere System call, però a quel punto il resto della gestione viene fatta dal Kernel. Vi dico, i primi PC avevano dei processori che non avevano queste funzionalità e quindi non era fisicamente possibile scrivere sistemi operativi affidabili, sicuri. A seconda dell'autore o del processore troverete la modalità Kernel indicata come modalità privilegiata, modalità supervisor, ring 0. Ecco gli Intel hanno questa idea, 4 livelli di priorità, ring 0, ring 1, ring 2, ring 3, di fatto ring 0 è la modalità Kernel e le altre sono modalità non privilegiate a scolare. Normalmente i Sistemi Operativi che ho visto su Intel usano 0 e 3 come ring. Per esempio le istruzioni per disabilitare gli interrupt è privilegiata. Un processo che è in esecuzione in modalità user non può disabilitare gli interrupt, se disabilitasse gli interrupt diventerebbe monopolista della CPU; non può cambiare le mappe di memoria della MMU. In realtà nei

sistemi moderni l'accesso alla memoria è non privilegiato perché è la configurazione della MMU che è privilegiata. Quindi il processo accede liberamente alla memoria, ma siccome gli indirizzi che genera sono indirizzi logici e la MMU glieli traduce, lui può generare tutti gli indirizzi che vuole, quelli che la MMU non li consente non li vedrà mai. ok? E per cambiare la mappa di configurazione della MMU ci vogliono operazioni privilegiate, quindi lui non le può fare, lo può fare solo il Kernel. (pag 34)

Alla partenza il processore è in modalità Kernel.

[... domanda studente e docente risposta... (min 52:50-53:45) ...].

Al bootstrap (quando si inizia) il sistema è in modalità Kernel. Quindi normalmente il sistema parte, fa l'inizializzazione, crea a mano lo stato del primo processo, chiama lo scheduler, lo scheduler di CPU è chiamato a dire "adesso qual'è il prossimo processo che va avanti", al boot non ha molte scelte c'è n'è solo uno, guardacaso sceglierà quello, e a quel punto cede il controllo al primo processo, e siccome lui può farlo cede il controllo e lo mette in modalità user. Il primo processo allora inizia ad eseguire, a fare il codice che deve fare, è in modalità user, non può fare altro che fare calcoli ed accedere alla memoria, e a quel punto delle due uno:

1. o accade un Interrupt 2. o accade una trap.

A quel punto quando accade l'interrupt o la trap, l'hardware passa da modalità utente a modalità Kernel ad eseguire il codice Kernel, e il gioco continua, viene gestito l'interrupt o la trap, si chiama lo scheduler,

si fa partire un processo o quello o un altro in modalità user e continua così il ciclo del sistema operativo. (pag 35)  
Questo è il meccanismo su cui si basano tutte le altre protezioni. Accedere direttamente ai device in modalità user è vietato, uno lo deve chiedere tramite una system call di fare l'operazione. Quindi il fatto che il Kernel funzioni in modalità privilegiata e controlli tutto, fa in modo che il Kernel possa, senza poter ricevere interferenze dal processo utente, decidere quello che si può e quello che non si può fare. Quando arriva una system call che dice "voglio accedere a quel file", dice "no, non è tuo e non hai il permesso", ma non c'è nulla sul disco, non c'è nulla hardware che faccia la protezione a livello del file-system, semplicemente è protetta la system call e la system call viene eseguita in modalità Kernel e il codice del Kernel va poi a vedere con tabelle sue, sue strutture dati quello che si può o non si può fare e risponde "picche" o risponde "ok" alla richiesta di operazioni. Questo è il nucleo centrale di protezione, mi raccomando modo Kernel e modo user sono modi del processore, non c'entra niente anche se c'è scritto supervisor, modo supervisor è una cosa del processore. Superuser ROOT sono cose del filesystem, sono utenti generati dal sistema, sono astrazioni che vi dà il sistema operativo. OK? Anche un processo di root è in modalità user, è un processo come gli altri, solo che la systemcall open va a vedere chi è l'utente (che

è un concetto implementato dal sistema operativo)  
"chi è l'utente che fa questa operazione?" dice "ah è root,  
allora a root è concesso". Se un processo che è in  
esecuzione come root vuole accedere a /dev/sda1  
(direttamente al disco) il processo è in modalità  
user in realtà non accede direttamente al disco. Il  
processo fa una system call open, la system call viene  
eseguita in modalità Kernel, dice "Può accedere in scrittura  
a /dev/sda1?" "sì" allora da l'OK alla open e va  
ad agire sul file. Ma il processo di root funziona  
in modalità user. (pag 36) ok, per l'ACCESSO alla  
MEMORIA la protezione, come si diceva, avviene attraverso  
la Memory Management Unit (MMU).

Perché c'è questo meccanismo? Se i processi potessero  
accedere alla memoria all'indirizzo che vogliono  
potrebbero modificare il codice, o gli altri degli processi,  
modificare i dati e il codice del sistema operativo,  
modificare l'interrupt vector e mettere il proprio  
gestore dell'interrupt. Comunque, come si diceva, la MMU  
fa la traduzione degli indirizzi logici in indirizzi fisici,  
è una funzione che ha il suo Dominio e il suo Codominio.  
Se voi applicate una funzione un dato potrà generare solo  
risoluzioni / valori nel Codominio. Se la memoria vietata non  
appartiene al Codominio potete dare tutti i dati che volete  
alla funzione e non succederà mai nulla. (pag 37) ok.

(pag 38) Le istruzioni di I/O sono privilegiate. Come fa un

processo utente a fare operazioni privilegiate? Chiama le **System call**, ovvero sono troppe generate da istruzioni specifiche. Nei vecchi sistemi Intel proprio il processo corrente generava un interrupt, direi 0x80, cioè aveva l'istruzione per fare un interrupt come quello dell'hardware. Nei moderni sistemi c'è una chiamata apposta, chiamata **SYSTEM CALL**, che risulta essere più efficiente. (pag 39, figura) Quindi: la systemcall si comporta come la gestione degli interrupt che abbiamo visto prima, software ma è un interrupt, quindi quando fa la syscall si salvano i registri, si gestisce, si fa l'operazione, si ripristinano i registri e si fa continuare il processo, oppure se è bloccante fra il ripristino servizi e il ritorno dell'interrupt viene scelto un altro processo se c'è di monda avanti.

Ecco mi fa venire in mente una cosa che vi dico prima della pausa, perché questo è l'ultimo lucido di questo parco. Probabilmente cambia appena non è lucido, ecco ho pensato che aggiornere questo lucido queste scrive qua a sinistra, invece che scrivere perché gli interrupt possono essere interrupt del hardware o del software. Qui mettendo queste parole fa confusione.

(pag 6)

Al posto delle parole hardware metterò processore, al posto delle parole software scrivete codice. Gli interrupt possono essere sia dei dispositivi sia per errori o systemcall. Per ciascuno di questi tipi c'è una parte di elaborazione fatta dal processore (in alto) e una parte che viene fatta dal codice. OK così non c'è sbruffamento. pausa (registrazioni misurate 1:03:45).