Assignment 2

1. Discuss client centric consistency models.

Client-centric consistency models are approaches used in distributed systems to ensure consistency of data perceived by clients accessing the system. These models prioritize the consistency requirements of individual clients or client sessions, tailoring the consistency guarantees to meet specific application needs and trade-offs.

1. Sequential Consistency (SC):
   * Each client sees the system as if operations are executed in the order they were issued by that client.
   * Guarantees that operations from a single client are observed in the same order by all clients.
   * However, it does not enforce a global total order of operations across all clients.
2. Causal Consistency (CC):
   * Preserves causal relationships between related operations.
   * If operation A causally precedes operation B, then all clients must see A before B.
   * Allows concurrent operations to be seen in different orders on different clients, as long as they are causally unrelated.
3. FIFO Consistency:
   * Ensures that operations from each client are seen in the order they were issued by that client.
   * Unlike sequential consistency, FIFO consistency does not require a global order of operations.
4. Monotonic Reads/Writes:
   * Guarantees that if a client reads or writes a value, subsequent reads by the same client will never observe earlier values.
   * Provides a form of consistency by ensuring that clients always see the most recent value they have written or observed.
5. Read Your Writes Consistency:
   * Ensures that once a client writes a value, it will always read that value or a more recent one.
   * Useful for maintaining session consistency in systems where clients interact with the same data across multiple operations.
6. Monotonic Reads with Staleness:
   * Allows clients to observe stale data but ensures that they will never observe older versions of data after seeing a more recent version.
   * Balances consistency and availability by allowing temporary staleness to improve performance.

Client-centric consistency models provide a range of options for developers to choose from based on the specific requirements of their applications. By tailoring consistency guarantees to individual clients or sessions, these models offer flexibility in managing the trade-offs between consistency, availability, and partition tolerance in distributed systems.

2. Write a note on file caching schemes.

File caching schemes are techniques used to improve the performance of file access operations by storing frequently accessed data in a cache memory. These schemes aim to reduce disk I/O operations and latency by serving read requests from the cache instead of accessing data directly from the underlying storage device. Various caching strategies are employed to manage the cache contents and optimize performance based on access patterns and resource constraints.

1. Least Recently Used (LRU):
   * Evicts the least recently accessed files from the cache when the cache reaches its capacity limit.
   * Assumes that recently accessed files are more likely to be accessed again in the near future.
2. Most Recently Used (MRU):
   * Evicts the most recently accessed files from the cache when the cache is full.
   * Based on the principle that recently accessed files are likely to be accessed again soon.
3. First-In-First-Out (FIFO):
   * Evicts the oldest files that were added to the cache when the cache is full.
   * Follows a strict queue-based eviction policy, regardless of access frequency or recency.
4. Random Replacement:
   * Selects files to evict from the cache randomly when the cache is full.
   * Simple but lacks the efficiency of more sophisticated replacement policies.
5. Adaptive Replacement Cache (ARC):
   * Dynamically adjusts the cache size and replacement policy based on recent access patterns.
   * Maintains separate lists for frequently and infrequently accessed files, adapting to changing workload characteristics.
6. Write-Through and Write-Behind Caching:
   * Write-through caching immediately writes updates to both the cache and the underlying storage, ensuring data consistency but potentially introducing write latency.
   * Write-behind caching delays writing updates to the underlying storage, improving write performance but risking data loss in case of system failure.

File caching schemes play a critical role in optimizing file access performance in various computing environments, including operating systems, databases, web servers, and distributed file systems. By strategically managing cache contents and balancing cache utilization with storage capacity, these schemes enhance overall system responsiveness and throughput, benefiting users and applications alike.