CS550 Homework #3

Vaishnavi Manjunath
Kalpana Pratapaneni
Kavya Ravella

5.5 Outline an efficient implementation of globally unique identifiers.

Definition:

A globally unique identifier (GUID) is a 128-bit number created by the Windows operating system or another Windows application to uniquely identify specific components, hardware, software, files, user accounts, database entries and other items. GUIDs are part of the universally unique ID (UUID) standard that is used in Windows and Windows applications. Different kinds of objects have different kinds of GUIDs - for instance, a Microsoft Access database uses a 16-byte field to establish a unique identifier for replication. The problem with UUIDs is that they are very big in size and don't index well. When your dataset increases, the index size increases as well and the query performance takes a hit.

Existing Example:

MongoDB's Object IDs are 12-byte (96-bit) hexadecimal numbers that are made up of -

1) a 4-byte epoch timestamp in seconds.

2) a 3-byte machine identifier.

3) a 2-byte process id

4) a 3-byte counter, starting with a random value.

Implementation:

GUID identifiers can be generated locally as well. Take the network address of the machine where the identifier is generated, append the local time to that address, along with a generated pseudo-random number. Although, in theory, it is possible that another machine in the world can generate the same number, chances that this happens are negligible.

6.1 Name at least three sources of delay that can be introduced between broadcasting the time over the network and the processors in a distributed system setting their internal clocks.

First delay can be considered as atmosphere as we have signal propagation delay. Second, signal collision delay while the machines with the radio station WWV receivers compete to get on the Ethernet. Third, there is packet propagation delay on the local-area network. And also, there is a delay in each processor after the packet arrives, due to interrupt processing and internal queueing delays.

6.2 Consider the behavior of two machines in a distributed system. Both have clocks that are supposed to tick 1000 times per millisecond. One of them actually does, but the other ticks only 990 times per millisecond. If UTC updates come in once a minute, what is the maximum clock skew that will occur?

The second clock ticks 990*1000 = 990,000 times per second.
Thus, we can say that, it is giving an error of 10 msec per second.
Thus, in a minute it will be 10*60 = 600 msec per minute.
Another way of looking at it is that the second clock is one percent slow.
So, after a minute the maximum clock off by 0.01 × 60 sec, or 600 msec.

6.3 One of the modern devices that have (silently) crept into distributed systems are GPS receivers. Give examples of distributed applications that can make use of GPS information.

GPS receivers have made a huge impact on our everyday life.

One of the examples is GPS based body-area networks. The sports and health care have introduced this to allow a person to keep track of his pace while exercising an outdoor sport. Data like pace, heart rate, number of steps are received from this device which can be hooked up to a computer and further analysis can be done.

Another example is maps. It is interfaced with standalone GPS devices and mobile technology to provide turn-by-turn directions. They are also used in car-navigation equipment. Finding a location, the distance to a location, estimated time of arrival, the fastest route and many more details can be received through this.

6.7 Consider a communication layer in which messages are delivered only in the order that they were sent. Give an example in which even this ordering is unnecessarily restrictive.

Transfer of an image doesn’t necessarily require ordering.

Multimedia data like an image is transmitted in the form of consecutive packets. During transmission, a few packets might get lost. So, each block is identified by its position in the original image. In this case, FIFO ordering is not required. The receiver will place the block in the current position.

7.2 / 7.7 Explain in your own words what the main reason is for actually considering weak consistency models. It is often argued that weak consistency models impose an extra burden for programmers. To what extent is this statement actually true?

Weak consistency model is considered when executing parallel programming and for efficient replication.

Efficient replication can be done only if avoid global synchronizations. This can be achieved by loosening the consistency constraints.

In weak consistency models, if a variable change, it is not necessary to inform all processes.

The weak consistency models may or may not be a burden.

Synchronization mechanisms like locks or transactions are used by programmers to protect shared data. The main idea is that they require a coarser gain of concurrency than one offered at the level of only read and write operations. However, it is expected that operations on synchronization variables adhere to sequential consistency.

7.9 What kind of consistency would you use to implement an electronic stock market? Explain your answer.

Casual consistency is probably enough. Casual consistency represents a weakening model of sequential consistency in that it makes a distinction between events that are potentially casually related and those that are not.

Casual consistent is used to implement electronic stock market by following below condition:

Writes that are potentially casually related must be seen by all processes in the same order. Concurrent writes may be seen in a different order on different machines.

The issue is that reactions to variations in stock values should be consistent. Changes in stocks that are sovereign can be seen in unlike orders.

7.10 Consider a personal mailbox for a mobile user, implemented as part of a wide-area distributed database. What kind of client-centric consistency would be most appropriate?

All of them, actually. Whatever changes might happen the owner should always see the same mailbox, no matter whether he is reading or updating it. In fact, the simplest implementation for such a mailbox may well be that of a primary-based local-write protocol, where the primary is always located on the user's mobile computer.

7.14 We have stated that totally ordered multicasting using Lamport's logical clocks does not scale. Explain why.

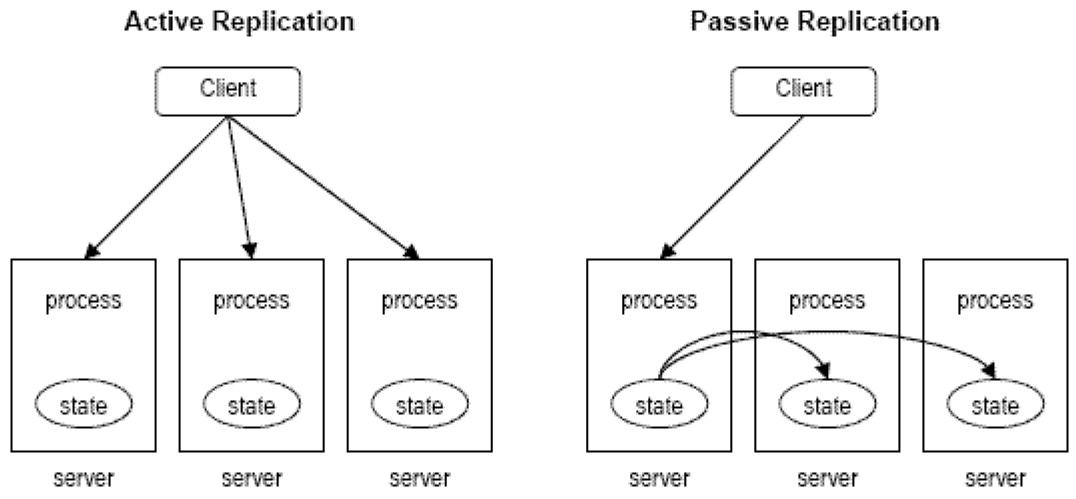
In distributed systems, physical clocks are not always precise, so we can't rely on physical time to order events. Instead, we use logical clocks to create a partial or total ordering of events. Lamport's logical clocks are local to each process or servers. They do not measure real time instead they measure only "events". They are useful for totally ordering transactions, by using logical clock values as timestamps.

Lamport’s way totally ordered multicasting requires that all servers are up and running, effectively hindering performance when one of them turns out to be slow or has crashed. This will have to detected by all the servers. As the number of servers grows, this problem is serious.

7.18 For active replication to work in general, it is necessary that all operations be carried out in the same order at each replica. Is this ordering always necessary?

In this distributed systems research area replication is mainly used to provide fault tolerance. The entity which is replicated is actually a process. Two replication strategies have been used in distributed systems.

1. Active replication
2. Passive replication



In active replication each client request is processed by all the servers.

The ordering during replication is not always necessary. Consider read operations that operate on nonmodified data or cumulative write operations. In principle, such situations allow ordering to be different at different replicas. However, it can be hard or impossible to detect whether, for example, two write operations are commutative.