**Senior Technical Lead -Veri5Digital**

1. A heavy file is being loaded in a function. If multiple threads, try calling the same

function, then the application crashes due to multiple objects being created and multiple threads loading the same file. If a single thread calls the function, then the application becomes slow as each thread is waiting for the previous thread to be completed.

Suggest a good approach to mitigate this issue.

**Ans:**

I will suggest Spring Batch processing within memory Database (H2) for this problem. Spring Batch provides functions for processing large volumes of data in batch jobs.

public largeFIleProcessor<Model> processor() {

return new FlatFileItemReaderBuilder<Model>()

.name("largeFileProcessor")

.resource(new ClassPathResource("Volts.csv"))

.delimited()

.names(new String[]{"property1", "property2"})

.lineMapper(lineMapper())

.fieldSetMapper(new BeanWrapperFieldSetMapper<Model>() {{

setTargetType(Model.class);

}})

.build();

}

name - Name of the ItemReader

resource - Specify path for the resource file to be read.

delimited - Builds delimited tokenizer.

names - Pass the fields that are to be read

lineMapper - Interface to map lines from file to domain object.

fieldSetMapper - Interface to map data obtained from a fieldset to an object.

You can also use

Files.*list*(Paths.*get*(".")).parallel().forEach(System.***out***::println);

1. What is false sharing and how to overcome it?

**Ans**:

"false sharing" happens when different memory address is assigned to the same cache line, writing data to one of it will cause another being kicked out of the cache. Result the kicked one have to reload the data again.

I really do not know the best solution but after studying I found certain notes to avoid this.

a. Use private data as much as possible.

b. Sometimes you can use [padding](https://en.wikipedia.org/wiki/Data_structure_alignment#Data_structure_padding) in order to align data, to make sure that no other variables will reside in the same cache that shared data reside.

Eg. You can add padding in string like [String.format()](https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#format-java.lang.String-java.lang.Object...-) method.

1. Multiple payment vendors are present in your application to serve payment requests.Suppose currently vendor A is being used and the vendor is to be changed to vendor B, try suggesting a good approach to do so without breaking the existing code.

**Ans:**

**Smart Routing pattern**

One of the solutions would be abstraction.

I would create a standard interface to deal with the common operations like makePayments etc. Have a set of domain objects that are passed into those methods.

For e. CardDetails, Address.

public interface PaymentMaster{

public PaymentResponse makePayment(CardDetails cardDetails, Address address);

}

class Payment1 implements PaymentMaster{

public PaymentResponse makePayment(CardDetails cardDetails, Address address){

///

}

}

class Payment2 implements PaymentMaster{

public PaymentResponse makePayment(CardDetails cardDetails, Address address){

///

}

}

etc.

Then use IoC (Inversion of Control) / DI (Dependency Injection) to decide which interface to use at runtime based on a provided config value.

1. Design a rest api for a job scheduler.

# Ans: REST API for job Scheduling

1. Create job scheduled

REQUEST

URI : http://{host}:{port}/schedulemanager/api/scheduler?accessToken=<accessToken>

Method: POST

Data: JSON

{......}

RESPONSE

Failure: HTTP CODE 400 or 404

Data: JSON

{

"errorMessage": "<error-message-string>",

"timestamp": <error-timestamp>

}

Success: HTTP CODE 201

Data: JSON

{.......}

2. Get job scheduled

REQUEST

URI: http://{host}:{port}/schedulemanager/api/scheduler/{jobName}?accessToken=<accessToken>

Method: GET

RESPONSE

Failure: HTTP CODE 400 or 404

Data: JSON

{..........}

Success: HTTP CODE 200

Data: JSON

{...........}

3. Get all job scheduled

REQUEST

URI: http://{host}:{port}/schedulemanager/api/scheduler?accessToken=<accessToken>

Method: GET

RESPONSE

Success: HTTP CODE 200

Data: JSON

{..............}

4. Update job scheduled

REQUEST

URI: http://{host}:{port}/schedulemanager/api/scheduler/{jobName}?accessToken=<accessToken>

Method: PUT

Data: JSON

{............}

RESPONSE

Failure: HTTP CODE 400 or 404

Data: JSON

{.............}

Success: HTTP CODE 200

Data: JSON

{..........}

5. Delete job scheduled

REQUEST

URI: http://{host}:{port}/schedulemanager/api/scheduler/{jobName}?accessToken=<accessToken>

Method: DELETE

RESPONSE

Failure: HTTP CODE 204

Success: HTTP CODE 200

1. Suggest an efficient way to handle large volume of traffic requests.

**Ans:**

The best solution will be

Autoscaling of servers and load-balancing:

Autoscaling is a process in Cloud computing which ensures that the correct number of instances is available to handle the load of the current application. It automatically increases the number of instances during high traffic to maintain the server performance and decrease the instances again as the traffic subsides.

A load balancer promptly performs the below functions:

* Dispenses client requests or network load resourcefully across the multiple servers.
* Assures high availability and reliability by sending requests only to servers that are online.
* Affords the flexibility to add or subtract servers as demand dictates.