Computer System Engineering Spring 2012

Midterm Examination

Problem 1: Naming (12')

In the evening of May 19th 2009, China Telecom(中国电信)'s networks in 6 provinces were almost halted. Later investigation indicates that the whole incident happened as follows:

- On May 18th, at around 10pm, DNS service provider DNSPod's servers were targeted with a DDoS (distributed denial of service) attack.
- China Telecom detected the abnormal traffic and blocked access to the IP address of DNSPod servers, to prevent them from draining resources from the machine rooms they occupied.
- On May 19th, at around 9pm, China Telecom's DNS servers started to receive massive numbers of name resolution requests of baofeng.com, and this caused traffic jam on the Internet.
- These requests were automatically sent by *Baofeng Player*. And DNSPod is the only DNS service provider responsible for resolving this domain.
- Later on May 19th, China Telecom's networks started to recover.

1. Your computer asks your DNS server, *M*, to find an IP address for domain name *www.baofeng.com*. Which of the following is always true of the name resolution process, assuming that all name servers are configured correctly and no packets are lost? (4')

A. *M* must contact one of the root name servers to resolve the domain name.
B. *M* must contact one of the name servers for *baofeng.com* to resolve the domain name.
C. If *M* had answered a query for the IP address corresponding to *www.baofeng.com* at some time in the past, then it can respond to the current query without contacting any other name server.
D. If *M* has a valid IP address of a functioning name server for *baofeng.com* in its cache, then *M* will get a response from that name server without any other name servers being contacted.

2. Which of the following indications is/are correct? (4')

A. The name resolution records for baofeng.com on DNSPod servers expire after about 23 hours.
B. *Baofeng Player* may automatically retry when failing to access *baofeng.com*
C. Using IP addresses instead of URL is more likely to access websites during the Internet halt.
D. If the problem were not solved in time, DNS servers that are responsible for *.com* domains may be under significant pressure after some time.

3. Which of the following measure(s) may China Telecom take to relieve the situation? (4')

A. Manually add resolution record of baofeng.com to caches of its DNS servers
B. Configure its DNS servers to reject all resolution requests for *baofeng.com*
C. Configure its DNS servers to delete all records of DNSPod servers in the forwarding lookup table
D. Configure its DNS servers to automatically retry on failures

Problem 2: File System (12')

Ben Bitdiddle develops a new file system, BFS, based on UNIX file system version 6, and deploys it on a computer. The total disk capacity on that computer is 32GB, of which about 4GB is used.

Ben's design remains almost the same as UNIX file system, except the way i-nodes store block addresses. (In this question, the term *block address* refers to the block pointers stored in i-nodes.) In BFS, each i-node contains 7 block addresses of 4 bytes each; the first 4 block addresses point to the first 4 blocks of the file, with the remaining 3 addressing the rest of the file. The 5th block address points to an indirect block, containing 128 block addresses, the 6th block address points to a double-indirect block, containing 128 indirect block addresses, and the 7th block address points to a triple-indirect block, containing 128 double-indirect addresses.

Size of each data block is 512-byte. Most files stored on the computer disk are relatively small, with sizes around 1KB. But recently, several extra-large files need be stored on the computer. Their sizes reach the limit of BFS, and Ben has to modify his design to solve this problem.

1. Calculate the size limit of a single file. (For your convenience, you may give the result like: *x* GB + *y* MB + *z* KB) (4')

2. Ben thought of enlarging data blocks to allow a 4GB file to be stored, but gave up the solution after some considerations. Please describe the major trade-off(s) in increasing data block size, and give one possible reason why Ben gave up this solution. You are encouraged to support your answer with simple calculation. (4')

Ben observes that there are 28 bytes allocated to block addresses in each i-node (7 block addresses at 4 bytes each, or 9 block addresses at 3 bytes each, or 5 block addresses at 5 bytes each), and 512 bytes allocated to block addresses in each indirect block (128 block addresses at 4 bytes each, or 170 blocks at 3 bytes each, or 102 blocks at 5 bytes each). He figures that he can keep the total space allocated to block addresses the same, but change the size of each block address, to increase the maximum supported file size. While the number of block addresses in i-nodes and indirect blocks will change, Ben keeps exactly one indirect, one double-indirect and one triple-indirect block address in each i-node.

3. Which of the following statements is/are correct? (4')

A. Increasing the size of a block address from 4 bytes to 5 bytes will allow a 1.5GB file to be stored.
B. Decreasing the size of a block address from 4 bytes to 3 bytes will allow a 2GB file to be stored.
C. Decreasing the size of a block address from 4 bytes to 2 bytes will allow a 8GB file to be stored.
D. Decreasing the size of a block address from 4 bytes to 3 bytes will allow any amount of files less than 2GB to be stored. (Until the disk capacity is used up)

Problem 3: RPC Semantic (18')

Ben is in charge of system design for *Paperence*, a new website for searching academic papers and their references. Luckily for him, Dan had recently completed implementing the provenance-tracking file system from design project 1. Ben uses a RPC interface to allow the web server to interact with the provenance-tracking file system. Ben's web server is **single thread**. You may assume the provenance-tracking file system never crash, but the **network** between web server and provenance-tracking file system is **unreliable** and may drop messages. Here are examples of the RPC interfaces:

// add a paper file to the system, returns a PaperID if success, -1 otherwise.
procedure ADD_PAPER(*file*)

// delete a paper in the system, returns boolean, true if the PaperID existed, false otherwise.
procedure DEL_PAPER(*PaperID*)

// mark Paper B as Paper A's reference, returns boolean, true if mark successfully, false otherwise.
procedure ADD_REF(*PaperID_A PaperID_B*)

// returns the paper file and a list of its reference PaperIDs.
procedure LIST_PAPER(*PaperID*)

1. **Suppose the web server always waits for one operation to succeed before submitting the next one. Neither the web server nor the provenance-tracking file system does anything to deal with the fact that the network can lose messages. Which of the following problems might Ben observe? (Select all that apply) (4')**
   A. The web server may wait forever for a reply from the provenance-tracking file system.
   B. The DEL_PAPER RPC can never reply when a paper is actually deleted.
   C. The LIST_PAPER RPC sometimes returns paper have been deleted.

2. **Ben modified the web server to re-send a request every 5 seconds until it gets reply from the provenance-tracking file system. The provenance-tracking file system has no modifications. Which of the following problems might the modification cause? (Select all that apply) (4')**
   A. The ADD_PAPER RPC sometimes returns -1 when the paper has been added successfully.
   B. The LIST_PAPER RPC sometimes returns no results when matching PaperID exists.
   C. The LIST_PAPER RPC sometimes returns paper have been deleted.
   D. The DEL_PAPER RPC sometimes returns true when the corresponding paper still exists.

3. **In the above scenario of web server, compare remote procedure call with local procedure call, which of the following observations are true? (Select all that apply) (4')**
   A. Multiple replies may be returned from one RPC call.
   B. New error types may be introduced by RPC.
   C. Recursion doesn't work in RPC.
   D. The RPC routine executes more system calls than local procedure call.

4. **Ben is concerned that his web server code might be slow and incorrect, so he comes to you for help. Below, Ben proposes several modifications to his website. For each choice, tell Ben whether it could improve the system throughput and describe your reasons. You should also tell Ben whether it would lead additional sources of incorrect results. (6')**
   A. Cache the results of LIST_PAPER.
   B. Run the web server and the provenance-tracking file system on the same machine to reduce the network delay overhead.

Problem 4: Virtualization (12')

Ben and Alice are two students study at Fudan University. They are both in Computer System Engineering class in software engineering school. They work as a team to complete their design projects. They want to use a system built up using the SEND, RECEIVE and a bounded buffer (As described in Section 5.2 of your text book) to discuss their design projects. The system runs on a server. Alice and Ben use their own notebook computer to SEND and RECEIVE with the server.

Alice first comes up with a design of SEND/RECEIVE system, and talk about it briefly with Ben. The code is as follows.

```
1    Shared structure buffer
2        message instance message[N]
3        long integer in initially 0
4        long integer out initially 0

5    procedure SEND(buffer reference p,    message instance msg)
6        while p.in – p.out = N do nothing
7        p.message[p.in mod N] = msg
8        p.in = p.in + 1

9    procedure RECEIVE(buffer reference p)
10       while p.in = p.out do nothing
11       msg = p. message [p.out mod N]
12       p.out = p.out + 1
13       return msg
```

1. **Which of the following is true for the Alice's implementation: (you can add your explanations to make your answer more clearly) (6')**
   A. If there is only Ben and Alice use this system and they do not send together or receive together, the system will always work well.
   B. If there is only Ben and Alice use this system and they sometimes send together or receive together, the system will always work well.
   C. If server machine that runs this system is single processor and has no pipeline, the system will always work well.
   D. Alice changes the thread scheduler of server machine so that SEND and RECEIVE are atomic procedures, then the new system will always work well.

Ben thinks Alice's design is not quite good, and he wants to find a better one. The Ben's simplified design of SEND/RECEIVE system is as follows:

```
1   Shared structure buffer
2       message instance message[N]
3       long integer in initially 0
4       long integer out initially 0
5       lock instance buffer_lock initially UNLOCK

6   procedure SEND(buffer reference p,   message instance msg)
7       // my_send_index (64-bit int)
8       while True:
9           acquire(p. buffer_lock)
10          if p.in - p.out < N:
11              my_send_index = p.in
12              p.in = p.in + 1
13              release(p. buffer_lock)
14              p.message[my_send_index mod N] = msg
15              return
16          release(p. buffer_lock)

17      procedure RECEIVE(buffer reference p)
18      // my_rec_index (64-bit int)
19      // msg (message)
20      while True:
21          acquire(p. buffer_lock)
22          if p.in > p.out:
23              my_rec_index = p.out
24              p.out = p.out + 1
25              msg = p. message [my_rec_index mod N]
26              release(p. buffer_lock)
27              return msg
28      release(p. buffer  lock)
```

2. **Which of the following is false for the Ben's implementation: (you can add your explanations to make your answer more clearly) (6')**
   A. The code is correct if there is one sender and one receiver executing at same time.
   B. The code is correct if there is one sender and many receivers executing at same time.
   C. The code is correct if there are many senders and one receiver executing at same time.
   D. The code is correct it there are many senders and many receivers executing at same time.

Problem 5: Performance (38')

Determine if the following statements are true or false, if false, explain why and what's the true one. (6')

A. When running a workload, the processor is utilized 85% of its cycles, and other 15% cycles are unused, so the utilization is 85%, and the overhead is 15%.
B. In a web server, the latency is the time from the issuing of request to the receiving of response, it equals to the total latencies of all of the service stages. And the throughput is the number of requests the server has processed, it is inversely proportional to the latency.
C. Speculation can perform operation if there are idle resources, because even if the speculation is wrong, there's no downside, but if there's busy resource, it should never be used to improve the performance.

There're many techniques that can be used to improve the throughput or reduce the latency, like concurrency, caching, scheduling and so on. We will consider 2 of them: concurrency and caching.

A. Concurrency (12')

There's a design hint saying "Instead of reducing latency, hide it.", which means if we cannot reduce the latency of a request because of limits, then we can hide it by overlapping it with other request. Let's take an example, in the web server like in Figure1:

Let's assume that the latency of the disk is 10ms, the memory is 1ms, and the network is 100ms, and there's no latency between the stages.

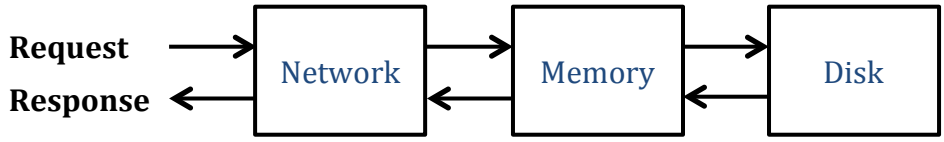


Figure 1

A1. If there's no concurrency, what's the total latency and throughput will be in this case? (3')

If we want to improve the throughput, then we can overlap requests, that we give each stage its own thread of computation so that it can compute concurrently: If a stage has completed its task and has handed off the request to the next stage, then the stage can start processing the second request while the next stage processes the first request, then the pipeline can work on several requests concurrently.

A2. What challenges may this approach meet? Say two, and try to tell one possible solution for each of them. (3')

A3. What's the bottleneck now if the above concurrency has been implemented? And what's the solution to that kind of bottleneck? (You can use a graph to describe) (6')

B. Caching (20')

Another kind of technique is caching, which is a typical example of "Optimize for the common case" design hint that can reduce the average latency for the whole system.

Suppose we have: AverageLatency $= \sum_{i=1}^{n}(F(i) * T(i))$, where n is the number of memory levels the system has, F(i) is the frequency of use of datum stored in level i, and T(i) is the latency of level i. Let's assume n is 3, hit rate of level 1 is 90%, hit rate of level 2 is 80%, and T(1) = 1ns, T(2) = 100ns, T(3) = 1ms.

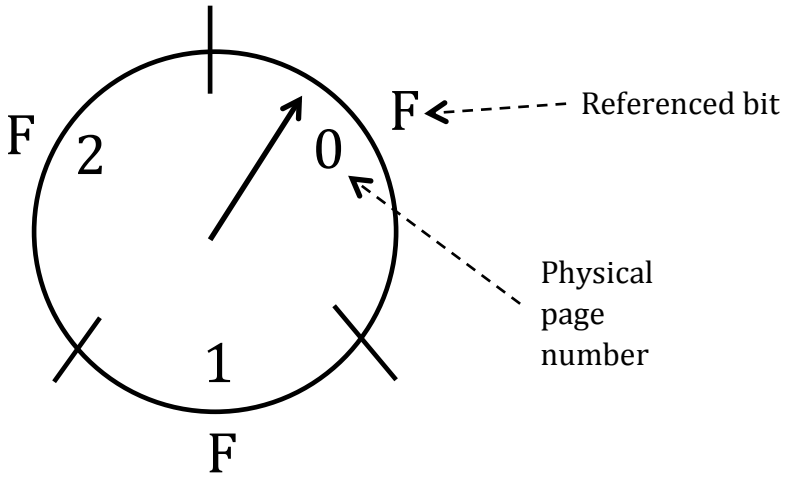
B1. What's the AverageLatency in this multilevel memory system, and how much it reduces compared with the single-level memory system (where there is only one level and T is 10ms)? (4')

Let's now consider about the page-removal policies. The LRU (least-recently-used) is to remove page that has the longest time since a page has been used.

B2. Fill in the following table. (If you don't know what to fill, just write down a '-') (4')

Table 1 LRU Page-Removal Policy with a Three-Page Primary Device												
Time	0	1	2	3	4	5	6	7	8	9	10	-
Reference String	-	3	4	2	6	4	3	7	4	3	6	-
Primary Device Contents	- - -	3 - -	3 4 -	3 4 2								Page Absent Time
Page Absent?	-	√	√	√								

Let's consider about another page-removal algorithm: clock algorithm (You can find it in text book P344 or the slide).



Suppose we have a primary device which has 3 physical blocks, every time a reference string P come, it will follow the pseudo-code:

```
if hit(P)
    res_block = block contains P
    referenced_bit[block contains P] ← T
    clock_arm does not change
else
    while referenced_bit[clock_arm] is T
        referenced_bit[clock_arm] ← F
        clock_arm ← next block
    res_block = block[clock_arm]
    referenced_bit[clock_arm] ← T
    clock_arm ← next block
return res_block
```

B3. Fill in the table 3. (If you don't know what to fill, just write down a '-'), note: '*' means the position of the clock arm; you also need to tell what the referenced bit is for each page block at that time (1 for True and 0 for False). (8')

Table 2 Clock Page-Removal Policy with a Three-Page Primary Device												
Time	0	1	2	3	4	5	6	7	8	9	10	-
Reference String	-	3	4	2	6	4	3	7	4	3	6	-
Primary Device Contents	* - -	3 * -	3 4 *	3* 4 2								-
Referenced Bit	0 0 0	1 0 0	1 1 0	1 1 1								Page Absent Time
Page Absent?	-	√	√	√								

B4. What's the advantages of the clock algorithm compared to RLU? Say at least two of them. (4')

Problem 6: 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 2012

Midterm Examination (Answer Paper)

Name_____ Student No. _____ Score_____

Name_____ Student No. _____

Table 1 LRU Page-Removal Policy with a Three-Page Primary Device												
Time	0	1	2	3	4	5	6	7	8	9	10	-
Reference String	-	3	4	2	6	4	3	7	4	3	6	-
Primary Device Contents	- - -	3 - -	3 4 -	3 4 2								Page Absent Time
Page Absent?	-	√	√	√								

Table 2 Clock Page-Removal Policy with a Three-Page Primary Device												
Time	0	1	2	3	4	5	6	7	8	9	10	-
Reference String	-	3	4	2	6	4	3	7	4	3	6	-
Primary Device Contents	* - -	3 * -	3 4 *	3* 4 2								-
Referenced Bit	0 0 0	1 0 0	1 1 0	1 1 1								Page Absent Time
Page Absent?	-	√	√	√								