classmate
Date
Page

NAME - SAHIL SILARE

ROLL NO - 19115071

SUBJECT - DISTRIBUTED SYSTEMS

classmate
Date
Page

19115071

Ans 2: Under the sender-initiated algorithm, load-distribution is initiated by an overloaded node, which tries to send a task to an underloaded node, also called the receiver.

The transfer policy of each sender-initiated algorithm is based on the concept of queue length .i.e. node is identified as sender if queue length exceeds threshold T.

The three sender-initiated algorithms are Random, Threshold & Shortest. Under the random algorithm, a task is simply transferred to a randomly selected node, with no information between the node to help them. Under the Threshold algorithm, a location policy can avoid useless task transfer by polling a node to determine whether transferring a task would make its queue length exceed T.

The previous 2 approaches makes no effort to identify the best destination node under the shortest location policy, a number of nodes are polled randomly to determine their queue length. The node with the shortest queue length is selected to transfer the task

In receiver initiated algorithm,

19115071

load distributing activity is initiated by an underloaded node (receiver) which attempts to get a task from an overloaded node. The algorithm's threshold policy bases its decision on the CPU queue length.

The choice of using sender-initiated or receiver-initiated algorithm depends on the kind of system we are dealing with. In case of high system loads, any of the sender-initiated algorithms described previously cause system instability. At such times, no node is likely to be lightly loaded, so a sender is unlikely to find a suitable destination node. However, polling activity in sender-initiated algorithms increases as the task arrival rate increases, eventually reaching a point where cost of load-sharing is greater that it's benefits. Thus, actions of sender initiated algorithms are not effective at high system loads & causes instability.

On the other hand, in receiver-initiated algorithms, polling starts when a node becomes a receiver However, three polls seldom arrive at sender just after new tasks have arrived

at the senders but before tasks have begun executing. Consequently, most tasks are pre-emptive & therefore costly.

Ans 3: Process migration is a special technique of process management where one process is transferred from one computing environment to another computing environment. This allows a process to execute on some other node, thus lowering the load on a particular node.

In process migration, we need to realize that we are not really moving a process from one node to another. Instead, what usually happens is that we follow certain steps: Firstly, we freeze the process. i.e: we stop its execution on the source node & save its state. Since, now the process state is totally static & serialised, it could be revived remotely or locally. Finally, we 'throw' the process & let it run as if nothing happened. This is very similar to cloning a process so it can be executed at a later time.

We note that since migration implies persistence of being, that is the reason process id is not changed. Since everything is copied from one

node to another, process id would stay the same.

This is done so the simulation of process on the destination node remains as close as possible as the original run of the process on the source node.

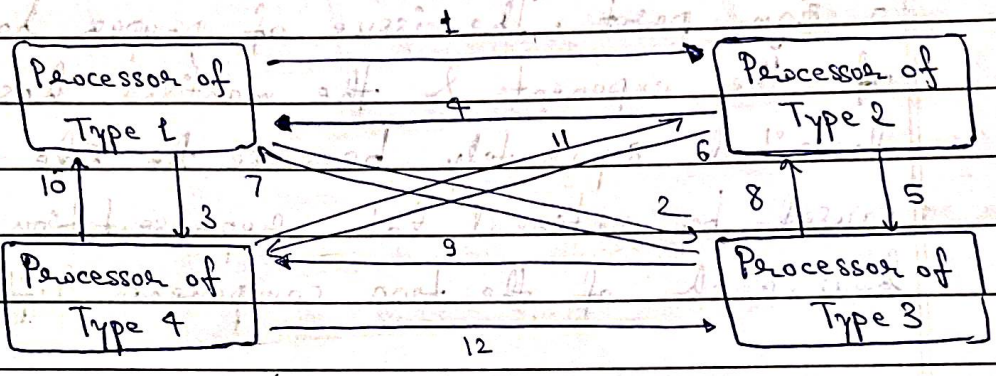
Also, the code executed by the process may have some use of process id, hence it was ordained that while migrating a process, try to keep everything as similar as possible.

Ans 4: The process of converting from a particular machine representation to external data representation format is called serializing. On the other hand, process of converting from external data representation to a particular machine representation is called deserialization. In standard data representation format is called external data representation, & its designer must handle the problem of different representations of data such as characters, integers & floating point issue. Again, in all

19115071

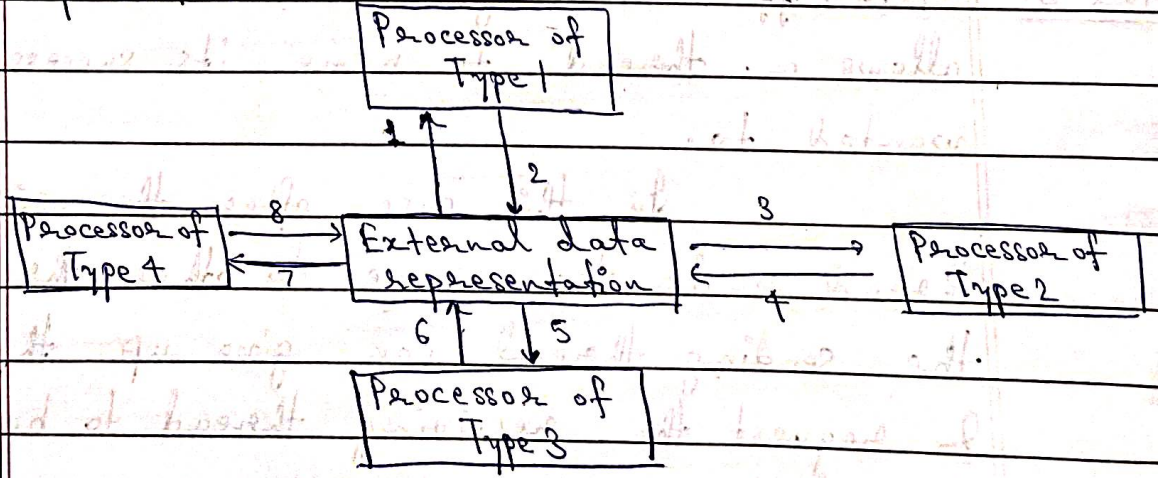
these issues, we need to take special care of floating point numbers.

The issues were discussed by Maguire & Smith [1988] for handling floating point numbers in external data representation schemes are given below:-



This example illustrates the need for 12 pieeces of translation software in a heterogenous system, having 4 types of processor.

We note th a large number of translation software are used. This can be reduced by using external da representation.



In this example, we only need 8 pieces of translation software in heterogenous system having 4 types of processors where external data representation is used.

A floating point number representation, consists of exponent part, a mantissa part, & a sign part. The issue of proper handling of the exponent & the mantissa has been described separately because the side effects caused by external data representation affect how each of the two components differently. We also need to consider the accuracy format, consistency & resources required for the same. By analyzing these factors well, one can design an efficient external data representation which can further improve the efficiency of the existing algorithm.

Ans 5: Handoff scheduling refers to a scheme that allows a thread to name its successor if it wanted to.

In this case, after the original thread sends the message to other thread, the sending thredd can give up the CPU & request the receiving thread to be allowed to run next.

19115071

On the other hand, in affinity scheduling a thread is scheduled on the CPU it last ran on in hopes that part of its address space is still in that CPU's cache. It is used in multiprocessor system.

Affinity scheduling faces a major disadvantage that it diminishes the chances of successfully doing load sharing because of the desire to retain a job on its current processor. In effect, affinity based scheduling injects a measure of global scheduling into the local scheduling policy.

On the other hand, handoff scheduling is useful when the identity of the thread must make process before the current thread runs again is known. Hence, this is useful if we know the next steps of the procedure we're running.

19115071

Ans 7: Serializability is a property that ensures that concurrently executing transactions do not interfere with each other. That is, the concurrent execution of a set of two or more transactions is serially equivalent in the sense that the ultimate result of performing them concurrently is always same as if they had been executed one at a time in some order.

A schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of conflict serializability & view serializability.

Conflict Serializability:

- Instructions I_i & I_j of transactions T_i & T_j respectively, conflict if & only if there exists some item Q accessed by both I_i & I_j, & at least one of these instructions wrote Q

1. I_i = read(Q), I_j = read(Q)
2. I_i = read(Q), I_j = write(Q)
3. I_i = write(Q), I_j = read(Q)
4. I_i = write(Q), I_j = write(Q)

19115071

View Serializibility:

Let S & S' be two schedules with the same set of transactions. S & S' are view equivalent if following three conditions are met, where Q is the data item & T_i is a transaction:

↳ If T_i reads the initial value of Q in schedule S, then T_i must, in schedule S', also read the initial value of Q.

↳ If T_i executes read(Q) in schedule S, & that value was produced by transaction T_j (if any), then transaction T_i must in schedule S' also read the value of Q that was produced by transaction T_j.

↳ The transaction that performs the final write(Q) operation in schedule S must perform the final write(Q) operation in schedule S'.

19115071

Ans 6:

Ans 6: (a) A distributed system that supports diskless workstations.

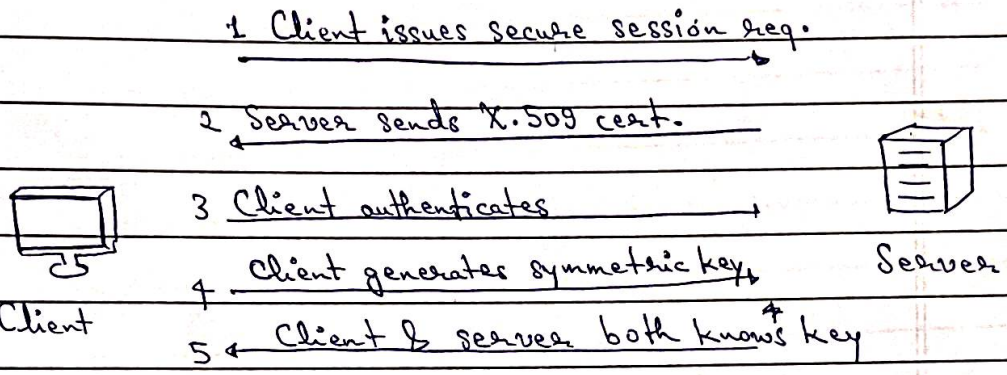
Ans 8: False sharing occurs when two different processes access two unrelated variables that reside in the same data block. In such situations, even though the original variables are not shared, the data block appears to be shared by two processes. The larger is the block size, the higher is the probability of false sharing, due to the fact that the same block may contain different data structures that are used independently.

The granularity of locking refers to the unit of lockable data items. In a file system supporting transactions, this unit is normally an entire file, a page, or record. If many transactions share files, the granularity of locking can have a significant impact on how many transactions can be executed concurrently. In the case of false sharing, even though the two data items can be accessed concurrent-ly by the two transactions, this in not allowed because the granularity of locking is a file. Notice that locking granularity increases concurrency by reducing the possibility of false sharing.

19115071

Ans 9: SSL Certificates are used to establish & authenticate client & server before actual communication session & transfer of data between them.

The following data shows what happens during SSL handshake



1. Client sends a request to the server for a secure session. The server responds by sending X.509 certifica to the client.

2. Client receives server's X.509 certificate.

3. The client authenticates the server, using a list of known certificate authorities.

4. Client generates a random symmetric key & encrypts it using server's public key.

5. Client & server both know the symmetric key & can use SSL encryption.

classmate
Date
Page

19115071

Ans 10: Caching is senseless because the tuple will have been removed from the JavaSpace when it was returned; it is ready for the client to keep. The main idea behind caching is to keep data local to avoid another server access. In the case of Jini, a JavaSpace is often used to explicitly synchronize processes. Caching doesn't play a role when process sync is needed but now for the case that a client caches the results returned by a lookup service. This is a completely different situation. The lookup service stores the information on the whereabouts of services. In this case, it may indeed make sense for a client to cache previously returned results & try to contact the returned services before going to lookup service again.