How to answer frequently asked real-time project questions?

linkedin.com/vaishnavi-muralidhar

AGENDA:

1. Understand why project questions are important

2. Commonly asked project questions with solutions

3. Q & A

4. Connect with me

linkedin.com/vaishnavi-muralidhar

Project Questions:

1. Cluster configuration for your project

2. Common spark properties for high processing spark jobs

3. Checks after ingesting data from sources.

4. How business requirements are stated?

Cluster configuration:

Task:

1. You were given a task to develop a new reporting pipeline which consumes data from various sources, dimensions and fact tables.

2. This pipeline has to be scheduled after your SLA hours due to which it has to be scheduled separately.

3. After writing the logic, its time to test the pipeline end-to-end and validate whether data was flowing correctly.

4. You have to define the cluster configuration for this new pipeline. How do you go about it?

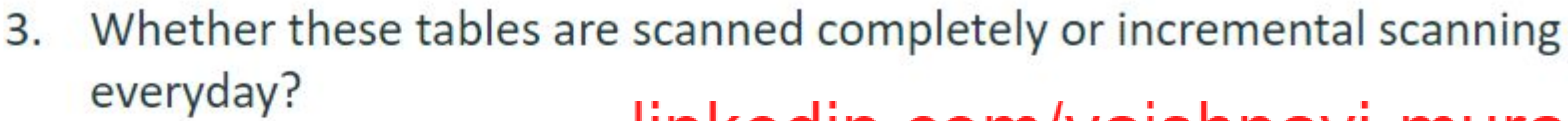
Cluster configuration:

Steps to answer:

Initial questions to ask yourself:

1. How many tables are involved (source tables, dims, facts) ?

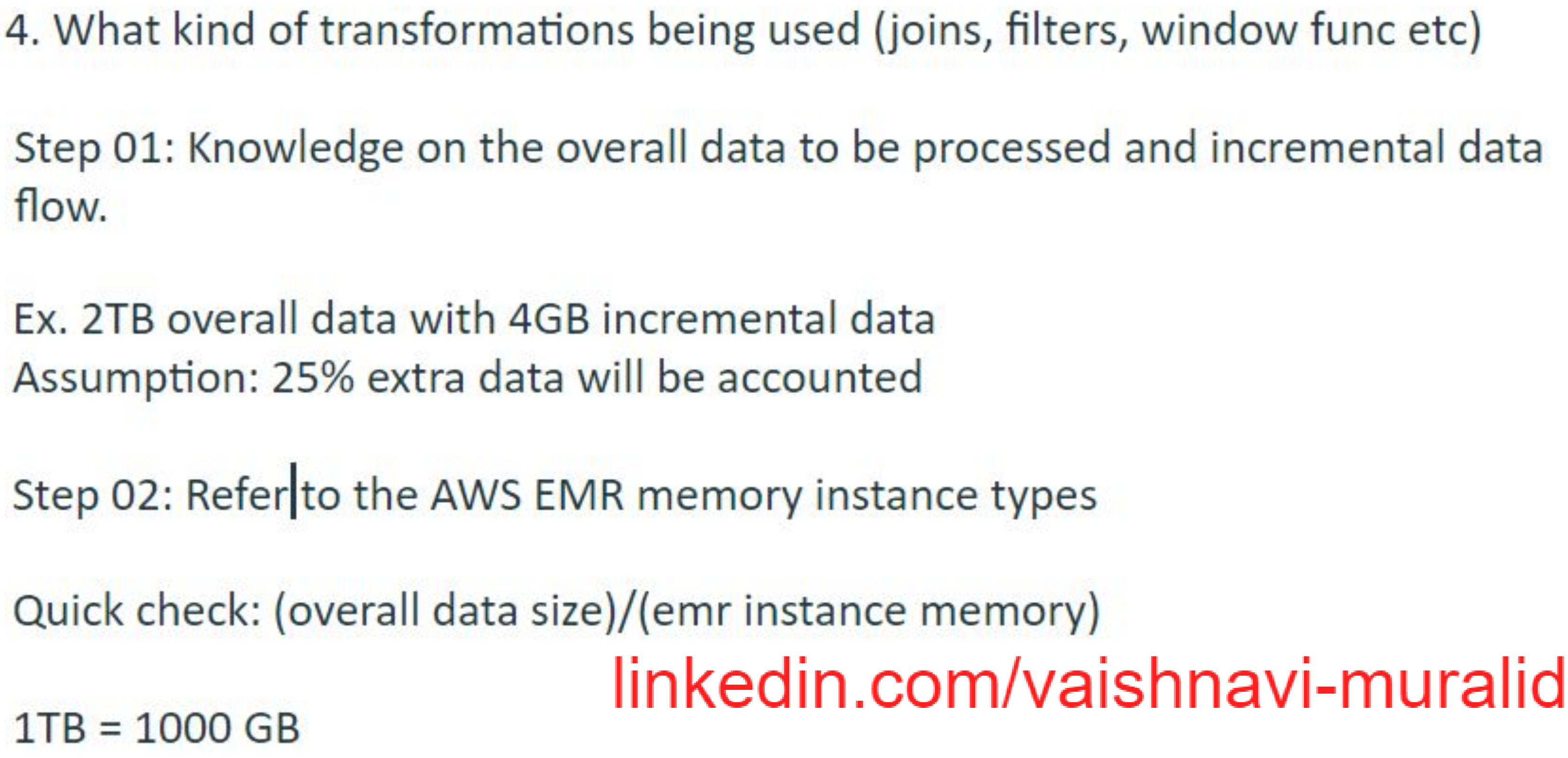
2. What is the average size of these tables?



3. Whether these tables are scanned completely or incremental scanning everyday?

linkedin.com/vaishnavi-muralidhar

Cluster configuration:



4. What kind of transformations being used (joins, filters, window func etc)

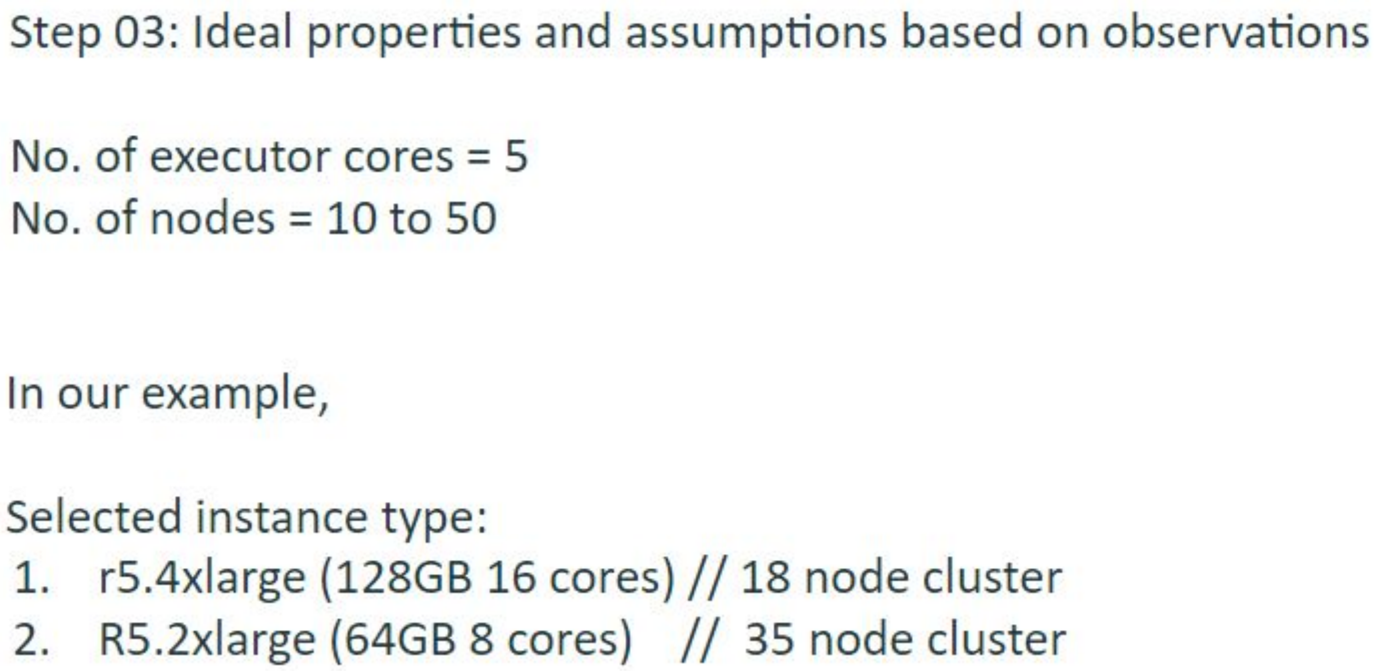
Step 01: Knowledge on the overall data to be processed and incremental data flow.

Ex. 2TB overall data with 4GB incremental data
Assumption: 25% extra data will be accounted

Step 02: Refer to the AWS EMR memory instance types

Quick check: (overall data size)/(emr instance memory)

1TB = 1000 GB



Step 03: Ideal properties and assumptions based on observations

No. of executor cores = 5
No. of nodes = 10 to 50

In our example,

Selected instance type:

1. r5.4xlarge (128GB 16 cores) // 18 node cluster
2. R5.2xlarge (64GB 8 cores) // 35 node cluster

linkedin.com/vaishnavi-muralidhar

R5.2xlarge EMR (64GB 8cores)	R5.4xlarge EMR (128GB 16cores)
Executors per node: 7/5 ~ 1 Usable memory: 64 - 10% = 57.6 GB Total node: 2250/57.6 ~ 39 Total executors: 39*1 = 39 Executor memory: 52 GB Memory overhead: 5GB (min 10% of usable memory) Driver memory: 52 GB Driver overhead: 5GB Total tasks: 39*7 = 273 Total memory of executor : (52+5) GB	Executors per node: 15/5 ~ 3 Usable memory: 128- 10% = 115.2 GB **Total node: 2250/115.2 ~ 20** Total executors: 20*3 = 60 **Executor memory: 34 GB (115/3) = 38.3** **Memory overhead: 4 GB (10% of usable mem)** **Driver Memory: 104GB (115 - driver overhead)** **Driver Overhead: 11GB (10% of 115)** Total tasks: 60*5 = 300 Total memory of executor : (34+4) GB

Which cluster to choose and what factors to consider?

Cost and Run Time

1. R5.4xlarge EMR : 1 node 0.252 USD/hourly

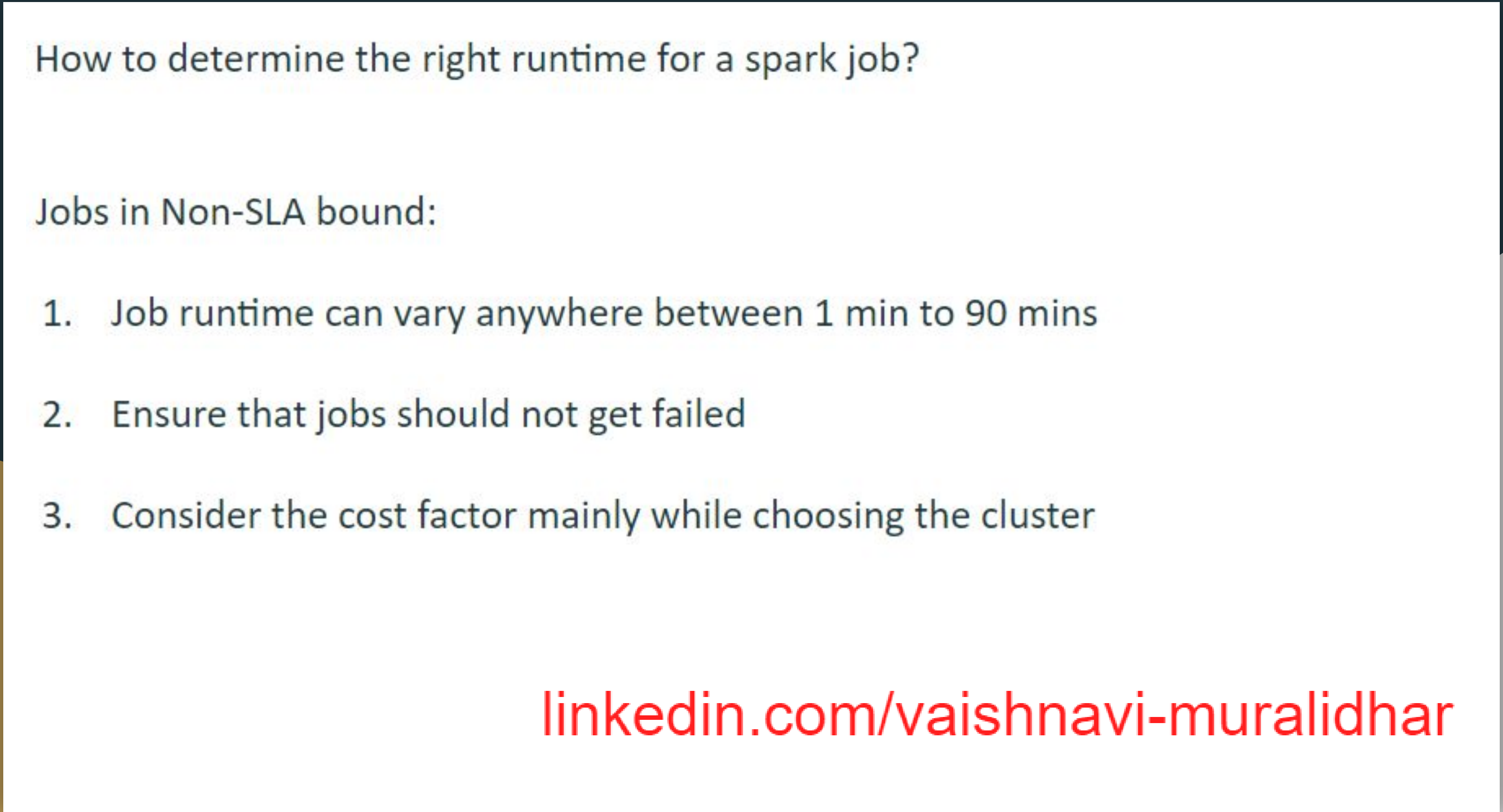
It takes 20 mins for job completion

Monthly cost: (21* (0.252/3) * 30) = $53

2. R5.2xlarge EMR : 1 node 0.126 USD/hourly

It takes 75 mins for job completion

Monthly cost: (40 * 0.1575 * 30) = $189

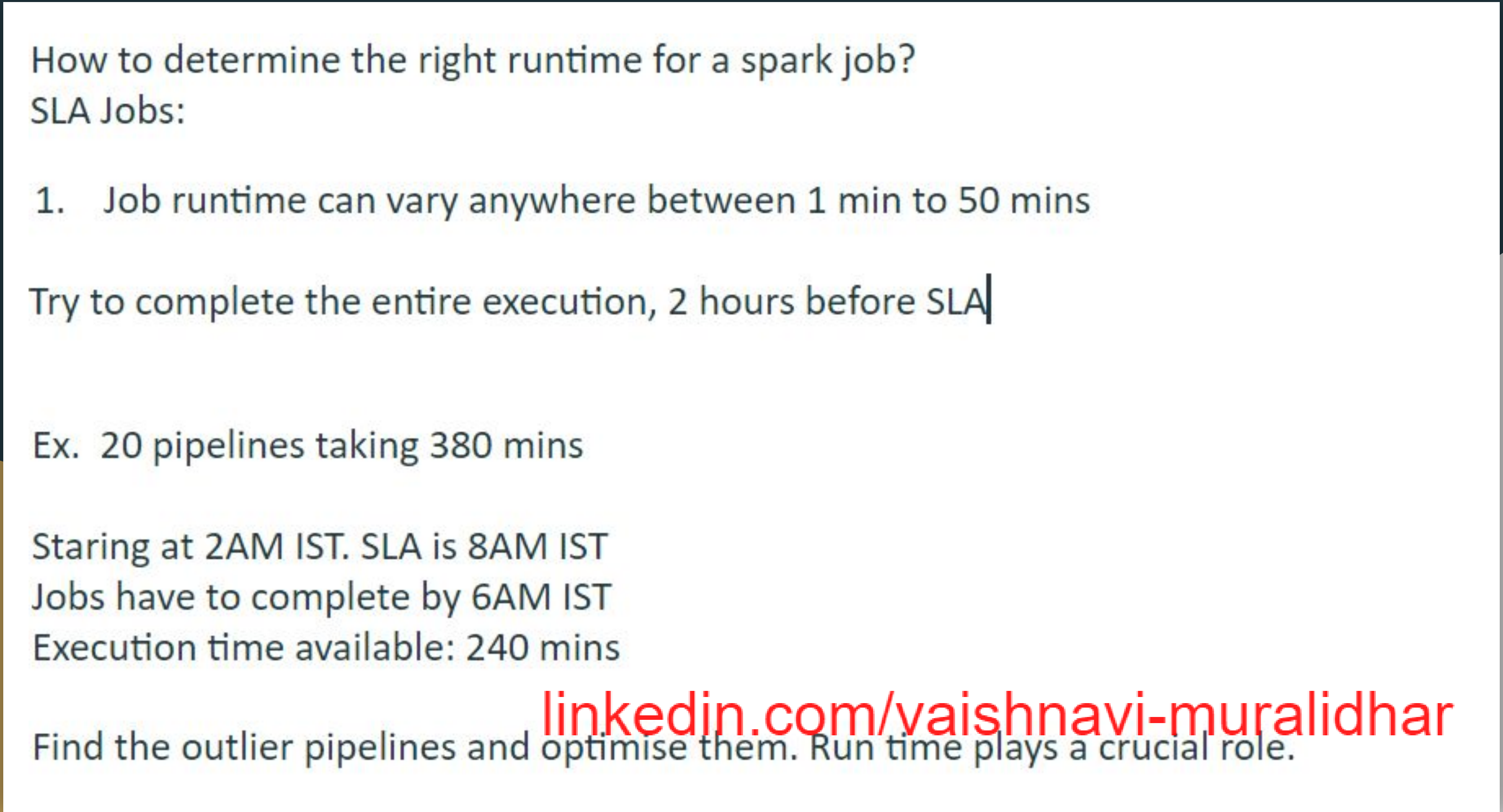


How to determine the right runtime for a spark job?

Jobs in Non-SLA bound:

1. Job runtime can vary anywhere between 1 min to 90 mins
2. Ensure that jobs should not get failed
3. Consider the cost factor mainly while choosing the cluster

linkedin.com/vaishnavi-muralidhar



How to determine the right runtime for a spark job?
SLA Jobs:

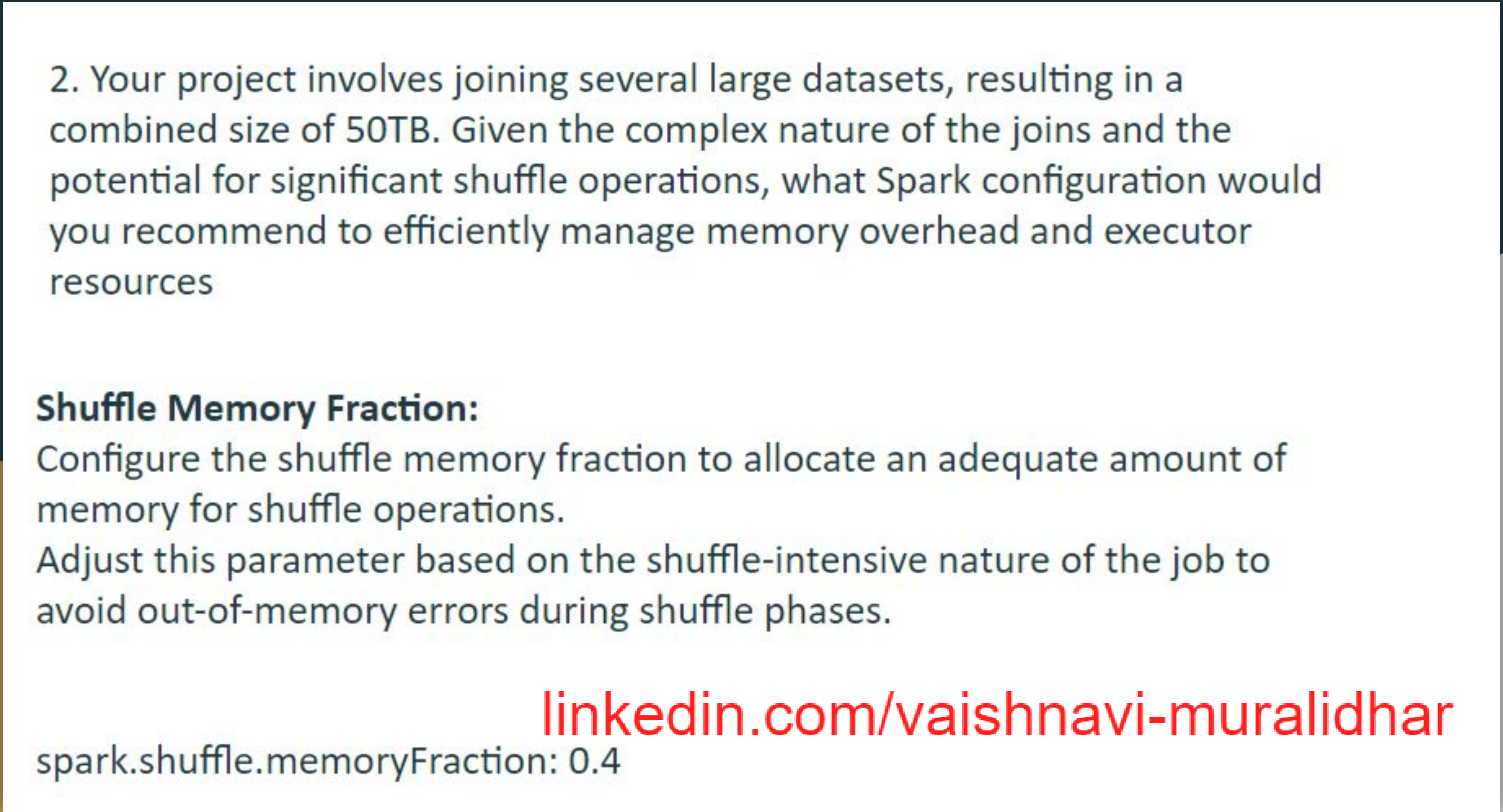
1. Job runtime can vary anywhere between 1 min to 50 mins

Try to complete the entire execution, 2 hours before SLA

Ex. 20 pipelines taking 380 mins

Staring at 2AM IST. SLA is 8AM IST
Jobs have to complete by 6AM IST
Execution time available: 240 mins

Find the outlier pipelines and optimise them. Run time plays a crucial role.



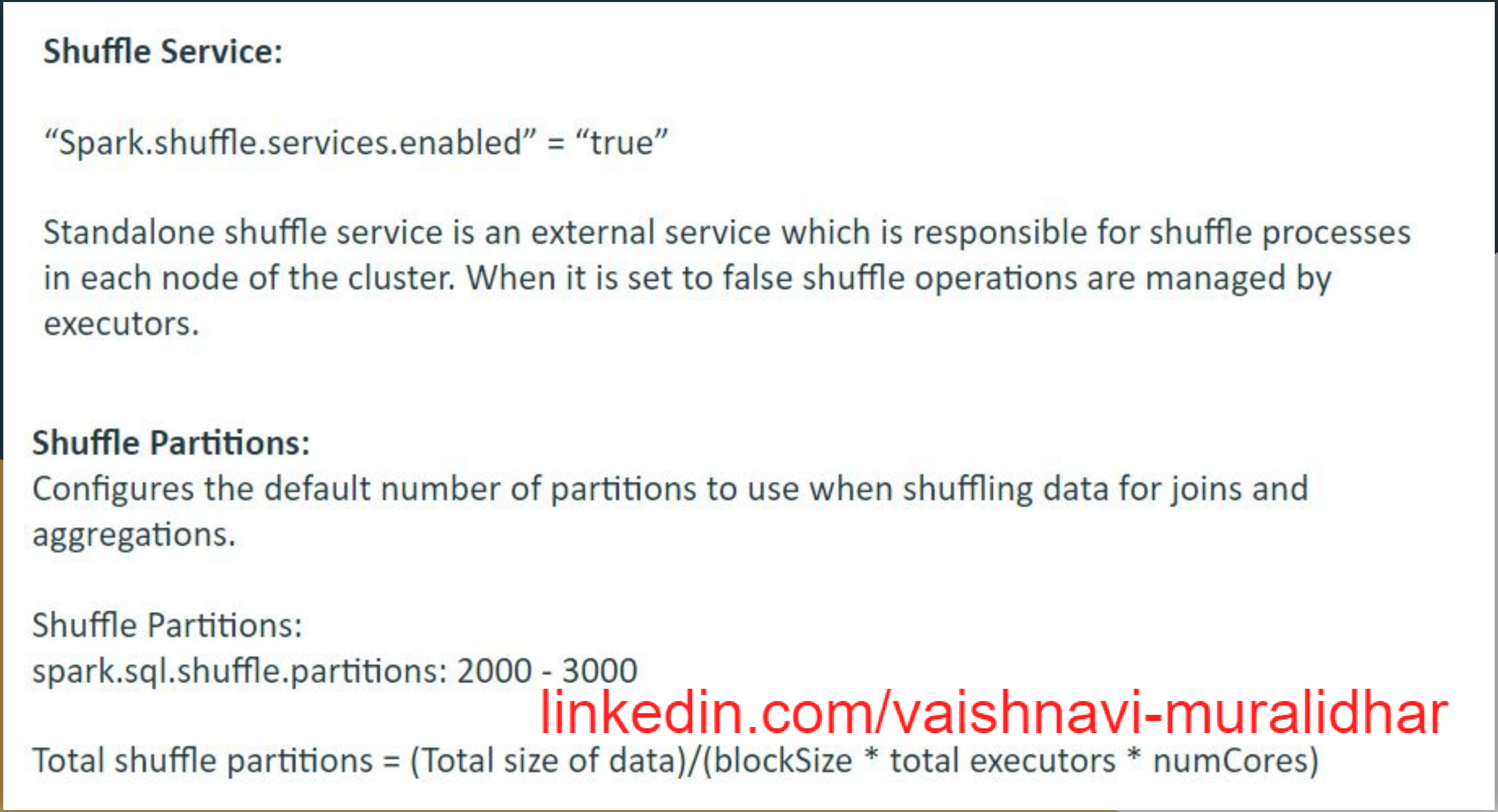
2. Your project involves joining several large datasets, resulting in a combined size of 50TB. Given the complex nature of the joins and the potential for significant shuffle operations, what Spark configuration would you recommend to efficiently manage memory overhead and executor resources

Shuffle Memory Fraction:

Configure the shuffle memory fraction to allocate an adequate amount of memory for shuffle operations.

Adjust this parameter based on the shuffle-intensive nature of the job to avoid out-of-memory errors during shuffle phases.

```
spark.shuffle.memoryFraction: 0.4
```



Shuffle Service:

“Spark.shuffle.services.enabled” = “true”

Standalone shuffle service is an external service which is responsible for shuffle processes in each node of the cluster. When it is set to false shuffle operations are managed by executors.

Shuffle Partitions:

Configures the default number of partitions to use when shuffling data for joins and aggregations.

Shuffle Partitions:
spark.sql.shuffle.partitions: 2000 - 3000

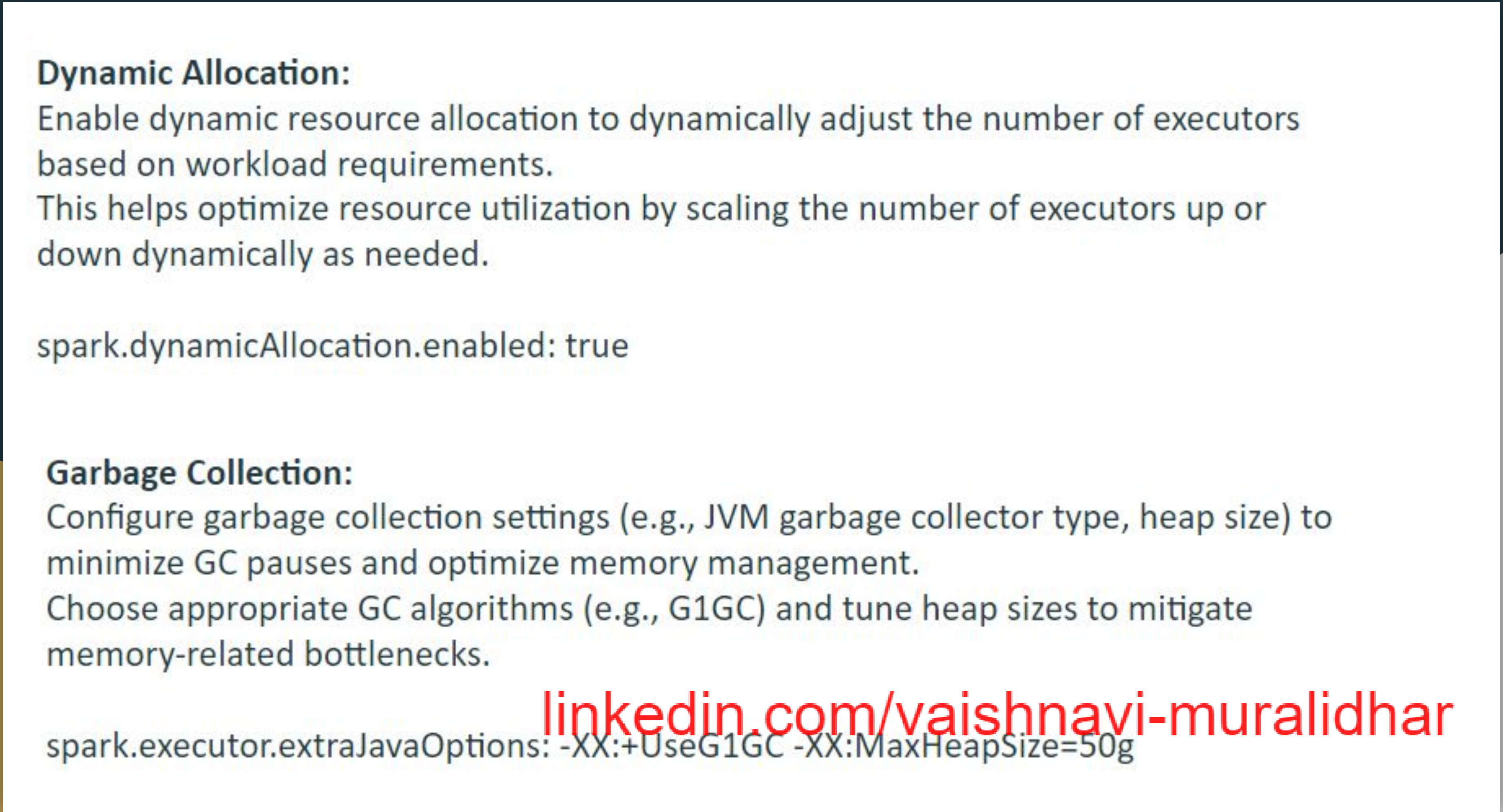
Total shuffle partitions = (Total size of data)/(blockSize * total executors * numCores)

Memory allocation:

Set the executor and driver memory size appropriately to ensure sufficient memory for data processing and shuffle operations.
Allocate an optimal number of executor cores considering the CPU resources required for parallel processing.

Set the memory overhead per executor to accommodate JVM overheads, off-heap storage, and other system-related memory requirements.
(calculate these parameters below)

spark.executor.memory:
spark.executor.cores:
spark.executor.memoryOverhead:
Spark.driver.memory:
spark.driver.memoryOverhead:



Dynamic Allocation:

Enable dynamic resource allocation to dynamically adjust the number of executors based on workload requirements.

This helps optimize resource utilization by scaling the number of executors up or down dynamically as needed.

```
spark.dynamicAllocation.enabled: true
```

Garbage Collection:

Configure garbage collection settings (e.g., JVM garbage collector type, heap size) to minimize GC pauses and optimize memory management.

Choose appropriate GC algorithms (e.g., G1GC) and tune heap sizes to mitigate memory-related bottlenecks.

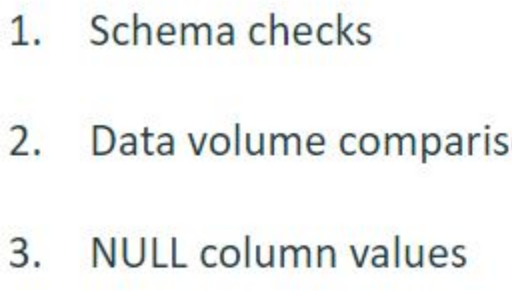
```
spark.executor.extraJavaOptions: -XX:+UseG1GC -XX:MaxHeapSize=50g
```

Broadcast Join Threshold:
Tune the broadcast join threshold to determine the size threshold for broadcasting smaller tables during join operations.
Use broadcast joins for smaller tables to reduce shuffle data and minimize network traffic.

```
spark.sql.autoBroadcastJoinThreshold: 10MB
```

linkedin.com/vaishnavi-muralidhar

What are the checks done after data ingestion:



1. Schema checks
2. Data volume comparison
3. NULL column values

How business requirements are stated ?

1. Complete new product/application onboarding
2. New feature enhancement
3. New source onboarding
4. New report to be generated

How business requirements are stated ?

1. EPIC (Initiative)
2. Description, examples(tables, sample data), columns to include, documentation
3. SLA to be met
4. Business value
5. Timeline for UAT validation, production deployment

linkedin.com/vaishnavi-muralidhar