Institute for Interdisciplinary Information Sciences (IIIS), Tsinghua University

Final Exam
Operating Systems – Spring 2016

Instructions

1. The point values for each question are marked in square brackets.
2. The exam time is 2.5 hours (150 min.).
3. The total possible points in the exam are 108.
4. Please answer in English on a separate piece of paper. After the exam, you can snapshot using a camera or scan it, and submit on learn.tsinghua.edu.cn as homework "final".
5. The exam is open book and open notes. However, you cannot talk to other people or search online during the exam. All questions must be directed to the administrating TAs over the WeChat Group or GoToMeeting channels – Please let everybody to see it. Private communications with the TAs are prohibited.
6. You understand that a violation on item 5 will result in an immediate reporting to the institute and education affairs office for disciplinary action, which will cause a Fail grade in this course, and may lead to the termination of your student status.

Don't Panic

Good Luck!

A. Paxos Failures (16 points)

Considering the Paxos protocol discussed in class. We have a Paxos-based system with 5 servers. In the following failure scenarios, please answer whether Paxos can reach consensus (both liveness and correctness). Note that everything else works normally except for the described failure. *EXPLAIN* your answer in 4 SENTENCES OR LESS (Answers longer than this may not get credit!). Also, answers without an explanation *GET NOCREDIT*.

1[4] One server runs 10x slower than other nodes due to overheating. (All requests sent to the server is queued infinitely).

2[4] A single faulty node sends a new ballot of every 10ms.

3[4] The network partition is partitioned in to 3-server part and 2-server part

4[4] A single server has a faulty disk storage, and thus forgets about the log after crash.

B. Web Server Scheduling (18 points)

Consider the following server in a public cloud:
The server has a single CPU core and it only has 1Mbps (1 mega**bits** per second) bandwidth in and out of the server. The server serves only a single file that is 1MB in size. Answer the following questions (for this question, you can safely ignore all network packet header overhead, the network round trip time / packet loss etc. and just focus on the throughput):

5[5] The server has only a single thread serving clients. There are 100 clients requesting the file, with a mean inter-arrival time of 16 seconds. What is the average time it takes for a client to finish downloading a file?

6[3] In the same setting as above, how about all 100 clients arrive at the same time?

7[5] As the system administrator realizes the problem of a single thread, she increases the thread count to 1000. Now what is the average download completion time for each client if they arrive all at the same time? (The network bandwidth is shared in a completely fair way.)

8[5] Consider that the 100 clients have different bandwidths ranging from 10Kbps to 10Mbps. The clients tell the server about their bandwidth at the request. Can you design a scheduling policy that optimizes the average response time? Is your policy preemptive or non-preemptive?

C. Virtual Memory (18 points)

Recall that the virtual memory system uses disks as the back-storage if we want to replace some pages and move these pages into disk. Inspired by this, Danny designed a

distributed shared memory system by replacing the disks with a distributed key-value storage system. We have a total of 128K 4KB physical page frames (512 MB in total).

9[4] Consider that the physical memory is empty to start with. The page size is 4KB. How many virtual memory pages do we need for the following trivial program (with no compiler optimization). What are these pages?

```
#define NUM_ELE = 1024 * 1024 *1024 // 1G elements

main() {
  int base = 1;
  int* arr = (int*) malloc(sizeof(int) * NUM_ELE);  // sizeof(int) = 4

  for (int j = 0; j < 10000; j ++) {
    for (int i = 0; i < NUM_ELE; i++)
      arr[i] = base + i;
   }
   base++;
  }
```

10[4] How many page faults the program above will generate using an LRU page replacement algorithm?

11[5] Can we increase the number of physical page numbers to reduce the number of page faults? If so, plot a figure with x-axis as # of physical pages and y-axis as the # of page faults and mark the key points. If not, please explain why.

12[5] Comparing to the local disk based back-storage, the key value storage that Danny uses has a throughput of about 1000MB/sec (using a 10GE network), but it has an overhead of establishing the connection that takes about 50ms. How would you change the paging subsystem to accommodate these new performance numbers? (We still have 512MB of physical memory).

D. Consistency (10 points)

13 [4] In two-phase commit (2PC) protocol, the participants send an ACK message to the coordinator after receiving global-commit/global-abort message. Why is the ACK messages needed?

14 [6] A quorum-based consistency system contains 5 servers. The read quorum is 3,

a) What is the minimal number of servers in the write quorum?
b) Danny proposes an idea to improve the read throughput by upto 3x, especially for large values (say, a 90MB value): we read the first 30MB from the first server, second 30MB from the second and the last 30MB from the last server. Does this scheme work? If so, what is the potential overhead making the throughput less than 3x? If not, can you make it work with a simple change?

E. Transaction (14 points)

Consider the locks for ACID transactions we have discussed in class. We can use read-write lock to implement ACID transaction, and the lock compatibility matrix is as the following:

	ACID-R	ACID-W
ACID-R	√	×
ACID-W	×	×

A recent research paper proposed two extra kinds of transactions in addition to ACID transactions: alkaline transaction and saline transaction1.

- ACID transactions are isolated from all other transactions.
- Alkaline transactions are isolated from other ACID and alkaline transactions.
- Saline transactions expose their intermediate states to all other transactions.
- A saline transaction can include several alkaline transactions.

15 [5] Please fill the Lock Compatibility Matrix below.

	ACID-R	ACID-W	Alka-R	Alka-W	Saline-R	Saline-W
ACID-R	√	×				
ACID-W	×	×				
Alka-R						
Alka-W						
Saline-R						
Saline-W						

16 [4] In addition to being cute, what is the benefit of having the two extra types of transactions?

17 [5] To further improve system performance of ACID transactions, Danny proposes that an ACID transaction can release the ACID-R lock immediately after it uses the data. What problem can this proposal cause? Please provide a concrete example.

F. Synchronization Primitives (10 points)

18 [5] so annoyed by the deadlock problem, Danny proposes a lock primitive with a (user level) supervisor thread that can always enter any locks within the same process by calling lock(), even if they are held by another thread. The holder of the thread is not notified. The supervisor thread can release the lock too (and after release, the lock becomes available so the next thread can get it). Please show an implementation of such a lock.

```
class SupervisedLock(){
void Lock(){
```

[1] English note: 这些名称都是比较萌的，ACID 是酸性，Alkaline 是碱性，saline 是两者结合，叫做盐。

```
}
void Unlock(){

}
}
```

19 [5] Why is the SupervisedLock a stupid primitive to provide at the user level? What kind of problem can it cause? How can the kernel deal with deadlocks without such primitive?

G. File Systems (12 points)

You have a 100-byte text file /tmp/1.txt and you run "cd /tmp; ln -s /tmp 2".

20 [4] Assume you executed "cat /tmp/2/2/2/1.txt", how many disk block do you need to read from disk? Write down the blocks for partial credit.

21 [4] When you type "rm /tmp/2/2/2/2/2/2/2/2/2/2/1.txt", how many blocks are changed on the disk?

22 [4] Why do we need a special file .. in each directory that is a hard link to the parent directory?

H. Block chains (10 points)

23 [5] We mentioned the waste of energy problem in the proof-of-work (PoW) scheme in Bitcoin. Danny proposes a new way to select the leader of the next round: Each node receives the GPS signal of the time (round to the second-level accuracy). Then the server computes a secure hash of (the server's IP address + the GPS time). If the hash of the location is in a specified range (a configurable "difficulty" parameter), the server is a leader. If there is no leader, the system will wait for the next leader. What can be a problem of this scheme?

24 [5] In the Bitcoin protocol we discussed in the class, you may think that you will broadcast block, if you solved the PoW puzzle, as soon as possible so you can be the first to receive the reward. Instead, a selfish miner would like to delay publishing the block. Can you explain how it can benefit by delaying the publication? (Hint: consider the fork / branch resolution protocol).

THE END

Congratulations!