Assignment (Lec. 13 & 14): Distributed Operating System Principles (COP5615)

Name: Vikas Chaubey**, UFID:** 3511 5826**, Email:** vikas.chaubey@ufl.edu

1) Monotonic Reads & Writes: The main reason to choose distributed systems is the benefits like scalability, locality and availability. Since processes are distributed on multiple machines in distributed system reliability and performance of the system is a big challenge hence replication of data is used in order to increase the reliability and performance of the system. **Monotonic reads and writes** are part of **Client centric consistency** models in distributed systems, which ensures the sequential consistency among the concurrent operations.

a) **Monotonic Read:** Distributed systems are group of processes on machines working concurrently and together in order to achieve a common goal. Different processes from different machines can update the same data item, this can create inconsistencies when the data is read. Monotonic data reads ensure that any given particular time the user process which is reading the data sees all updates made in the data item at any given point of time without the concern of from which server the reading is taking place. For example, if a data item is read by a process then for any successive read operations made by that process on that data item will always return most recent value and it will not be able to read the older values of that data item. Another example is when a person is reading emails while he is travelling or on the move then each time, he connects to a different email server the server fetches the most recent updates from the previous server that person was connected to.

b) **Monotonic Write:** Monotonic writes ensures that the write operation by a process on any data item is completed before any other successive write operation attempt made by the same process. In other words, if a process makes two successive writes to the same data item then all processes will see the first data write before they see the second data write made to the data item. Monotonic writes are applicable to the writes made by the same process, they are not applied on the operations performed by different processes. For example, monotonic writes are applicable in maintaining correct order of versions of replicated files in distributed system.

c) **Read Your Writes Consistency:** This consistency ensures that if a data item is updated by a process then this value will be available for all successive read operations made by that process on that data item.

d) **Writes Follows reads consistency:** This consistency ensures that updates on an data item are propagated after performing the previous read operations by the same process.

2) Content Delivery Network (CDN) for content Distribution: In a distributed system content delivery networks are networks which constitute different geographically distributed servers. The goal of having **CDN** is to provide high availability and high performance, this is accomplished by distributing the service spatially with respect to the end user or process which is making that service request. In CDN’s machines which are physically nearest to the request maker respond first. caching is another important task which Content Delivery Networks perform. Usually caching is used to improve read and write performance. For a query request CDN’s first explore the caches to find whether the requested content is available in cache, if it’s available then response is sent. If it’s not found, then CDN redirects the request to the nearest possible server which has the content.

What is Cache Consistency?

Caching can improve read and write performance of distributed system significantly, but it also introduces overhead complexity of ensuring consistency. Cache consistency requires that all the writes which are made to a memory location are done in some sequential order. Cache consistency is a type of weaker consistency than the process consistency.

Cache Consistency Protocols:

1) **Primary based protocols:** Class of consistency protocols which are simpler to implement. For example, sequential ordering is one of the primary based protocols. In these types of protocols each data item in data store has an associated primary which coordinates the write operations on that data item.

2) **Read Write Protocols:** This type of protocol is primary based protocol and it also supports replication. In this protocol the read operations are performed locally while the write operations are forwarded to a single server.

3) **Local Write Protocols:** In this Protocol the primary copy of data item is moved among the different processes, if processes intend to make an update to the copy then they can do so. Hence in this protocol the write operation could be performed locally, and each process can read the local copy of data item as well. When any process makes an update then the update is forwarded among all replicas and all processes perform the update locally.

4) **Quorum Based Protocols:** In this approach voting is used among processes to perform read/write operations. In this protocol a client requests several other servers to do read or write operation on data item. Client must send the request to at least a fixed number of servers (Total number of servers in cluster /2 + 1) in order to make an agreement to do write operation on data item. Similarly, for read operations the client sends requests to (Total number of servers in cluster /2 + 1) servers in order to receive the associated version number of data item , if all version numbers received by the client are most recent then read operation completes.

3) Fault Tolerance: Fault tolerance in the distributed system refers to the ability of the system to provide the expected service despite the presence of certain failures in the system by exploiting **redundancies in the space and time**. Even if the operational quality of the system reduces then in a properly designed fault tolerant system it reduces proportional to the severity of the failure. When a component in a distributed system is not performing up to its specification then it is considered to be a failure.

Dependability: a component in the distributed system might depend on service of another component. Hence the correctness of component's behavior could be interdependent.

Requirements related to dependability:

1) Availability 2) Reliability. 3) safety 4) Maintainability

Failure Models:

1) **Crash Failure:** server halts but after some time resumes its service.
2) **Omission failure:** server fails to receive and send messages.
3) **Timing failure:** server response lies outside a given time frame.
4) **Response failure:** server response is incorrect, or value of response is wrong
5) **Arbitrary failure:** server response is arbitrary

Requirements of fault tolerant Model:

1) There should not be a single point of failure
2) The failing component should be isolated
3) Propagation of the failure should be contained
4) Availability of means to avert the system to a previous state prior to failure.

What is triple Modular System?

It is a fault tolerant form of N- Modular redundancy system. In this system Three systems works on a process together. The output of the process is decided by the majority voting among three systems, this way a single output is generated. If any system fails to function properly then other two systems can do corrections and mask the fault in the system.

References:

1) https://en.wikipedia.org/wiki/Consistency_model#Read-your-writes_consistency
2) https://jepsen.io/consistency/models/monotonic-writes
3) https://geode.apache.org/docs/guide/111/managing/monitor_tune/cache_consistency.html
4) https://www.webopedia.com/TERM/C/CDN.html#:~:text=CDN%20is%20short%20for%20content,and%20the%20content%20delivery%20server.
5) https://geode.apache.org/docs/guide/111/managing/monitor_tune/cache_consistency.html
6) https://www.researchgate.net/publication/221225499_Research_on_Triple_Modular_Redundancy_Dynamic_Fault-Tolerant_System_Model