System and Network Security (CS 5470)

End Semester Examination (Spring 2017)

International Institute of Information Technology, Hyderabad

Time: 3 Hours Total Marks: 60

Instructions: Answer *ALL* questions. This is a closed books and notes examination. Write your answers sequentially as given in the question paper and also all the parts of a question at the same place. **No query is allowed in the examination hall.** Use of Regular Calculator is allowed.

---

1. **(a)** Explain the steps used in the Encapsulating Security Payload (ESP) protocol under tunnel mode for providing security of IPv4 packets.
   (b) To protect replay attacks, IPSec (IP security) makes use of sliding receiving window and unique sequence number to each IP packet. To remedy the drawbacks found in this procedure, Zhao and Wu proposed an improved and effective solution to that method. Discuss the Zhao-Wu's method explaining how the anti-replay receiving window differs from the sliding receiving window in the original IPSec.
   (c) It is known that ESP does whatever Authentication Header (AH) does with additional functionality (privacy). Why do you need AH then?
   (d) Explain how do you apply AH and ESP in IPv6?

   [4 + 4 + 1 + 1 = 10]

2. **(a)** Briefly explain the operational description of the Pretty Good Privacy (PGP) used for e-mail security explaining the function, algorithms used and their description.
   (b) Explain in detail the confidentiality and authentication services mathematically with the help of diagrams provided in PGP.

   [5 + 5 = 10]

3. **(a)** Explain the the objectives of the Secure Socket Layer (SSL) protocol used for providing security at the transport layer.
   (b) Describe the SSL handshake protocol.
   (c) What are the fields present in the Record Protocol Header (RPH) of the SSL Record Protocol?

   [3 + 5 + 2 = 10]

4. **(a)** Explain the dual signature. What is the need of the dual signature in the Secure Electronic Transaction (SET) to protect credit card transactions?
   (b) Write the formats of Authorization Request (AuthReq) and Capture Response (CapRes) messages used in SET.
   (c) What are participants in SET?

   [(2 + 2) + (2 + 2) + 2 = 10]

5. (a) Consider an elliptic curve $E_{11}(1,6)$ and a point $P = (10,2)$ in $E_{11}(1,6)$. For a scalar $k \in Z_p^*$, the ECC point multiplication $k.G$ requires k ECC point additions. Suppose you want to solve this problem with the help of the fast repeated doublings method, which requires only $\log_2(k)$ ECC point additions instead of k ECC point additions.
Using this method only, compute $5P$.
(b) Consider again the elliptic curve $E_{11}(1,6)$ and a base point $G = (2,2)$ in $E_{11}(1,6)$. Suppose two parties, users A and B agree on elliptic curve $E_{11}(1,6)$ and a base point $G = (2,2)$. User B selects a private key $n_B = 7$ and computes the corresponding public key $P_B = n_B.G = (7,2)$.
Let the ciphertext produced by user A using the public key P_B of user B be $C_m = (C_1, C_2) = ((0,7),(7,2))$ and then it was sent to user B. Calculate the original plaintext P_m by the user B after receiving the ciphertext C_m.

$C_2 - n_B C_1$

$k G, \quad P_m + k P_B$ [5 + 5 = 10]

6. Consider the following authentication scheme, which was discussed in the class [**Sravani Challa, Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Alavalapati Goutham Reddy, Eun-Jun Yoon, and Kee-Young Yoo. "Secure Signature-Based Authenticated Key Establishment Scheme for Future IoT Applications," in IEEE Access, Vol. 5, pp. 3028-3043, 2017. (2015 SCI Impact Factor: 1.249)**].
With respect to this scheme, answer the following questions:
(a) What was the threat model used in this scheme?
(b) Explain the sensing device registration phase and user registration phase.
(c) Discuss the importance of using the timestamps and ECC-based signatures in this scheme.

[2 + (2 + 3) + 3 = 10]

********** **End of Question Paper** **********

Database Systems

End Semester Exam

24th April 2017 **Duration: 3 hrs, Marks: 100**

1. **No clarifications during the exam.**
2. **Make *reasonable assumptions* and *clearly state* them to answer *ambiguous* questions.**
3. **Show your steps. Be concise and organized.**
4. **Calculators allowed. Sharing of calculators *not* allowed.**

(a) With respect to transaction serializability, briefly describe what is:
(A) 2PL
(B) Validation test
(C) Timestamp ordering

[9]

(b) Consider this schedule:
$R_2(a), R_1(a), W_1(a), R_3(b), R_3(c), R_2(a), W_2(a), R_2(c), W_3(c), R_1(d), R_3(d), W_1(d)$
Is this schedule valid in each of the above protocols? Describe why or why not, for each one.

[15]

(c)
I. What is the precedence graph for the schedule of part b?
II. Is the schedule conflict-serializable? If so, what are all the equivalent serial schedules ?

[5]

The following is a sequence of redo-log records written by two transactions T and U: <START T>; <T, A, 10>; <START U>; <U, B, 20>; <T, C, 30>; <U, D, 40>; <COMMIT U>; <T, E, 50>; <COMMIT T>. Describe the action of the recovery manager, including changes to both disk and the log, if there is a crash and the last log record to appear on disk is:
a. <START U>
b. <COMMIT U>
c. <T, E, 50>
d. <COMMIT T>

[10]

Consider the following sequence of undo log records: <START S>; <S, A, 60>; <COMMIT S>; <START T>; <T, A, 10>; <START U>; <U, B, 20>; <T, C, 30>; <START V>; <U, D, 40>; <V, F, 70>; <COMMIT U>; <T, E, 50>; <COMMIT T>; <V, B, 80>; <COMMIT V>. Suppose that we begin a non-quiescent checkpoint immediately after one of the following log records has been written (in memory):
a. <S, A, 60>
b. <T, A, 10>
c. <U, D, 40>
d. <T, E, 50>
For each, tell:
i. When the <CKPT> record is written, and
ii. For each possible point at which a crash could occur, how far back in the log we must look to find all possible incomplete transactions.

[16]

4) The Megatron 777 disk has the following characteristics:

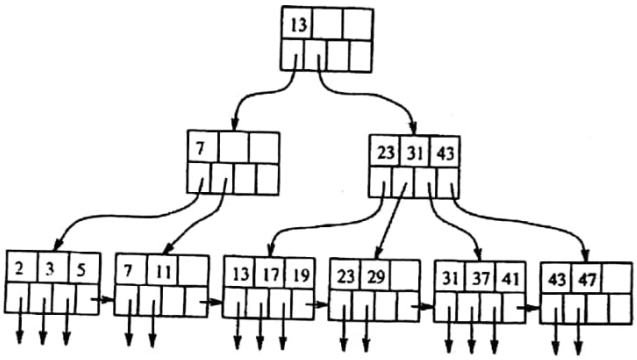
1. There are 10 surfaces, with 10,000 tracks per surface.
2. Tracks hold an average of 1000 sectors of 512 bytes each.
3. 20% of each track is used for gaps.
4. The disk rotates at 10,000 rpm.
5. The time it takes the head to move n tracks is $1+0.001n$ milliseconds.

We have a 1 GB sized relation R of 10,000,000 tuples. Each tuple of 100 bytes has several fields, one of which is the sort key field, which may not be a primary key. The machine on which sorting occurs has one Megatron 777 disk and 50 MB of main memory available. Disk blocks are 4096 bytes. How long would it take to sort R using 2-phase, multiway merge sort.

[10]

5) Describe steps, using diagrams if necessary, to execute the following operations on the shown B+ tree:

(a) Lookup all records less than 10
(b) Insert a record with key 6
(c) Delete record with key 7



[5]

6) Suppose we are using RAID level 4 (i.e. 1 redundant disk for parity), with 4 data disks and 1 redundant disk. For simplicity, assume blocks are a single byte.

(a) Give the block of the redundant disk if the corresponding blocks of the data disks are: 01010110, 11000000, 00111011, and 11111011.

(b) Data disk 1 has failed. Recover the block of that disk if the contents of disks 2 through 4 are: 01010110, 11000000, and 00111011, while the redundant disk holds 11111011.

[10]

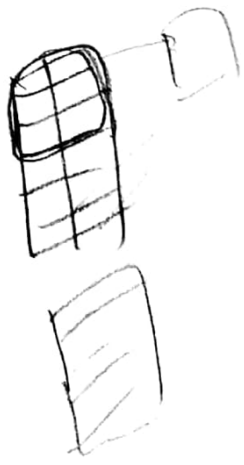
7) Suppose blocks hold either 3 records, or 10 key-pointer pairs. As a function of n, the number of records, how many blocks do we need to hold a data file and:

(a) A dense index? (b) A sparse index?

[10]

8) Write the iterator functions pseudo code for a tuple-based nested-loop join. If B(S) = B(R) = 10,000 and M=1000, what is the number of disk I/O's required for nested-loop join.

[10]



END SEMESTER EXAMINATION
IIIT, Hyderabad, Spring 2017
Subject: Introduction to Parallel Scientific Computing
(Code: TS17005)
Instructor: Dr. P. Kumar

26.04.2017
09:00 - 12:00

Time: 3 hours

Maximum Marks: 100

Instructions

There are 4 pages and 26 questions. All questions are compulsory. Calculators are not allowed.

True or False? Justify your answer.

1. Given N processors ranked from 1 to N, they can compute $N!$ using MPI_Reduce(). [3]

2. Latency hiding is a technique to overlap communication with computation in order to increase the speed and efficiency of a parallel program. Typically, it involves using blocking send and receive routines. [3]

3. Data locality is critical only for message passing computer and not for shared address space computers, because shared address space computers have additional hardware that makes the entire memory globally accessible. [3]

4. 2D partitioning of data always leads to higher degree of concurrency then 1D partitioning and are always preferred. [3]

5. It is often easier to get good load balancing if concurrency is derived using data decomposition as opposed to task decomposition. [3]

Multiple Choice Questions

6. Suppose that MPI_COMM_WORLD consists of three processes 0, 1, and 2, and suppose the following code is executed. [3]

```
int x, y, z;
switch(my_rank){
  case 0:
    x = 0; y = 1; z = 2;
    MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Send(&y, 1, MPI_INT, 2, 43, MPI_COMM_WORLD);
    MPI_Bcast(&z, 1, MPI_INT, 1, MPI_COMM_WORLD);
    break;
  case 1:
    x = 3; y = 8; z = 5;
    MPI_Bcast(&x, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
    break;
  case 2:
    x = 6; y = 7; z = 8;
    MPI_Bcast(&z, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Recv(&x, 1, MPI_INT, 0, 43, MPI_COMM_WORLD, &status);
    MPI_Bcast(&y, 1, MPI_INT, 1, MPI_COMM_WORLD);
    break;
}
```

What are the values of x, y, and z on each process after the code has been executed?

i)

	x	y	z
P0	0	1	2
P1	3	8	5
P2	6	7	8

ii)

	x	y	z
P0	0	1	8
P1	0	8	5
P2	1	8	0

iii)

	x	y	z
P0	0	1	2
P1	0	2	5
P2	1	2	0

iv)

	x	y	z
P0	0	1	4
P1	0	4	5
P2	1	4	0

v) none

7. Suppose the following two code snippet are running in a shared memory environment. [3]

Student's name:

Please go on to the next page...

```
// Thread 1                    // Thread 2
x = 3;                         x = 0;
x = x + 1;                     x = 2 * x;
```

7. After both threads terminate, it is not possible that the shared variable x takes a value of
i) 8 ii) 0 iii) 7 iv) 6 v) 2 vi) none

8. Consider a communicator with 4 processes. How many total `MPI_Send()`'s and `MPI_Recv()`'s would be required to accomplish the following

```
MPI_Allreduce ( &a, &x, 1, MPI_REAL, MPI_SUM, comm );
```

i) 3 ii) 4 iii) 12 iv) 16 v) None of these [3]

Descriptive Questions

9. What is the difference between critical section and atomic? Provide an example where atomic is more beneficial than critical. [3]

10. What is the difference between task and data parallelism? [3]

11. What is the maximum possible speedup if 60% of your program can run in parallel on a 3 core machine? What if you have infinite number of processors? [3]

12. Two-third of a method M can be parallelized. How many processors are needed to achieve a speedup of 2? [3]

13. Describe a parallel message passing algorithm (pseudocode) to find the factorial of all numbers between 1 and N in O(log N) steps using N processors. [3]

14. Demonstrate a fragment of a multithreaded program written in pseudocode which may produce different results from run to run due to round-off errors. [3]

15. Demonstrate a fragment of a multi-threaded program written in pseudocode in which race condition occurs. [3]

16. What does UMA and NUMA stand for? What is the major difference between NUMA machine and message passing architecture? [3]

17. Explain how would you parallelize the following piece of C-code. Justify your choices. Insert necessary OpenMP pragmas. [3]

```
#define N 1000
int i, j;
float a[N][N];
for (i = 0; i < N; i++)
  for (j = 1; j < N; j++){
     if (i != j)
       a[i][j] = -2 * a[i-1][j];
     else
       a[i][i] = 1;
  }
```

18. The value of π can be approximated by the following integral [4]

$$\int_0^1 \frac{4}{1+x^2} = \pi.$$

1. Describe a strategy for splitting this problem up into one that can be used with MPI.
2. Write a message passing pseudocode that implement your approach.

19. Consider the code segment [3]

```
#pragma omp parallel for
{
  for(i=0; i<n; i++)
    x[k[i]] = x[k[i] + 1] + B[i];
}
```

Are there any races in this loop? Explain.

20. Transform the following loop to parallel form using OpenMP, if possible. [3]

```
k = 1;
for(i = 0; i < n; i++){
  k = 2 * k;
  x = b[i] + c[i];
  a[k] = x + sin[x];
}
```

21. Consider the code segment [3]

```
#pragma omp parallel
{
#pragma omp for nowait
   for(i=0; i<n; i++){
     a[i] = b[i] + c[i];
     for (; a[i]>x[i]; ){
        a[i] = a[i] - x[i];
     }
   }
 #pragma omp for nowait
    for(i=0; i<n; i++){
      c[i] = d[i] + e[i];
    }
}
```

Does each of the nowait directives improve performance? Explain.

22. Consider the code segment. [3]

```
...
t = a + b;
y = c * d;
t = t + y;
for (i = 0; i < n; i++){
  x[i] = x[i] + t;
}

for (i = 0; i < n; i++){
  printf("x[%d] = %f\n", i, x[i]);
}
```

Consider the following translation to parallel form:

```
#pragma omp parallel private(y)
{
  t = a + b;
  y = c * d;
  t = t + y;
#pragma omp for nowait
 {
  for(i = 0; i < n; i++)
    x[i] = x[i] + t;
 }
```

```
}

for (i = 0; i < n; i++)
  printf("x[%d] = %f\n", i, x[i]);
```

Is the above parallel code correct? If not, then suggest corrections.

23. A small college wished to assign unique identification numbers to all of its students. The administration wants a six digit number to be used, but with a lot of constraints. Provide a pseudo-code for a parallel shared memory program to 'count the number' of six-digit combinations satisfying the given constraints:

    - The first digit may not be zero
    - The consecutive digits may not be the same
    - The sum of the digits may not be 7,11, or 13

    [7]

24. Use MPI_Send and MPI_Recv to implement your own version of MPI_Bcast and MPI_Scatter (syntax is not what is being asked, the functionality is being questioned. A clear understanding of what each one DOES is important to answer the question). [8]

25. **(Ring Communication)** Given is a uni-directional ring network with n processing nodes, thus, one processing node n_i can only send messages to its successor $n_i + 1$, and receive messages from its predecessor $n_i - 1$. All processing nodes now want to communicate in parallel-first sending a message to their successor before receiving a message from their predecessor. If done in the described manner, there shouldnt arise any complications. But what happens if we change the order? Now every processing node first wants to receive a message from its predecessor before sending a message to its successor. Write a Message passing pseudocode for a uni-directional ring network with n processing elements. Thus, according to the situation described above, every processing node first wants to receive a message from its predecessor before itself sending a message to its successor without running into a deadlock. When finished, every processing node should confirm with a short message printed to the console. [8]

26. **(A method of identifying prime numbers)** First the set of all natural numbers $P = 2, ..., n$ from 2 to an upper limit n is formed. Then, after searching for the minimum p of set P the first prime number all multiples of p from P are deleted. Successively repeating this step until $p = \lfloor \sqrt{n} \rfloor$ there are only prime numbers left in P. Write a message passing code following the algorithm described above to find all prime numbers in a set $P = \{2, ..., n\}$. For a faster processing, the main work-namely to delete all multiples of a prime number-should be done in parallel. Thus, the set P has to be divided into equal parts P_i and distributed to all processes $i \in [0, \text{max\_processes}[$. Each process i searches for its local minimum p_i from its subset P_i and sends it to the master process. The master process determines the global minimum $p = \min p_i$, $i \in [0, \text{max\_processes}[$, and tells it to all slave processes, that now they can delete all multiples of p from their subsets P_i. This step will be successively repeated until the condition $p \leq \lfloor \sqrt{n} \rfloor$ becomes false. Now every subset P_i of process i only consists of prime numbers. In the end, the largest prime number found should be printed to the console. [10]

Non-blocking send/receive

List of MPI routines

1. `MPI_Bcast(void* data, int count, MPI_Datatype datatype, int root, MPI_Comm communicator)`
2. `MPI_Send(void* data, int count, MPI_Datatype datatype, int destination, int tag, MPI_Comm communicator)`
3. `MPI_Recv(void* data, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm communicator, MPI_Status* status)`
4. `MPI_Scatter(void* send_data,int send_count,MPI_Datatype send_datatype, void* recv_data,int recv_count,MPI_Datatype recv_datatype,int root,MPI_Comm communicator)`
5. `MPI_Reduce(void* send_data, void* recv_data, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm communicator)`