COMP 533 Midterm

Thu, Mar 9, 2017, 12:30-1:45

Instructions
1. Please spread out and try and sit in alternate seats.
2. This is a closed book exam.
3. Write on the exam itself.
4. Write on the opposite side of a question page if there is not enough space to solve a problem.
5. There are:
   - 6 numbered pages including this one; there are no blank numbered pages.
   - 4 questions.
   - 75 possible points. Point values appear next to each question.
6. You have 75 minutes.
7. Explain all answers.
8. If you need to make any assumptions to clarify a problem, write your assumptions down. Only reasonable assumptions get full credit.

Name (in Capitals) and Onyen

____________________________________

Pledge: I have neither given nor received unauthorized aid on this exam.
(signed)____________________________________

Please do not write below
1. _____/10  2. _____/17  3. _____/28  4. _____/20

Total: _____/ 75
1. [10 pts.] THREADS

Explain why a multi-threaded program can run:
   a. faster than an equivalent single-threaded program.

   b. slower than an equivalent single-threaded program.
2. [17 pts.] **Thread Synchronization**
Consider an example of thread synchronization you used (or were required to use) in the NIO assignment.

Identify the dependencies between the two tasks, by describing what services, if any, each thread provides to the other.

Explain why the two threads:

a) were not combined into one (be as specific as you can be on what specific problems will arise in the assignment if they were combined).

b) needed to be synchronized
3. [28 pts.] IPC Performance
Consider a distributed operating system that provides a global thread space in which a thread in one process can name threads in the same and other processes, possibly executing on different hosts. The operating system supports the following operations:

```c
void async_send (receiving_thread, byte_buffer)
ByteBuffer send_wait (receiving_thread, byte_buffer)
ByteBuffer receive_wait (sending_thread, byte_buffer)
void reply (byte_buffer)
```

`async_send()` does not block; `receive_wait()` waits for a message from the specified sender; `send_wait()` waits until the receiver replies using `reply()`; and `reply` is a non-blocking call that sends its argument to the last sender from which the replying thread received a message via `send_wait`, which is returned by the last `send_wait()` executed by the sending thread. The communicating threads may be on the same or different machines. When they are on different machines, the distributed operating system uses TCP/IP as the underlying network protocol.

Consider a pair of communicating processes, S1 and R1, that execute the following code:

```c
S1
for i ← 1 to N
    async_send (R1, message)
end
and another communicating pair, S2 and R2, that execute the following code:

S2
for i ← 1 to N
    result ← send_wait (R2, message); 
end

R1
for i ← 1 to N
    receive_wait (S1, message)
end

R2
for i ← 1 to N
    receive_wait (S2, message)
end
```

The following four experiments are performed and the times taken by the sending process to execute the loop observed:

1. S1 and R1 execute on the same computer. Time taken: 4 seconds.
2. S1 and R1 execute on different computers. Time taken: 2 seconds.
4. S2 and R2 execute on different computers. Time taken: 7 seconds.

If you cannot justify any of the following differences, explain why not.

Explain why experiment:

a) (4) takes more time than (2). (Distributed Wait vs Distributed Async)
b) (3) takes more time than (1). (*Local Async vs Local Wait*)

c) (2) takes less time than (1). (*Distributed Async vs Local Async*)

d) (4) takes more time than (3). (*Distributed Wait vs Local Wait*)
4. [20 pts.] IPC Design
Assume a version of the above distributed operating system without async_send. Let us call this system DOS. Distinguish between DOS and RMI based on various dimension of the IPC design space we saw in class and other factors specific to these two systems that you identify. Give only the differences. You do not have to discuss the consequences of the differences or justify your classification. However, you may get partial points for a justification for an unexpected classification. You can get more than the maximum points as extra credit, based on how many differences you point out.