Programmazione di sistema

Iniziato	venerdì, 14 maggio 2021, 20:03
Stato	Completato
Terminato	venerdì, 14 maggio 2021, 22:10
Tempo impiegato	2 ore 6 min.
In ritardo	6 min. 4 secondi
Valutazione	**0,00** su un massimo di 15,00 (**0**%)

Domanda 1

Risposta non data

Punteggio max.: 3,00

SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE
LE RISPOSTE SI/NO VANNO MOTIVATE

Sia dato un sistema di memoria virtuale con paginazione, nel quale vengono indirizzati i Byte. Il sistema dispone di TLB (Translation Look-aside Buffer). La tabella delle pagine ("page-table") viene realizzata con uno schema a due livelli, nel quale un indirizzo logico di 64 bit viene suddiviso (da MSB a LSB) in 3 parti: p1, p2 e d, p2 ha 14 bit e d 13 bit. Non si utilizzano ulteriori strutture dati (quali tabelle di hash o inverted page table) per velocizzare gli accessi. La memoria virtuale viene gestita con paginazione a richiesta.

Si risponda alle seguenti domande:

A) Supponendo che la memoria RAM abbia tempo di accesso di 160 ns, si calcoli il TLB miss ratio (frequenza di miss), tale da garantire effective access time EAT_{PT} <= 200 ns, assumendo che il tempo di accesso alla TLB sia trascurabile.
Si tratta di un lower bound o di un upper bound per la TLB miss ratio?

B) Si consideri ora la frequenza di page fault p_{PF}. Una valutazione sperimentale mostra che un page fault viene servito in 4 ms in media, e che il degrado di prestazioni (aumento di tempo di esecuzione) causato dai page fault è compreso tra 10% e 20%. Calcolare l'intervallo di valori di p_{PF} compatibile con la valutazione sperimentale.

C) Una seconda valutazione sperimentale (una simulazione) viene fatta usando due stringhe di riferimento w_1, w_2, aventi lunghezza, rispettivamente, $len(w_1)=10^6$, $len(w_2)=2*10^6$ e probabilità di stringa p_1 e p_2. Le simulazioni hanno generato 500 page fault in totale (100 su w_1 e 400 su w_2) e stimato una probabilità empirica f = 0.00013 (frequenza attesa) che avvenga un page fault nel sistema reale. Si calcolino i valori di p_1 e p_2.

LE RISPOSTE SONO FORMULATE IN INGLESE IN QUANTO COMUNI AL CORSO IN INGLESE DI SYSTEM AND DEVICE PROGRAMMING

A) TLB miss ratio = ...

upper or lower bound?

T_{RAM} = 160 ns (RAM access time)
EAT_{PT} = 200 ns
$x = miss_{TLB} = 1 - h_{TLB}$
2 level (hierarchical) PT => 2 reads for PT lookup

$EAT_{PT} = h_{TLB}*T_{RAM} + (1-h_{TLB})*3T_{RAM} = (1 + 2*(1-h_{TLB}))T_{RAM} = (1 + 2x)\ T_{RAM}$
$2x = EAT_{PT}/T_{RAM} - 1 = 1/4$
$x = 1/8 = 0.125$
Upper bound as a higher miss ratio increases EAT_{PT}

B) Range of values for p_{PF}

T_{PF} = 4 ms (PF service time)

$p_{PF}*T_{PF}$ = increase in time due to page faults
$0.1 < p_{PF}*T_P / EAT_{PT} < 0.2$
$0.1 < p_{PF}*(4/ 200)*10^6 < 0.2$
$0.1 < 2*10^4 p_{PF} < 0.2$
$5*10^{-6} < p_{PF} < 10^{-5}$
$p_{min} = 5*10^{-6}$
$p_{Max} = 10^{-5}$

C) Calculation of string probabilities p_1 and p_2

F_1=100, F2=400 (F_1+F_2 = 500)
p1 + p2 = 1 (we just have two strings)
$f = p_1*F_1/len(w_1) + p_2*F_2/len(w_2)$

$1.3*10^{-4} = p_1*100/10^6 + p_2*400/(2*10^6)$
$p_1 = 1 - p_2$

$(1-p_2)*10^{-4} + 2p_2*10^{-4} = 1.3*10^{-4}$
$1+p_2 = 1.3$
$p_2 = 0.3$
$p_1 = 0.7$

Domanda 2

Risposta non data

Punteggio max.: 3,00

SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE
LE RISPOSTE SI/NO VANNO MOTIVATE

Un disco è organizzato in blocchi fisici e logici di una data (stessa) dimensione BS. Il disco contiene più partizioni: la partizione A è formattata con un file system che alloca staticamente NM=12.5K blocchi per i metadati (direttori, file control block e una bitmap, per la gestione dei blocchi liberi), e ND=100M blocchi per i dati dei file.

Si risponda alle domande seguenti:

A) Calcolare la dimensione del blocco BS, considerando che NM/4 blocchi sono riservati alla bitmap, che contiene un bit per ognuno degli ND blocchi di dato.

B) Si sa che un file control block (FCB) ha dimensione 512B e NM/2 dei blocchi per metadati sono riservati al FCB. Calcolare il massimo numero di file che possono essere memorizzati nel file system.

C) Si supponga che la bitmap indichi un rapporto del 50% tra blocchi liberi e usati (free/used, cioè 1 blocco libero ogni 2 usati, "usato" significa "allocato"), e che esattamente 5M ($M = 2^{20}$) blocchi liberi siano "isolati" (preceduti e seguiti da un blocco occupato/allocato). Si calcoli la massima dimensione di un intervallo di blocchi liberi contigui, assumendo la configurazione di bitmap più favorevole. Si ripeta la risposta supponendo di essere nella situazione meno favorevole.

LE RISPOSTE SONO FORMULATE IN INGLESE IN QUANTO COMUNI AL CORSO IN INGLESE DI SYSTEM AND DEVICE PROGRAMMING

A) compute the block size BS, considering that NM/4 blocks are reserved to the bitmap, that has one bit for each of the ND data blocks.

The bitmap size can be expressed both given the total mount of bits (ND) and the number of blocks reserved to the bitmap (NM/4)
|bitmap| = ND/8 Bytes = NM/4 * BS
BS = ND/(2*NM) Bytes = 100M/25K Bytes = 4KBytes

B) We know that a file control block (FCB) has size 512B and NM/2 metadata blocks are reserved to FCBs. Compute the maximum number of files that can be stored in the file system.

The maximum number of files corresponds to a full occupancy of the FCB reserved space

$NF_{Max} * |FCB| = NM/2$

Let's express |FCB| in terms of blocks (we could work on Bytes as well): |FCB| = 1/8 Block

$NF_{Max} = NM/(2*|FCB|) = 12.5K/(2/8) = 4*12.5K = 50K$

C) Compute the size of the largest free block interval |LargestFree| in the most favourable and in the least favourable scenarios (sizes have to be expressed as numbers of blocks):

the bitmap has ND bits, corresponding to ND blocks
ND/3 blocks are free (1 free, 2 used means 1 free every 3 blocks), 5M free blocks are "isolated"

Most favourable:
All non "isolated free blocks are contiguous, so the largest interval on contiguous free blocks is given by
|LargestFree| = ND/3 - 1M blocks.
ND = 100M
|LargestFree| = 33.3M - 5M blocks = 28.3M blocks

Least favourable
The least favourable situation is when all non "isolated" free blocks are grouped by couples of 2 contiguous free blocks interleaved with allocated blocks.
|LargestFree| = 2 blocks

Domanda 3

Risposta non data

Punteggio max.: 3,00

SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE
LE RISPOSTE SI/NO VANNO MOTIVATE

Si considerino le ottimizzazioni implementate per migliorare le prestazioni complessive della gestione della memoria virtuale.

A) Spiegare brevemente la tecnica COW (Copy on Write): che cosa è e come migliora le prestazioni della gestione della memoria? I miglioramenti sono in termini di tempo? Di occupazione di memoria? Di entrambi tempo e memoria? (Motivare)

B) Spiegare perché l'IO sulla partizione di swap (backing store) è più veloce dell'IO mediante file system (anche se a parità di disco). E' possibile che lo spazio di indirizzamento virtuale di un processo sia più grande della dimensione dell'area di swap (partizione di swap o swapfile)? (Motivare)

C) Quale è il ruolo del bit di modifica (dirty bit) associato a una pagina? Il dirty bit (bit di modifica) viene memorizzato solo nella tabella delle pagine oppure viene anche immagazzinato nella TLB (quando disponibile su un dato processore)? (Motivare)

LE RISPOSTE SONO FORMULATE IN INGLESE IN QUANTO COMUNI AL CORSO IN INGLESE DI SYSTEM AND DEVICE PROGRAMMING

A) Briefly explain the COW (Copy on Write) technique: what is it and how does it improve memory management performance? Is the expected gain/improvement on time? On memory size? On both time and memory? (motivate)

- COW is an optimisation allowing two processes (parent and child) to share their pages upon child creation. Pages are duplicated (so they become two independent pages) when either process (parent or child) modifies them by writing.
- Performance is improved both on time and memory: time, because the process creation is faster as pages are initially shared (part of the time is simply deferred to a later writing, but just for written pages), memory, because new pages are generated just if/when written

B) Explain why the IO on a swap partition (backing store) is faster than file system IO (even if on the same device). Can the virtual address space of a given process be larger than the size of the swap area (size of the swap partition or file)? (motivate)

It is faster because the swap space is allocated in larger chunks and less management is needed than file system.
Yes it can be larger, as the swap space just need to provide storage for the used pages, not foe the unused ones (think for instance of the possibly unused pages in the "middle" of the address space.

C) What is the role of the modify (dirty) bit associated to a page? Is the modify (dirty) bit just stored in the page table or is it also stored in the TLB (when available in a given processor)?

The modify bit keeps track of the state of a page with respect of its copy on disk, or with respect to a given time interval of observation.
Though the bit is stored in each page table entry, storing it also within each TLB entry allows best performance (no PT access is due with TLB hit), but requires hardware support, in order to set the bit when writing the page.

Domanda 4

Risposta non data

Punteggio max.: 3,00

SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE
LE RISPOSTE SI/NO VANNO MOTIVATE

Si considerino tre kernel thread OS161 che realizzano un lavoro di trasferimento dati di tipo produttore/consumatore (1 produttore, 2 consumatori). I thread condividono un buffer per i dati, implementato come un struct C di tipo `struct prodConsBuf` definita come segue:

```
struct prodConsBuf {
  data_t data;
  int dataReady;
  struct lock *pc_lk;
  struct cv *pc_cv;
};
```

Quando non ci sono dati presenti nel buffer, il flag dataReady vale 0.
Quando il produttore scrive un nuovo dato (puntato da dp) nel buffer (campo data), pone il flag dataReady a 1 e invia un segnale a entrambi i consumatori (il lavoro viene svolto dalla funzione producerWrite).

```
void producerWrite(struct prodConsBuf *pc, data_t *dp);
```

I consumatori chiamano iterativamente la funzione consumerRead.

```
void consumerRead(struct prodConsBuf *pc, data_t *dp);
```

Quando i consumatori ricevono un segnale, solo uno dei due può trovare il flag dataReady al valore 1. Quindi legge il dato (copiandolo dal buffer alla memoria puntata da dp), azzera il flag, infine ritorna/esce dalla funzione consumerRead. Il consumatore che trova il flag dataReady al valore 0 non fa nulla e si rimette in attesa di un altro segnale.
Il lock viene usato per mutua esclusione. La condition variable viene usata per le segnalazioni produttore-consumatore. Si consideri unicamente la sincronizzazione da produttore a consumatore: NON SI RICHIEDE CHE IL PRODUTTORE VERIFICHI CHE UN MESSAGGIO SIA LETTO CORRETTAMENTE DA UN CONSUMATORE. Si può quindi assumere che il produttore effettui una nuova chiamata a `producerWrite` dopo che il precedente dato è stato acquisito correttamente dai consumatori.

Rispondere alle domande seguenti:

A) E' possibile che il buffer condiviso sia collocato nello stack del thread, oppure deve essere una variabile globale o altro?

B) Siccome i campi `pc_lk` e `pc_cv` sono puntatori, dove andrebbero chiamate le funzioni `lock_create()` and `cv_create()`? Nel thread produttore, nel consumatore? Da qualche altra parte?

C) Si scriva una possibile implementazione delle funzioni `producerWrite` e `consumerRead`. Non sono richieste altre parti di codice del produttore e/o del consumatore.

LE RISPOSTE SONO FORMULATE IN INGLESE IN QUANTO COMUNI AL CORSO IN INGLESE DI SYSTEM AND DEVICE PROGRAMMING

A) Can the shared structure be located into the thread stack, or should it be a global variable or other?

> Each thread has its own thread stack, so a shared data cannot reside there. A global variable is the usual option. Other options include dynamic allocation (by kmalloc), if properly handled.

B) As the `pc_lk` and `pc_cv` fields are pointers, where should the `lock_create()` and `cv_create()` be called? In the producer thread, in the consumer threads? Elsewhere?

> It could be done in each of them, provided that they properly synchronize: e.g. one thread does initializations, the other threads wait.
> But a more common solution could be that initializations are done by another thread, the parent/master thread, who is creating the producer and the consumers.

C) Provide an implementation of functions `producerWrite` and `consumerRead`. No other producer and/or consumer code is needed, just the functions.

```
/* error checking/handling instructions are omitted for
simplicity */

void producerWrite(struct prodConsBuf *pc, data_t *dp) {
  /* lock for mutual exclusion */
  lock_acquire(pc->pc_lk);
  /* copy data: done here by simple assignment -
     a proper function could be used instead */
  pc->data = *dp;
  pc->dataReady = 1; /* set flag */
  /* signal all consumers */
  cv_broadcast(pc->pc_cv,pc->pc_lk);
  /* just release the lock and return */
  lock_release(pc->pc_lk);
}

void consumerRead(struct prodConsBuf *pc, data_t *dp) {
  /* lock for mutual exclusion */
  lock_acquire(pc->pc_lk);
  while (!pc->dataReady) {
    cv_wait(pc->pc_cv, pc->pc_lk);
  }
  /* copy data: done here by simple assignment -
     a proper function could be used instead */
  *dp = pc->data;
  pc->dataReady = 0; /* reset set flag */
  /* just release the lock and return */
  lock_release(pc->pc_lk);
}
```

Domanda 5

Risposta non data

Punteggio max.: 3,00

SE I RISULTATI SONO NUMERI, RIPORTARE PASSAGGI INTERMEDI RILEVANTI E/O FORMULE USATE
LE RISPOSTE SI/NO VANNO MOTIVATE

Si consideri in OS161 una possibile implementazione delle system call relative al file system, basate su una tabella di processo e su una di sistema, definite come segue

```
/* system open file table */
struct openfile {
  struct vnode *vn;
  off_t offset;
  unsigned int countRef;
};
/* this is a global variable */
struct openfile systemFileTable[SYSTEM_OPEN_MAX];
...
/* user open file table: this a field of struct proc */
struct openfile *fileTable[OPEN_MAX];
```

Si risponda alle domande seguenti (motivando le risposte):

A) Perché `systemFileTable` è una variabile globale, mentre `fileTable` è un campo di `struct proc` ?

B) Si supponga che due processi user chiamino open() sullo stesso file. Ci saranno due entry (righe) in `systemFileTable` per quel file, oppure una sola entry (puntata, quindi condivisa, da entrambi i processi) ?

C) Si vuole implementare il supporto per la system call SYS_lseek. Si riporta la descrizione delle funzione lseek:

> *`off_t lseek(int fd, off_t offset, int whence);`*
>
> *`lseek()` repositions the file offset of the open file description associated with the file descriptor `fd` to the argument offset according to the directive `whence` as follows:*
>
> *SEEK_SET*
>
> > *The file offset is set to offset bytes.*
>
> *SEEK_CUR*
>
> > *The file offset is set to its current location plus offset bytes.*
>
> *SEEK_END*
>
> > *The file offset is set to the size of the file plus offset bytes.*

Quale/i struttura/e dati sarà/saranno interessata/e dall'implementazione della system call SYS_lseek?

1. la process file table?
2. una entry della systemFileTable (per ogni chiamata di lseek)

3. più di una entry della systemFileTable (per una chiamata di lseek)
4. il vnode del file

(sono possibili selezioni multiple, per ogni selezione occorre dire cosa si fa sulla relativa struttura dati)

LE RISPOSTE SONO FORMULATE IN INGLESE IN QUANTO COMUNI AL CORSO IN INGLESE DI SYSTEM AND DEVICE PROGRAMMING

A) Why is `systemFileTable` a global variable, whereas `fileTable` is a field of `struct proc` ?

> `systemFileTable` a global variable because it is unique and shared by all processes, whereas we have one `fileTable` for each process, so we can directly allocate it within the `struct proc`.

B) Suppose two user processes call open() for the same file, will there be two entries in the `systemFileTable` for the file, or just one entry (pointed to, thus shared, by the two processes) ?

> There will be two entries, because each of them needs a different offset within the file (two processes read/write at different locations within the file), so a single entry cannot be shared. A single `struct openfile` (an entry in the `systemFileTable`) can be shared only as a result of operations such as the dup and dup2 system calls..

C) We need to implement the support for the SYS_lseek system call. Which data structures will be affected by the implementation of the SYS_lseek system call?

> 1. the process file table? NO
> 2. one entry of the systemFileTable (for a given call to lseek) ? YES
> 3. multiple entries of the systemFileTable (for a given call to lseek) ? NO
> 4. the vnode of the file? NO
>
> Just one openfile entry within the systemFileTable will be modified in order to update its offset field, as the lseek operation just modifies the offset for one process (same as with read/write operations.