COMPUTER SYSTEM ENGINEERING, SPRING'13

MID-TERM EXAMINATION

Problem-1: Client/Server (24')

SJTU-park

Chenny is in charge of designing an online system, SJTU-park, which helps to manage the parking spots inside the SJTU campus. Chenny decides to use server/client system. She gets a machine to run the SJTU-park server. Multiple SJTU-park clients communicate with the server concurrently by the remote procedure calls (RPC). For each RPC call, server will fork a new thread to handle it.

Chenny designs a Chenny's RPC protocol that:

1. On the client side, we deal with RPC requests one by one. Every time a RPC request is sent or retransmitted, we increase and assign the new sequence number to it. And client will ignore the response if its sequence number does not consist with current sequence number on the client side.

2. On the server side, we maintain a *c_seq* number for each client connection. *C_seq* records the maximal sequence number ever seen from the same client connection. Server will ignore client request with sequence number less than or equal to *c_seq.*

A simple pseudo code is as bellow:

```
procedure send_rpc( rpc_req ){
   seq_num ← seq_num+1
   send request with seq_num
   only received response consist with seq_num
   while(response timeout)
        seq_num ← seq_num+1
       retry sending
}
```

```
procedure receive_rpc(connection_id, req, seq_num){
   if(seq_num > c_seq(connection_id) ) then
        sequence(connection_id) ←seq_num
       deal with request and response with seq_num
   else   ignore the request
}
```

Since clients are installed on the cars, the network may drop, delay, duplicate or reorder

messages.

Chenny's initial design is as bellow:

Server:

```
procedure order_spot( client_id ){
   while TRUE
      lock( access_lock )
        for i ← 0 to NSPOTS do
           if(available[i] = -1) then
              available[i] ← client_id
              start_time[i] ← get_time_now
              unlock( access_lock )
              return i        // clinet gets spot i
      unlock( access_lock )
}
```

```
procedure get_booked( client_id ){
      for i ← 0 to NSPOTS do
           if(available[i] = client_id) then
                start_time[i] ← get_time_now
                return i        // client gets spot i
      return -1        // cannot find a booked spot
}
procedure leave_spot( spot ){
      cost ← caculate( get_time_now - start_time[i] )
      available[spot]← -1
}
```

Client:

```
procedure c_get_spot( client_id ){
        spot ← rpc( get_booked(client_id) )
        if(spot < 0)
              spot ←
rpc( order_spot(client_id) )
}
```

```
procedure c_leave_spot( spot ){
         rpc( leave_spot(spot) )
}
```

*Note:

1. Leaving a spot must happen after ordering a spot.

2. Client_id is an identical id for every car calculated from car number.

3. There is external booking system updating information of the booked parking spots into the *available* array. The update is ensured to be atomic.

Questions:

1. Does Chenny's RPC protocol work successfully? What problem will it cause? Show one

example. (4’)

2. Try to fix the problem caused by Chenny’s RPC protocol, in different ways:

a) make modification in Chenny’s protocol as little as possible. (4’)

b) make modification of Chenny’s server/client implementation. (Tips: Modifying the arguments of these procedures is allowed. We assume the corresponding arguments on client side will be modified as well.) (4’)

3. Chenny fixed the problem in Chenny’s protocol. She finds when several clients concurrently request for parking spots, there is a long latency when calling “rpc(order_spot(client_id))”. Which way in question 2 may be used by Chenny to fix the problem? Why? (4’)

4. Chenny wants to optimize the latency problem with RPC chain. Try to modify the client function *c_get_spot(client_id)*, implement the new server function *get_spot(client_id)*, and draw the new call path. Show the benefit and the new problems of SJTU-park system after applying the optimization. (4’)

5. Suppose SJTU-park client will transparently re-connect to server after crash. What may happen if SJTU-park client crash? Design a possible solution to make SJTU-park client stateless and recover from crash without disturbing the SJTU-park server. (4’)

Problem-2: Concurrent Programming (24’)

Question 1:

Alice and Bob want to raise goldfish. As we all know, goldfish will eat everything until nothing left, and it is likely to be stuffed to death. So Alice and Bob have to feed their goldfish once and exactly once every day. Alice and Bob decide to use ***flag*** as a lock to synchronize raising fish.

Below are the different strategies. Are these strategies right? If not, please give an interleaving case to elucidate its problem. (6’)

```
noFeed = 1;
void feed_fish() { noFeed = 0;}
```

Strategy 1:

Alice:

```
if (!flag){
        flag = 1;
        if (noFeed)
            feed_fish();
}
```

Bob:

```
if (!flag){
        flag = 1;
        if (noFeed)
            feed_fish();
}
```

*Note: the variable ***flag*** is initialized to 0.

Strategy 2:

Alice:

```
flagOfAlice = 1;
if (!flagOfBob){
        if (noFeed)
            feed_fish();
}
flagOfAlice = 0;
```

Bob:

```
flagOfBob = 1;
if (!flagOfAlice){
        if (noFeed)
            feed_fish();
}
flagOfBob = 0;
```

Strategy 3:

Alice:

```
while (noFeed){
   flagOfAlice = 1;
   if (!flagOfBob){
         if (noFeed)
              feed_fish();
   }
   flagOfAlice = 0;
}
```

Bob:

```
while (noFeed){
   flagOfBob = 1;
   if (!flagOfAlice){
         if (noFeed)
              feed_fish();
   }
   flagOfBob = 0;
}
```

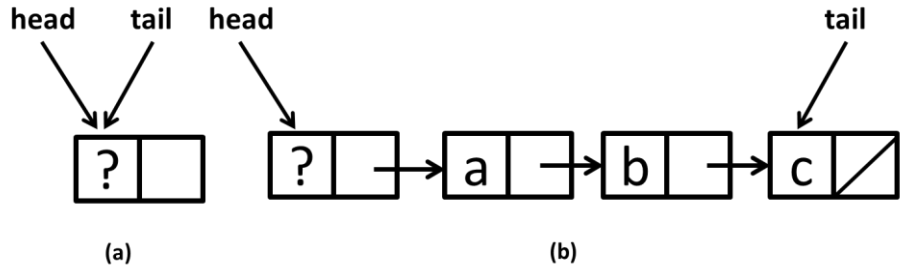
Question 2:

In our class, we learn how to implement bounded buffer with the lock, now we need implement a **First-In-First-Out** Queue with atomic instruction **C**ompare-**A**nd-**S**wap. This queue should support multiple producers and multiple consumers, and it should be implemented by **C**ompare-**A**nd-**S**wap (**CAS**) atomic instruction instead of lock.

For CAS, it means that it compares the contents of a memory location with a given value and, only if they are the same, modifies the contents of that memory location to a given new value; this is done as a single atomic operation. In this problem, function ***CAS*** wraps the CAS instruction, you can call it to do the atomic Compare-And-Swap, and the return value indicates whether the operation is successful or not.

Detailed information: for *enqueue* operation, the new element should be inserted at the tail of the queue, for *dequeue* operation, the element at the head of the queue should be removed. For simplicity, we have declared the queue's ***head*** and ***tail*** for you. ***head*** points to the first node in the list, which is a dummy useless node; the remaining nodes contain the values in the queue. ***tail*** points to the last node in the list. For issuing *dequeue* operation on an empty queue, you can simply exit (using **exit(-1)**).

Figure (a) is an empty queue, and Figure (b) is the queue with three elements (a, b, c).



For the queue in Figure (b), dequeue should remove the element of a, enqueue(d) should append element d after element c. Please implement enqueue and dequeue by using CAS(), as shown below. In this problem, it is not required to free memory, for simplicity. (6')

```
struct Node {
   int value;
   struct Node* next;
};
bool CAS(Node* * addr, node * oldVal, node * newVal)

Node *head=new Node, *tail=head;

void enqueue(int x){
      // your code here
}

int dequeue( ){
      // your code here
}
```

Question-3:

The definition of data race is: 1) multiple concurrent threads access a shared variable, 2) at least one access is write, 3) the threads use no explicit mechanism to prevent the accesses from being simultaneous. For each of the following code snippets, please answer two questions below: (12')

1. Is there any data race in it?
2. Does it work well?

A.

```
static MyObj *obj = NULL;
void InitObj() {
  if (!obj)
    obj = new MyObj();
}
void Thread1() {
  InitObj();
}
void Thread2() {
  InitObj();
}
```

B.

```
bool inited = false;
void Init() {
  // May be called by multiple threads.
  if (!inited) {
    mu.Lock();
    if (!inited) {
      // .. initialize something
    }
    inited = true;
    mu.Unlock();
  }
```

C.

```
/* Ref() and Unref() may be called from several threads. Last Unref() destroys the object. */
class RefCountedObject {
 ...
 public:
  void Ref() {
   mu.Lock(); ref_++;     mu.Unlock();
  }
  void Unref() {
   mu.Lock(); ref_--;     mu.Unlock();
    if (ref_ == 0)
      delete this;
  }
 private:
   mutex_lock mu;   //lock
  int ref_;
};
```

D

```
bool done = false;

void Thread1() {
  while (!done) {
    do_something_useful_in_a_loop_1();
  }
  do_thread1_cleanup();
}

void Thread2() {
  do_something_useful_2();
  done = true;
  do_thread2_cleanup();
}
```

Problem-3: Virtualization (24’)

1. Ben is reading “A Comparison of Software and Hardware Techniques for x86 Virtualization”. He got confused about some points. Please help Ben mark each of the following statements True or False. (4’)

True/False: The VMM uses binary translation (BT) to insert instructions before every guest kernel memory-referencing instruction to detect whether the instruction modifies a PTE (Page Table Entry).

True/False: The VMM write-protects the pages that make up the guest’s page tables.

True/False: The VMM only needs to provide one shadow page table for one guest VM.

True/False: The shadow page table is identical to the guest kernel’s page table, except that some PTE’s present or writable bits differ.

2. Ben knows that in a traditional OS (without virtualization), each process has a separated address space, which is identified by page table base pointer (CR3 in X86). During context switch, the OS loads the new page table base pointer into CR3 by issuing instruction “movl xxx, %%cr3”. However, in virtualized environment, such instruction will cause a trap to the VMM, which will then do the instruction for the guest. The process is named “trap-and-emulation”. Please describe the detailed procedure of how the VMM emulates this instruction. (Hint: the procedure starts from the guest VM, and also ends with the guest VM.) (3’)

3. Ben is learning the shadow page mechanism in virtualization. Considering the following case: a guest OS is now executing a single process, whose page table is shown below on the left. **The page table is stored in guest physical page 4.** The VMM (virtual machine monitor) uses the table shown on the right to translate guest physical pages to host physical pages. The VMM generates two page tables for each process: one is for the kernel and one for the user process. Please explain why it is needed to use two shadow page tables? (3’)

Note: P is for Present, W is for Writable, U is for User/Kernel (1 is for user).

Virt Page	Phys Page	P	W	U
0	5	1	1	1
1	9	0	1	1
2	11	1	0	1
3	4	1	1	0
4	7	1	1	0

Guest Phys Page	Host Phys page
4	6
5	2
6	4
7	3
11	8

4. Please help Ben fill in the two shadow page tables below that the VMM should use when running the VM. You can use a dash (-) to indicate values that don't matter. (6')

User shadow PT, stored in host phys page12 Kernel shadow PT, stored in host phys page13

Virt Page	Phys Page	P	W	U
0	2			
1				
2	8			
3				
4				

Virt Page	Phys Page	P	W	U
0	2			
1				
2	8			
3				
4				

5. In Ben's company, a physical machine usually runs tens of guest VMs with identical OS. Ben finds that a lot of memory pages, either within a VM or between different VMs, have same contents. For example, many pages just contain all zeros. Please help him add a new feature to the VMM to save physical memory, and describe your design. (4')

6. Ben uses online-banking for payment. He has a MacBook-Air, but the online-bank requires Windows XP, so Ben installs a Windows XP in a VM using VMware workstation. However, since Windows XP has a lot of security vulnerabilities, Ben is worrying that his bank account and password may be stolen by some malware (e.g., Trojans). He's wondering: what if I enhance the VMM to protect the IE browser (for online-banking) from the OS, which may be infected by malware? Please help Ben to achieve this goal and describe your design. (Hint: the key point is to ensure that the OS cannot access application's data. You can assume that all the I/O communication is safe with encryption). (4')

Problem-4: Performance (20’)

Since the Bob’s company grows bigger and bigger, his boss decided to add a platform to share software and media files to save the outgoing bandwidth. The boss asks Bob to set up a file server to storage and server file for members. After several days’ designing, Bob decides to following *“The simpler, the better”* rule. Here comes the pseudo code for his implementation:

```
procedure Service ():
    while true:
        request = Receive_Request ()
        if request.type == “GET”:
            file = Get_File_From_Disk(request.fileID)
            Reply(file)
        else if request.type==”PUT”:
            file = Get_File_From_Request(request)
            Put_File_To_Disk(request.fileID,file)
            Reply(OK)
```

Bob sets up the service on a spare PC with one disk and one network interface card (1 Gbps). There is no other recourse consuming process on this PC. The disk has an average seek time of 5 milliseconds, a complete rotation takes 6 milliseconds and its throughput is 40 MB/s for write and 80 MB/s for read at full speed (when no seeks are required).

Assuming that all files are 1 GB and the LAN bandwidth of Bob’s company is 1Gbps. The file system of this PC has no cache and allocates data for a file in 4K blocks. The blocks are allocated randomly, which means disk blocks of the same file can be all over the disk.

1. How long does a user take to put a file to the server? And how long to get a file? (Assuming there is only one user using the service) (4’)

The performance hurts Bob deeply and deeply. He analyses the time cost for each part and finds out the bottleneck is the disk. He decides to add a 1-GB cache in memory for the disk.

```
Cache[1GB]
procedure Service ():
    while true:
        request = Receive_Request ()
        if request.type == “GET”:
            file = Get_From_Cache(request.fileID)
            if file==NULL:
                file = Get_File_From_Disk(request.fileID)
                Put_To_Cache(request.fileID,file)
            Reply(file)
        else if request.type==”PUT”:
            file = Get_File_From_Request(request)
            Put_To_Cache(request.fileID,file)
            Reply(OK)
```

2. Assuming getting from cache or putting to cache take 0 milliseconds, how long does a user take to put a file to the server now? And how long does it take to get the same file exactly after putting? (4’)

The result makes Bob very happy, so he asks some friends to use the service for testing. After his friends begin putting and getting files, Bob feels hurting again: the service is not responding for some users, and some users fail to get what they’ve put to the server.

3. What's the most likely reason(s) for these problems? Please explain. (4')

In order to solve these problems, Bob changes his code to use multi-threads and makes some changes.

```
Cache[4GB]
for i from 1 to 30 do create_thread(Service)
    procedure Service ():
        while true:
            request = Receive_Request ()
            if request.type == "GET":
                file = Get_From_Cache(request.fileID)
                if file==NULL:
                    file = Get_File_From_Disk(request.fileID)
                    Put_To_Cache(request.fileID,file)
                Reply(file)
            else if request.type=="PUT":
                file = Get_File_From_Request(request)
                Put_To_Cache(request.fileID,file)
                Reply(OK)
                Put_File_To_Disk(request.fileID,file)
```

4. One friend of Bob thinks he should use locks to protected cache, but Bob doesn't think so. Do you agree with Bob? Please give your reasons. (4')

5. Still, Bob doesn't satisfy with the performance, can you give him some more advices? (4')

Problem 5: Survey (8')

Please answer the following questions about the course. Your opinions are of great importance for us to improve the course. Thanks!

1. Which aspect do you like most in this course? (2')
2. Which aspect do you dislike most in this course? (2')
3. Please list your suggestions for improving the recitation, the more the better. Thanks. (4')

COMPUTER SYSTEM ENGINEERING, SPRING'13

MID-TERM EXAMINATION

Name_____ Student No. _____ Score_____