Name: _____

CS 6/73201 **Advanced OS**

Final Exam

Wednesday 9 May 2001

1. **For each of the following pairs of terms, briefly explain how the two are similar and are different (note that I am not asking you to define the terms!):**

   a. **SIMD, MIMD (5 points)**

   Both have multiple processors and can operate on different pieces of data in parallel.

   In SIMD, the processors all execute the same instruction at the same time, while in MIMD, each processor could be executing a different instruction.

   b. **Multiprocessor, multicomputer (5 points)**

   Both are MIMD machines, usually with tightly coupled software.

   Multiprocessors have shared memory and tightly coupled hardware, while multicomputers have distributed / private memories and either tightly coupled hardware (parallel machines) or loosely coupled hardware (distributed machines).

   c. **UMA, NUMA (5 points)**

   Both have hardware access to remote memory.

   In a UMA machine, all processors can access a particular memory in the same amount of time. In a NUMA machine, a processor can access its local memory more quickly than it can access a remote memory, hence "non-uniform" access to different memories in terms of latency.

   d. **NUMA, DSM (5 points)**

   Both provide access to remote memory, allowing the programmer to access a global memory space. Both have software-controlled caching, managed by the OS.

   NUMA provides direct hardware access to remote memory, while DSM provides the illusion of direct hardware access, while in reality the DSM system uses message-passing to gain access to remote data.

2. **Define "connection-oriented communication". (5 points)**

   In connection-oriented communication, the parties must explicitly connect, then they can exchange data (as a stream of bytes, delivered in the same order they were sent), then they must explicitly release the connection.

Name: _____

3. Summarize Lamport's algorithm for distributed mutual exclusion. (25 points)

When a thread wants to enter the CS, it (1) adds the request to its own request queue, and (2) sends a timestamped request message to all threads in that CS's request set

When a thread receives a request message, it: (1) adds the request to its own request queue, and (2) returns a timestamped reply message

A thread enters the CS when both: (1) its own request is at the top of its own request queue (its request is earliest), and (2) it has received a reply message with a timestamp larger than its request from all other threads in the request set

When a thread leaves the CS, it: (1) removes its own (satisfied) request from the top of its own request queue, and (2) sends a timestamped release message to all threads in the request set

When a thread receives a release message, it: removes the (satisfied) request from its own request queue (Perhaps raising its own message to the top of the queue, enabling it to finally enter the CS)

4. What is a "false deadlock" and why is it bad to report one? (10 points)

False deadlock refers to a false report of deadlock by a deadlock detection algorithm — reporting a deadlock that does not exist, due to outdated information or messages in transit.

It is bad to falsely report a deadlock, because a deadlock recovery algorithm will be needlessly invoked, perhaps terminating one or more processes.

5. Explain how the "probe" message in Chandy, Misra, and Haas' algorithm for distributed deadlock detection follows the path of a circular hold and wait, and thus detects a deadlock. (15 points)

When a process requests resources that are not available, it blocks and sends a probe message to the process holding the requested resources. That process forwards the message on to any processes holding resources that it has requested. If the probe reaches the processes that originally sent it, it must have followed a circular hold-and-wait path, since the message is only sent from processes holding a resource to processes from which they are waiting on resources. Since no preemption of resources is allowed, and resources are held in a mutually-exclusive manner, this hold-and-wait condition establishes deadlock.

6. Two common concurrency control mechanisms for atomic transactions are two-phase locking and strict two-phase locking. The major difference between these two mechanisms is that two-phase locking has a shrinking phase as locks are released, while in strict two-phase locking all locks are held until the transaction completes.

a. What problem does this shrinking phase cause in two-phase locking? (10 points)

The shrinking phase releases the locks gradually, thus exposing intermediate results to other transactions. If another transaction uses one of those results, and the first transaction later aborts, the second transaction must also be aborted. If its results have likewise been used

by other transactions, those transactions must also be aborted. This process of one abort affecting others is called "cascading aborts", and is not desirable.

b. Strict two-phase locking eliminates the shrinking phase to avoid this problem, but eliminating the shrinking phase adds a new problem. Explain. (10 points)

Since all locks are now held until the end of the transaction, no other process can use the locked data until then, reducing the overall concurrency among the transactions.

7. With regard to distributed file systems, what is "UNIX semantics"? (10 points)

UNIX semantics means that the result of a write goes immediately to the file, so a read always returns the last value written.

8. In process migration, three alternatives in transferring the address space are total freeze, pre-transfer, and transfer on reference. Briefly explain these three alternatives. (15 points)

Total freeze: the process is frozen, the address space is transferred to the destination node, and then the process is restarted there

Pretransfer: the process continues to execute on the source node as the address space is transferred to the destination node, then the process is frozen, and any memory data modified during the first transfer is transferred, then the process is restarted on the destination node

Transfer on reference: parts of the address space are transferred to the destination node only when they are referenced on that node.

9. Explain how a sender-initiated load distribution algorithm can become unstable at high system loads. (10 points)

At high system loads, most of the processes will be senders, and have very little chance of finding a receiver. However, if the system is already heavily loaded, all the fruitless work and messages trying to find a receiver will load the system even further, possibly pushing it into instability.

10. How does a cluster differ from a distributed system, in terms of architecture, and in terms of tasks for which the system is typically used? (20 points)

In terms of architecture, clusters are a subclass of distributed systems. Clusters are usually small-scale homogeneous systems, while distributed systems may be large and heterogeneous.

In terms of tasks, clusters are usually dedicated to a particular task (e.g., high-availability web service, computing farms), while distributed systems may be more general-purpose (e.g., web services, workload sharing).