**Neha Maddali**

1. **(12 pts.) Describe briefly the concept of *false cycle* in distributed transaction management.**

False cycle in a distributed transaction management refers to a situation where a distributed transaction appears to be stuck in an ongoing cycle of rollback and retry, even when there is not any actual conflict or failure in the system. Coordinating the outcome of transactions become more complex when transactions may scan multiple databases or recourses. False cycles can impact system performance and availability, as resources may be locked, and transactional integrity may be compromised. But this can be prevented with careful design and implementation of techniques like timeout adjustments, and retry limits.

1. **(12 pts.) Describe briefly the (difference between) *transformations* and *actions* in Spark.**

In Spark, transformations and actions are two fundamental operations used for processing data in distributed computing environments. Transformations are operations that produce a new distributed collection or DataFrame from an existing one without executing any computation. They are not immediately computed when called, rather their execution is deferred until an action is performed. Transformations are used to define the data processing operations to be performed on the RDD or DataFrame. Actions are operations that return a value or produce a side-effect like writing data to the disk or printing it to the console. Actions trigger the execution of the transformations that have been defined on the RDD or DataFrame and the results are returned to the driver program or stored in external storage. Actions are used to trigger the actual computation and obtain results from the RDD or DataFrame.

1. **(15 pts.) What are the benefits and drawbacks of the *Majority Protocol* for distributed lock management?**

The Majority Protocol is a fault-tolerant, scalable and simple distributed lock management protocol. But, it can introduce performance delays, is vulnerable to complexity, and can impact availability in certain situations. So, it’s important to carefully consider the trade-offs and choose the appropriate lock management protocol based on the specific requirements and characteristics of the distributed system.

1. **(15 pts.) Describe briefly the CAP theorem. Provide an example of a NoSQL database from the “AP” spectrum.**

The CAP theorem is a principle of distributed computing that states that it is impossible for a distributed system to simultaneously provide all three guarantees of consistency, availability and partition tolerance. CAP theorem says that a distributed system can only provide two of these guarantees at the same time. This means that in the face of a network partition, the system must choose between consistency and availability. An example of a NoSQL database from the “AP” spectrum of the CAP theorem is Apache Cassandra. Cassandra is a highly scalable and distributed NoSQL database that is designed to handle large amounts of data across multiple nodes in a distributed cluster. It prioritizes availability and partition tolerance over consistency, so it is an ideal choice for applications that require high availability and fault tolerance. In Cassandra, updates and writes can be made to different nodes in the cluster without waiting for a global consensus so it allows for high availability and low latency operations. But because of its eventual consistency model, it may experience temporary inconsistencies in data across nodes during network partitions, prioritizing availability and partition tolerance over consistency,

1. **(15 pts.) Describe briefly the concept of a *polystore*.**

A polystore is a data management approach that involves using multiple data storage technologies to coexist within a single system. Different data models or storage technologies are used to store and manage different types of data in a heterogeneous manner. By allowing for multiple data models or storage technologies to be used within a single system, a polystore can provide more flexibility, scalability, and performance for handling diverse data types.

1. **(12 pts.) Describe briefly the difference between *streaming* algorithms and *online* algorithms.**

Streaming algorithms are designed to process data that arrives in a continuous stream where the data is typically too large to be stored in its entirety, and may not be available for repeated processing. These algorithms process data element by element as it arrives and makes decisions or produce results based on the current elements in the stream. But, these algorithms have limited memory requirements and are optimized for processing data in a single pass without the need for storing and revisiting part data.

Online algorithms are designed to process data that arrives incrementally or in small batches, but with the possibility of revisiting past data. These algorithms make decisions or produce results based on the current data, but also take into account the past data that may be stored or available for reference. These algorithms typically have more flexibility in terms of accessing past data and optimizing decision-making based on historical information.

1. **(19 pts.) Consider the following schedule of 3 transactions accessing 3 different data-items: Would you say that this schedule is conflict-serializable (justify your answer)?**

Table

Description automatically generated

The schedule could be represented as a precedence graph. Each transaction is represented by a node, and there is a directed edge from transaction to another if there is a conflicting operation between them and the first transaction precedes the second transaction in the schedule. In the precedence graph, there are no cycles, which means that the schedule is conflict-serializable. A conflict occurs when two transactions access the same data item and at least one of the operations is a write. From the given schedule these are the conflicts:

* TR1 reads B, while TR2 writes B
* TR2 writes A, while TR1 reads A
* TR2 writes C, while TR3 reads C
* TR3 writes C, while TR2 writes C
* TR3 reads B, while TR1 writes B
* TR3 writes B, while TR2 writes B

The given schedule is conflict serializable based on these conflicts and as there are no cycles in the precedence graph.