Programmazione di sistema

Iniziato	sabato, 8 maggio 2021, 18:07
Stato	Completato
Terminato	sabato, 8 maggio 2021, 18:48
Tempo impiegato	41 min. 37 secondi
Valutazione	**0,00** su un massimo di 14,00 (**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), su cui si misura sperimentalmente un “hit ratio” del 98%. 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*, rispettivamente di 40 bit, 12 bit e 12 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 300 ns, si calcoli il tempo effettivo di accesso (EAT) per il caso proposto (TLB hit ratio = 98%), assumendo che il tempo di accesso alla TLB sia trascurabile.

B) Si consideri ora la frequenza di page fault p. Ipotizzando che un page fault sia servito in 5 ms, quanto deve valere p affinché si possa garantire un degrade massimo del 20% per EAT (causato sia della TLB che dei page fault)? Si tratta di un valore massimo o minimo per p?

C) Al fine di valutare un algoritmo A di sostituzione pagine, si sono simulate/provate tre stringhe di riferimento w_1, w_2, w_3, aventi lunghezza, rispettivamente, $len(w_1)=10^6$, $len(w_2)=2*10^6$, $len(w_3)=5*10^6$. Le simulazioni hanno generato, rispettivamente, 200, 100 e 50 page fault. Le tre stringhe hanno probabilità (di rappresentare una generica esecuzione nel sistema reale) $p_1=0.5$, $p_2=0.2$, $p_3=0.3$. Si calcoli la probabilità empirica f (frequenza attesa) di un page fault nel sistema reale.

A) Calcolo EAT_{PT} con TLB hit ratio = 98%:

T_{RAM} = 300 ns (RAM access time)
h_{TLB} = 0.98 (TLB hit ratio)
2 level (hierarchical) PT => 2 reads for PT lookup

$EAT_{PT} = h_{TLB}*T_{RAM} + (1-h_{TLB})*3T_{RAM} = (1 + 2*(1-h_{TLB}))T_{RAM} = 1.04 * T_{RAM} = 312$ ns

B) Calcolo P per garantire un degrado massimo del 20% per EAT.
Si tratta di un valore massimo o minimo per p?

T_{PF} = 5 ms (PF service time)

$EAT_{PF} = (1-p)*EAT_{PT} + p*T_{PF} = (1-p)*312 + p*5*10^6$ ns
$EAT_{PF} \leq 1.2 * T_{RAM} = 360$ ns
$(1-p)*312 + p*5*10^6 \leq 360$
$(5*10^6 - 312)*p \leq 360 - 312$
$(5*10^6)*p \leq 360 - 312$ (removed negligible term)
$p \leq 48/5*10^{-6} = 9.8*10^{-6} \approx 10^{-5}$

Valor minimo o massimo (dire perché) ? Valor massimo, vista la disequazione, oppure osservando che al crescere della probabilità di Page Fault aumenta EATPF, che deve essere inferiore e un limite massimo

C) Calcolo la probabilita empirica f di un page fault nel sistema reale, date stringhe w_1, w_2, w_3, $len(w_1)=10^6$, $len(w_2)=2*10^6$, $len(w_3)=5*10^6$, probabilità $p_1=0.5$, $p_2=0.2$, $p_3=0.3$ e numeri di page fault 200, 100, 50:

F_1=200, F2=100, F_3=50

$f = p_1*F_1/len(w_1) + p_2*F_2/len(w_2) + p_3*F_3/len(w_3) =$
$= 0.5*200/10^6 + 0.2*100/(2*10^6) + 0.3*50/(5*10^6) =$
$= (100 + 10 + 3)*10^{-6} = 1.13 * 10^{-4}$

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 file eseguibile contiene, in sequenza:

- un header di dimensione 5KB,
- un segmento di testo (programma e altro) di dimensione 10.5MB,
- un segmento dati (globali) di dimensione 9 MB.

Il file viene salvato in un file system F basato su inode standard (10 puntatori diretti, 1 indiretto singolo, 1 doppio, 1 triplo), basati su blocchi di dimensione 4KB e indici (puntatori a blocchi) da 32bit.

Si risponda alle domande seguenti:

A) E' necessario che il segmento di testo e il segmento dati inizino all'inizio di un blocco?

B) Quale è la dimensione (indipendente dal file system) del file e quale è la sua occupazione (misurata in blocchi) nel file system F?

C) Calcolare la frammentazione interna (per i soli blocchi di dato)

D) Quanti blocchi di indice sono necessari per rappresentare il file? (dire anche a quali livelli appartengono, se indiretto singolo, doppio e/o triplo)

E) Supponendo di mantenere invariati l'header e il segmento di testo, quanto dovrebbe essere grande il segmento dati per rendere necessario il livello indiretto triplo?

A) E' necessario che il segmento di testo e il segmento dati inizino all'inizio di un blocco?

NO. Perché il contenuto del file è indipendente dal file system e dal disco in cui viene memorizzato. Il formato del file eseguibile viene trattato a un livello più alto rispetto all'implementazione del file system e all'IO su disco.

B) Quale è la dimensione (indipendente dal file system) del file e quale è la sua occupazione (misurata in blocchi) nel file system F?

|file| = |header|+|text segment| + |data segment| =
= 5KB+10.5MB+9MB = 5 + 19.5*1024 KB = 5+20480-512 KB = 19973 KB = 19.5 MB

occupazione = ceil(19973 KB / 4KB) = 4994 blocchi (ceil(): intero superiore)e

C) Calcolare la frammentazione interna (per i soli blocchi di dato)

framm = 4994*4KB – 19973KB = 19976 – 19973 = 3KB (3/4 dell'ultimo blocco)
Si noti che la frammentazione interna potrebbe essere calcolata trascurando i due segmenti tex/data in quanto multipli di un blocco

D) Quanti blocchi di indice sono necessari per rappresentare il file? (dire anche a quali livelli appartengono, tre indiretto singolo, doppio e/o triplo)

Un blocco contiene fino a 4KB/4B = 1K = 1024 indici
ND: numero blocchi dato
NI: numero blocchi indice

ND_0 = 10 direct data blocks
NI_0 = 0 (no index blocks)

ND_1 = 1K = 1024 single indirect data blocks
NI_1 = 1 (1 index block)

ND_2 = 4994 – (10 + 1024) = 3960 double indirect data blocks
NI_2 = 1 outer (first level) index block + ceil(3960/1024) = 4 inner (second level) index blocks = 5 index blocks

NI_{tot} = NI1 + NI2 = 1 + 5 = 6

E) Supponendo di mantenere invariati l'header e il segment di testo, quanto dovrebbe essere grande il segmento dati per rendere necessario il livello indiretto triplo?

$|file|_{3,min} = 1B + |file|_{2,MAX} = 1B + ND_{2,MAX}*4KB =$
$= 1B + (10 + 1K + 1M)*4KB = 4GB + 4MB + 40KB + 1B$
$|data\ segment|_{min} = file|_{3,min} - |header! - |text\ segment| =$
$= 4GB + 4MB + 40KB + 1B – 5B – 10.5\ MB \approx 4GB – 6.5MB =$
4089.5MB

Domanda 3

Risposta non data

Punteggio max.: 2,00

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

Un processo user vuole effettuare un input da un dispositivo di IO a caratteri (character device) mediante una strategia basata su polling del dispositivo. Per evitare un'attesa troppo lunga (nel loop di polling) da parte del driver del dispositivo, si chiede all'autore del programma eseguito dal processo user, di fare direttamente polling mediante un loop di lettura del registro di stato del dispositivo.

A) E' possibile effettuare questa operazione? (se si, dire come, se no, dire perché)

B) Supponendo che il dispositivo sia associato alla tastiera e lo si gestisca in interrupt, ci si aspetta che venga generato un interrupt per ogni carattere ricevuto oppure un interrupt per ogni "invio" (cioè enter, fine-riga)? (motivare)

Dato un altro dispositivo di IO, gestito a blocchi:

C) E' possibile che una sola lettura (mediante *read()*), tenti di acquisire un vettore di dimensione più grande di un blocco?

D) Dato il driver che realizza la lettura, tale operazione sarà realizzata mediante DMA (qualora possibile) oppure no? Perché?

A) E' possibile effettuare questa operazione? (se si, dire come, se no, dire perché)

NO. Perché un dispositivo di IO richiede istruzioni privilegiate, con la CPU in modalità kernel. Un processo user non ha accesso al dispositivo e non ne può leggere il registro di stato, a mano che sia implementata una opportuna system call in grado di fornire tale servizio.

B) Supponendo che il dispositivo sia associato alla tastiera e lo si gestisca in interrupt, ci si aspetta che venga generato un interrupt per ogni carattere ricevuto oppure un interrupt per ogni "invio" (cioè enter, fine-riga)? (motivare)

Ci si attende un interrupt per ogni carattere. La tastiera è un dispositivo a caratteri. Una eventuale bufferizzazione dei caratteri in una riga, tale da poter effettuare correzioni prima dell'invio, va gestita via software (a meno di avere una tastiera speciale con queste caratteristiche)

Dato un altro dispositivo di IO, gestito a blocchi,

C) è possibile che una sola lettura (mediante read()), tenti di leggere dal dispositivo un vettore di dimensione più grande di un blocco?

SI: E' possibile, né il formato di un file né la dimensione di un blocco possono vincolare l'operazione read(). Se il vettore è più grande si un blocco, verrannoletti più blocchi.

D) Dato il driver che realizza la lettura, ci si aspetta, che la realizzi mediante DMA (qualora possibile) oppure no? Perchè?

SI: Se è disponibile un DMA controller, l'acquisizione mediante DMA è più efficiente perché consente alla CPU di lavorare in parallelo e perché in DMA si fanno complessivamente meno transiti di dati sul bus (la metà rispetto all'IO programmato.

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

Sia dato un sistema OS161. Si supponga di aver aggiunto a `kern/conf/conf.kern` le istruzioni

```
defoption exam
optfile project syscall/exam.c
```

e di aver creato in `kern/conf` il file EXAM, copiato dal file `DUMBVM`.

A) Si dica se sono sufficienti le azioni appena descritte, più l'esecuzione, in `kern/conf` di `./config EXAM`, affinché venga compilato il file opzionale `syscall/exam.c` quando si effettuano, in `kern/compile/EXAM`, i comandi

```
bmake depend
bmake
```

B) Quale, tra i file `exam.h` e `opt-exam.h`, viene generato automaticamente dal comando `./config EXAM`? Viene generato sempre oppure, oppure solo se in `EXAM` compare l'istruzione

```
options exam
```

C) Cosa contiene il file (quello generato automaticamente, di cui alla domanda precedente)?

D) Si supponga di inserire, nel file main.c l'istruzione

```
exam_init();
```

Tenendo conto che la funzione `exam_init()` viene realizzata nel file `syscall/exam.c`, come è possibile fare in modo che l'istruzione sia presa in considerazione e compilata solamente nelle versioni del kernel in cui viene abilitata l'opzione exam?

A) Si dica se sono sufficienti le azioni appena descritte, più l'esecuzione, in `kern/conf` di `./config EXAM`, affinché venga compilato il file opzionale `syscall/exam.c` quando si effettuano, in `kern/compile/EXAM`, i comandi

```
bmake depend
bmake
```

NO. Non sono sufficienti. Le istruzioni definiscono l'opzione e la dipendenza del file syscall/exam.c dall'opzione. Ma occorre abilitare/attivare tale opzione, nel file EXAM. Diversamente il file exam.c non solo non verrà compilato, ma neppure considerato nella generazione delle dipendenze.

B) Quale, tra i file `exam.h` e `opt-exam.h`, viene generato automaticamente dal comando `./config EXAM`? Viene generato sempre oppure, oppure solo se in `EXAM` compare l'istruzione

```
options exam
```

> `opt-exam.h`. Il file viene generato anche se EXAM non contiene `options exam`. Affinchè il file .h venga generato, è sufficiente che `conf.kern` contenga la definizione dell'opzione exam. Un eventuale file `exam.h` può, se necessario/opportuno, essere creato "manualmente", e va aggiunto, come opzionale oppure no, a seconda dei casi, in `conf.kern`.

C) Cosa contiene il file (quello generato automaticamente, di cui alla domanda precedente)?

> `opt-exam.h`, a parte la protezione dall'inclusione multipla, contiene una sola direttiva al pre-compilatore:
> `#define OPT_EXAM 1`
> oppure
> `#define OPT_EXAM 0`
> in funzione del fatto che EXAM contenga o meno l'istruzione `options exam`

D) Si supponga di inserire, nel file `main.c` l'istruzione

```
exam_init();
```

Tenendo conto che la funzione `exam_init()` viene realizzata nel file `syscall/exam.c`, come è possibile fare in modo che l'istruzione sia presa in considerazione e compilata solamente nelle versioni del kernel in cui viene abilitata l'opzione exam?

> Occorre utilizzare la compilazione condizionale, sfruttando la macro OPT_EXAM, definita in opt-exam.h. Quindi l'intestazione del file `main.c` (oppure uno dei .h da esso inclusi) dovrà contenere
> `#include "opt-exam.h"`
> e l'istruzione da condizionare (in `main.c`) andrà scritta come
>
> ```
> #if OPT_EXAM
> exam_init();
> #endif
> ```

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

Sia dato un sistema OS161, nella versione base. Si considerino le definizioni delle `struct thread` e `struct proc` parzialmente qui riportate:

```
struct thread {
  char *t_name;
  ...
  void *t_stack;
  struct switchframe *t_context;
  struct cpu *t_cpu;
  struct proc *t_proc;
  /* add more here as needed */
};

struct proc {
  ...
  unsigned p_numthreads;
  struct addrspace *p_addrspace;
  struct vnode *p_cwd;
  /* add more here as needed */
};
```

Si risponda alle domande che seguono:

A) Si dica come sono correlate/collegate le due struct nel caso in cui un processo abbia più di un thread.

B) Si dica anche se, dato un puntatore `t` a una `struct thread`, è possibile ricavare l'elenco di tutti gli altri thread dello stesso processo e stamparne il nome. Qualora non sia possibile, si proponga come modificare (aggiungere campi) a `struct proc` e/o `struct thread`, in modo da poter realizzare l'operazione (dato un processo, elencare i suoi thread). Scrivere la funzione `printOtherThreadNames(struct thread *t)` che svolge tale compito.

C) Si consideri la `struct proc`. Si spieghi perché, nonostante il campo `p_addrspace` permetta di raggiungere il puntatore allo stack di un processo user, anche la `struct thread` contiene un puntatore a stack. Si tratta di un'informazione ridondante (c'è un solo stack a cui puntano direttamente il thread e indirettamente il processo), oppure sono due stack diversi?

D) Si spieghi brevemente il ruolo del puntatore `t_context` nella `struct thread`.

A) Si dica come sono correlate/collegate le due struct nel caso in cui un processo abbia più di un thread.

La `struct thread` punta (campo `t_proc`) alla `struct proc` del processo. La `t_proc` non contiene puntatori ai thread del processo, ma contiene solo il conteggio di tali thread (campo `p_numthreads`).

B) Si dica anche se, dato un puntatore `t` a una `struct thread`, è possibile ricavare l'elenco di tutti gli altri thread dello stesso processo e stamparne il nome. Qualora non sia possibile, si proponga come modificare (aggiungere campi) a `struct proc` e/o `struct thread`, in modo da poter realizzare l'operazione (dato un processo, elencare i suoi thread). Scrivere la funzione `printOtherThreadNames(struct thread *t)` che svolge tale compito

Nella versione attuale l'operazione non è possibile in quanto una `struct proc` non punta ne a un vettore ne a una lista di puntatori ai relativi thread. Si esclude, perché considerata costosa, la possibilità di iterare su tutti i thread attivi nel kernel (a patto di poterlo fare) testando per ognuno se punti al processo dato.

Un modo per consentire l'operazione potrebbe essere l'aggiunta alla `struct proc` di un puntatore alla lista dei suoi thread,

```
struct proc {
  …
 /* add more here as needed */

  struct thread *p_threadListHead;
}
```

aggiungendo alla `struct thread` un campo che funga da link/next per tale lista

```
struct thread {
  ...
  /* add more here as needed */
  struct thread *t_threadListNext;
}

printOtherThreadNames(struct thread *t) {
  struct proc *p=t->t_proc;
  struct thread t1;
  /* no error check: assume t_name is not NULL */
  for (t1=p->p_threadListHead; t1!=NULL; t1=t1-
>t_threadListNext)
    if (t1!=t) kprintf(t->t_name);
}
```

C) Si consideri la `struct proc`. Si spieghi perché, nonostante il campo `p_addrspace` permetta di raggiungere il puntatore allo stack di un processo user, anche la struct thread contiene un puntatore a stack. Si tratta di un'informazione ridondante (c'e' un solo stack a cui puntano direttamente il thread e indirettamente il processo), oppure sono due stack diversi?

> Si tratta di due stack diversi: quello puntato dall'addrspace è lo stack user del processo (attualmente creato nell'ambito della load_elf(), quello puntato dal campo `t_stack` della `struct thread` è lo stack kernel, usato, ad esempio, per salvare switchframe e trapframe.

D) Si spieghi brevemente il ruolo del puntatore `t_context` nella `struct thread`.

> Punta allo switchframe del thread, che permette di salvare/ripristinare il contesto del thread ad ogni context switch (cambio del thread in esecuzione sulla CPU). Si vedano ad esempio le funzioni `thread_switch()` e `switchframe_switch()`.