Concurrency Homework › sudha.anil@gmail.com

Full Name:	Anil Gathala
Email:	sudha.anil@gmail.com
Test Name:	**Concurrency Homework**
Taken On:	13 Jul 2018 06:49:19 PDT
Time Taken:	66301 min 30 sec/ 43200 min
Work Experience:	> 5 years
Invited by:	Soham
Invited on:	11 Jul 2018 07:31:58 PDT
Tags Score:	

100%
70/70

scored in **Concurrency Homework** in 66301 min 30 sec on 13 Jul 2018 06:49:19 PDT

Recruiter/Team Comments:

No Comments.

	Question Description	Time Taken	Score	Status
Q1	Rate Limiter › **Multiple Choice**		5/ 5	
Q2	Request Dispatcher › **Multiple Choice**		5/ 5	
Q3	Concurrent Hashmap › **Multiple Choice**		5/ 5	
Q4	Timer › **Multiple Choice**		5/ 5	
Q5	Dictionary › **Multiple Choice**		5/ 5	
Q6	Bounded Hash Set › **Multiple Choice**		5/ 5	
Q7	Read-Write Map › **Multiple Choice**		5/ 5	
Q8	Lock Order Deadlock › **Multiple Choice**		5/ 5	
Q9	Implement Thread Local Storage › **Multiple Choice**		5/ 5	
Q10	Implement Fair Thread Scheduling › **Multiple Choice**		5/ 5	
Q11	Implement a dispatch execution class with sub-system locks › **Multiple Choice**		5/ 5	
Q12	Create a thread pool and worker queues › **Multiple Choice**		5/ 5	
Q13	Implement a connection pool › **Multiple Choice**		5/ 5	
Q14	Odd Even › **Multiple Choice**		5/ 5	

QUESTION 1

Correct Answer

Score 5

Rate Limiter › Multiple Choice

QUESTION DESCRIPTION

Design a rate limiter for concurrent API requests that can be used for throttling API requests from any given API client based on configured quotas and security policy.

(Question is purposely vague. Assume a basic rate limiting policy, like token bucket. In its simple form, one can also ask you to write code for this question. See: http://stackoverflow.com/questions/667508/whats-a-good-rate-limiting-algorithm)

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 2

Correct Answer

Score 5

Request Dispatcher › Multiple Choice

QUESTION DESCRIPTION

Design a request dispatcher for a web-server that accepts and processes incoming web requests concurrently and responds synchronously.
Secondary concern: How can this design be modified to support asynchronous response?

(Question is purposely vague. Make your best assumptions, keeping in mind this is an interview question)

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 3

Correct Answer

Score 5

Concurrent Hashmap › Multiple Choice

QUESTION DESCRIPTION

Implement a concurrent hash-based map using lock striping.

Note: Apply a locking strategy that offers better concurrency and scalability; instead of synchronizing every method on a common lock, restricting access to a single thread at a time, utilize a finer-grained locking mechanism called lock striping to allow a greater degree of shared access.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 4

Correct Answer

Score 5

Timer › Multiple Choice

QUESTION DESCRIPTION

Design a “timer” scheduler that can take a number of specified tasks and schedule them for execution after specified time-interval delay. What would be some of the design considerations to ensure efficiency, resiliency and support for monitoring progress?

It is very easy to imagine a complex scheduling system when solving this problem (think Cron + Stopwatch + Timer). But keep it limited in scope.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 5

Correct Answer

Score 5

Dictionary › Multiple Choice

QUESTION DESCRIPTION

Design an “efficient” simple dictionary that can be accessed by multiple concurrent users and editors simultaneously. While the users are looking up for definitions for specific words, the editors are either creating definitions for new words, or updating the ones for existing words. There are many more users (readers) of the dictionary than there are editors.

(The question is purposely vague. Make your best assumptions, that are also reasonable for an interview question)

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 6

Correct Answer

Score 5

Bounded Hash Set › Multiple Choice

QUESTION DESCRIPTION

Use a semaphore to implement a “bounded” hash set (or any collection, for that matter), a set that provides additional semantics of a blocking bounded collection to the underlying set.

Note: Here, we need to use a Semaphore to turn the underlying collection into a blocking bounded collection. The semaphore is initialized to the desired “bound” for the collection.
The underlying Set implementation knows nothing about the bound.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Complete
- [ ] Not Complete

No Comments

QUESTION 7

Correct Answer

Score 5

Read-Write Map › Multiple Choice

QUESTION DESCRIPTION

Implement a “Read-Write Map” leveraging an existing reentrant Read-Write lock so that it can be shared safely by multiple readers and writers, and yet prevent reader-writer or writer-writer conflicts.

Note: Mutual exclusion is a conservative locking strategy that prevents writer-writer and writer-reader overlap, but it also prevents reader-reader overlap. In many cases, data structures are “read-mostly”; in these cases, multiple readers should be able to access the data structure at once. This is what read-write locks allow: a resource can be accessed by multiple readers or a single writer at a time, but not both.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 8

Correct Answer

Score 5

Lock Order Deadlock › Multiple Choice

QUESTION DESCRIPTION

Demonstrate a lock-order deadlock using a code example and then induce simple lock ordering to avoid the deadlock.

Note: Think of threads as the nodes of a directed graph whose edges represent the relation “Thread A is waiting for a resource held by thread B”. If this graph is cyclical, there is a deadlock. This is likely to happen if two threads attempt to acquire the same set of locks in a different order.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 9

Correct Answer

Score 5

Implement Thread Local Storage › Multiple Choice

QUESTION DESCRIPTION

You are working on a compiler team. You are tasked with providing an api for the compiler to implement thread-local storage. Thread-local storage is equivalent to a per-thread static variable (seehttps://en.wikipedia.org/wiki/Thread-local_storage for more detailed explanation).

You must implement the following API for the compiler. NOTE this api is for the compiler's use, not the end user. That is why it is simpler than the standard TLS interface.

// Called once before any calls to GetThreadLocalStorage on any thread.
void InitThreadLocalStorage()
// id is a process-wide id to reference the storage. If the id does not exist,
// the storage is allocated to size cb. If id exists, return the corresponding
// thread-local variable.
void* GetThreadLocalStorage(int id, size_t cb)
// remove the storage allocated for the given id.
void RemoveThreadLocalStorage(int id)

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 10

Correct Answer

Score 5

Implement Fair Thread Scheduling › Multiple Choice

QUESTION DESCRIPTION

Your legacy code requires fair scheduling. You used to use the ReentrantLock with the fair schedule flag set to true. Legal just informed you that there was a patent case that was just won and all existing such implementations are patented. They need you to re-implement ReentrantLock using just the standard (unfair) mutex.

Create a class LegalReentrantLock with implements fair scheduling using mutex.

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 11

Correct Answer

Score 5

Implement a dispatch execution class with sub-system locks › Multiple Choice

QUESTION DESCRIPTION

You have a dispatch queue. This is a standard producer-consumer concurrent queue. All tasks in the queue have a Task interface (see below). The tasks have a LocksNeeded flag. There are three lock flags [FileSystem, Database, Model]. When a consumer thread requests a task to execute, the dispatch system will ensure the thread has the locks requested in the LocksNeeded flag set. Ensure that the system is efficient and free of deadlocks.

```
#define FLAGS_NO_LOCK 0
#define FLAGS_FILESYSTEM_LOCK 1
#define FLAGS_DATABASE_LOCK 2
#define FLAGS_MODEL_LOCK 4
interface Task
{
int GetFlags();
void Run();
}
```

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not complete

No Comments

QUESTION 12

Correct Answer

Score 5

Create a thread pool and worker queues › Multiple Choice

QUESTION DESCRIPTION

A thread pool is a group of pre-instantiated, idle threads which stand ready to be given work. These are preferred over instantiating new threads for each task when there is a large number of short tasks to be done rather than a small number of long ones. This prevents having to incur the overhead of creating a thread a large number of times.
Implementation will vary by environment, but in simplified terms, you need the following:
A way to create threads and hold them in an idle state. This can be accomplished by having each thread wait at a barrier until the pool hands it work. (This could be done with mutexes as well.)

A container to store the created threads, such as a queue or any other structure that has a way to add a thread to the pool and pull one out.
A standard interface or abstract class for the threads to use in doing work. This might be an abstract class called Task with an execute() method that does the work and then returns.

When the thread pool is created, it will either instantiate a certain number of threads to make available or create new ones as needed depending on the needs of the implementation.

When the pool is handed a Task, it takes a thread from the container (or waits for one to become available if the container is empty), hands it a Task, and meets the barrier. This causes the idle thread to resume execution, invoking the execute() method of the Task it was given. Once execution is complete, the thread hands itself back to the pool to be put into the container for re-use and then meets its barrier, putting itself to sleep until the cycle repeats.

[See: http://www.ibm.com/developerworks/library/j-jtp0730]

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 13

Correct Answer

Score 5

Implement a connection pool › Multiple Choice

QUESTION DESCRIPTION

Implement a simple and lightweight connection pool.

(If you're using Java, then you can make use of ConcurrentLinkedQueue)

[e.g. http://www.javacodegeeks.com/2013/08/simple-and-lightweight-pool-implementation.html]

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not Complete

No Comments

QUESTION 14

Correct Answer

Score 5

Odd Even › Multiple Choice

QUESTION DESCRIPTION

Create two threads. Let one thread print Odd numbers and the other thread print Even numbers.

* Assume natural numbers (1, 2, 3...), from 1 thru 100.
* Numbers must be printed in their natural order i.e. 1 and then 2 and then 3 etc.

Read: http://stackoverflow.com/a/30809023/327310

CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)

- [x] Completed
- [ ] Not complete

No Comments