**Northeastern University – Silicon Valley**

CS 6650 Scalable Dist Systems

**Homework #4** [100 points]

**Name: Nisarg Patel**

**Email:** [**patel.nisargs@northeastern.edu**](mailto:patel.nisargs@northeastern.edu)

***INSTRUCTIONS: Please provide clear explanations in your own sentences, directly answering the question, demonstrating your understanding of the question and its solution, in depth, with sufficient detail. Submit your solutions [PDF preferred]. Include your full name. Do not email the solutions.***

Answer the following questions using explanation and diagrams as needed. No implementation needed.

Study **Chapter 16 from** Coulouris Book

1. 16.1 [10 points]

**Ans:**

(i) the server can reply immediately, telling the client to try again later

Advantages: The client(worker) does not have to wait for master to add a task in the bag.

Drawbacks: Client have to make multiple calls to the master if the bag is empty. This method can lead to starvation(unfairness) in the waiting worker as the request from other workers may be processed before the worker sends request again.

(ii) make the server operation (and therefore the client) wait until a task becomes available.

Advantages: Worker have to make a single call to the master to get a task. Fairness can be achieved using this

Disadvantages: If no task is available to take, the worker has to wait till master adds a task in the bag.

(iii) use callbacks.

Advantages: //

Disadvantages: //

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. 16.2 [10 points]

**Ans:**

If T -> U

|  |  |  |
| --- | --- | --- |
| Interleaving – A | Interleaving – B | Interleaving – C |
| T1: x = read(j) | T1: x = read(j) | T1: x = read(j) |
| T2: y = read(i) | U1: x = read(k) | T2: y = read(i) |
| U1: x = read(k) | T2: y = read(i) | T3: write(j, 44) |
| T3: write(j, 44) | T3: write(j, 44) | U1: x = read(k) |
| T4: write(i, 33) | T4: write(i, 33) | T4: write(i, 33) |
| U2: write(i, 55) | U2: write(i, 55) | U2: write(i, 55) |
| U3: y = read(j) | U3: y = read(j) | U3: y = read(j) |
| U4: write(k, 66) | U4: write(k, 66) | U4: write(k, 66) |

If U -> T

|  |  |  |
| --- | --- | --- |
| Interleaving – D | Interleaving – E | Interleaving – F |
| U1: x = read(k) | U1: x = read(k) | U1: x = read(k) |
| T1: x = read(j) | U2: write(i, 55) | U2: write(i, 55) |
| U2: write(i, 55) | U3: y = read(j) | U3: y = read(j) |
| U3: y = read(j) | T1: x = read(j) | T1: x = read(j) |
| T2: y = read(i) | T2: y = read(i) | T2: y = read(i) |
| U4: write(k, 66) | U4: write(k, 66) | T3: write(j, 44) |
| T3: write(j, 44) | T3: write(j, 44) | T4: write(i, 33) |
| T4: write(i, 33) | T4: write(i, 33) | U4: write(k, 66) |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Concurrency Control and Java Apps
2. Explain the 3 different concurrency control methods (Ch 16 Coulouris Book – Locks based, Optimistic and Time-stamp Ordering) brifely. Compare and contrast how they achieve concurrency control. [10 points]

**Ans:**

**Locks:** In locks-based concurrency control, the object that is used during a transaction is locked by the server to perform operations and other server requests that tries to access that object must wait till the object is unlocked. In strict two-phase locking, the locks are held when the object is first accessed during the transaction and released when the transaction either commits or aborts.

**Optimistic Concurrency Control:** It assumes that the likelihood of two or more transactions that tries to access the same object is rare (hence optimistic). It consists of 3 phases:

**(i) Working Phase:** Each transaction proceeds without any wait and stores the updates.

**(ii) Validation Phase:** When transaction ends, the operations are validated whether it conflicts with other transactions or not. If conflict, then a resolution has to be made and some transaction has to be aborted in most of the cases.

**(iii) Update Phase:** The changes made during working phase to all the objects are made permanent.

**Timestamp Ordering:** During start of transaction, it is assigned a timestamp. It follows the rule that the write operation of a timestamped transaction is valid only if the previous read or write operation on that object is done by an earlier timestamped transaction. Similarly, read operation for a timestamped transaction is valid only if the previous write operation is done by an earlier timestamped transaction. If the transaction is failing the rule in any of the object, then it is aborted immediately.

**Comparing the three concurrency control methods:**

The two-phase locking and the timestamp ordering are pessimistic approaches and detects the conflicts as soon as the object is accessed. The optimistic concurrency on the other hand is optimistic in the way that the conflicts are detected close to the commit time.

The two-phase locking makes the transaction wait till the conflicting object is available to access. The timestamp ordering and the optimistic concurrency aborts the transaction if conflict is detected.

The two phase locking method includes overhead of lock object. Furthermore, it includes waiting of processes, it is susceptible to deadlocks. Thus, deadlock prevention measures like timeout and deadlock detection also increasing time overhead.

Optimistic concurrency control and timestamp ordering are both time and space efficient since they abort immediately if it has to, but can result in starvation. Furthermore, if there are long transactions, and it fails, then substantial amount of work has to be repeated (more in case of optimistic as all transaction is processed before validation phase).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. **Java Concurrency In Practice** by Brian Goetz is the practice manual for Dist Systems. Study Chapter 6 (Task Execution) which is a small chapter.

Identify 5 different programming devices (e. g. Tasks, Exec Framework) used in Java for Task Execution. Then, briefly explain how you might use these to implement a basic

Locks based concurrency control in a simple Web App. [15 points]

**Ans:**

**(i) Tasks in Threads:**

**(ii) Executor Framework**

**(iii) ThreadPool**

**(iv) Result-Bearing Tasks**

**(v) Completion Service**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Refer to Oracle Database Administrator’s Guide page 29-23 [15 points]

**Transaction Processing in a Distributed System**

<https://docs.oracle.com/html/E25494_01/ds_concepts004.htm> But get the PDF, it is easier to read.

What are distributed transactions in Oracle DB? How are the different from Remote Transactions? Give examples. How does Oracle DB use Naming service and 2-Phase Atomic Commit Protocols to manage distributed transactions?

**Ans:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Study Alibaba Fescar note. How does Fescar manage distributed transactions? What protocols are used? Describe in detail. How does it avoid Deadlock? [20 points]

**Ans:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Transaction API (JTA) allows applications to perform distributed transactions. For a small e-commerce retail store, explain a design for implementation using JTA. Show the key Classes and how you use them to resolve concurrency issues and deadlock. [20 points]

**Ans:**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_