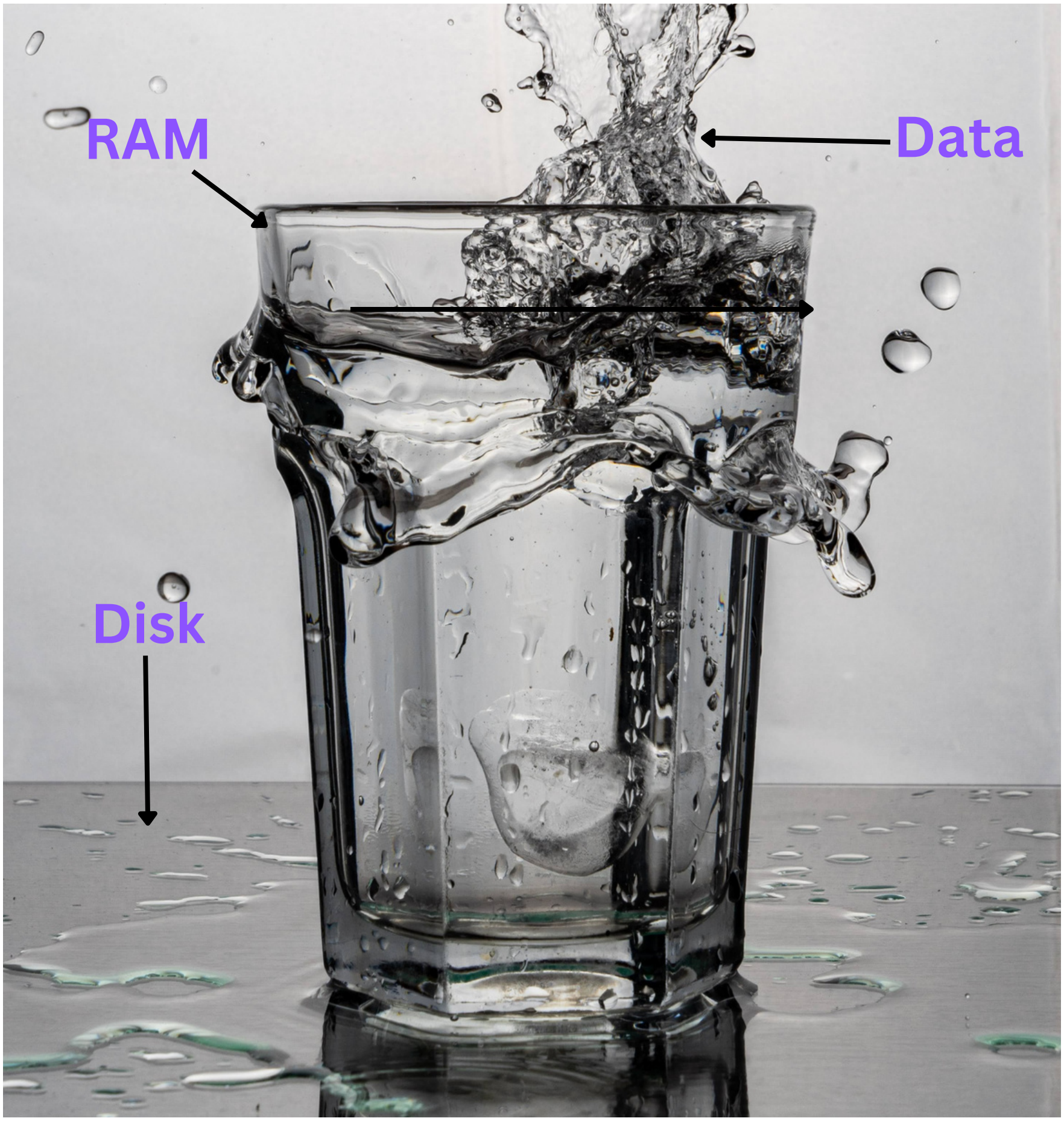
Data Spill in Apache Spark

The glass is neither half full nor half empty



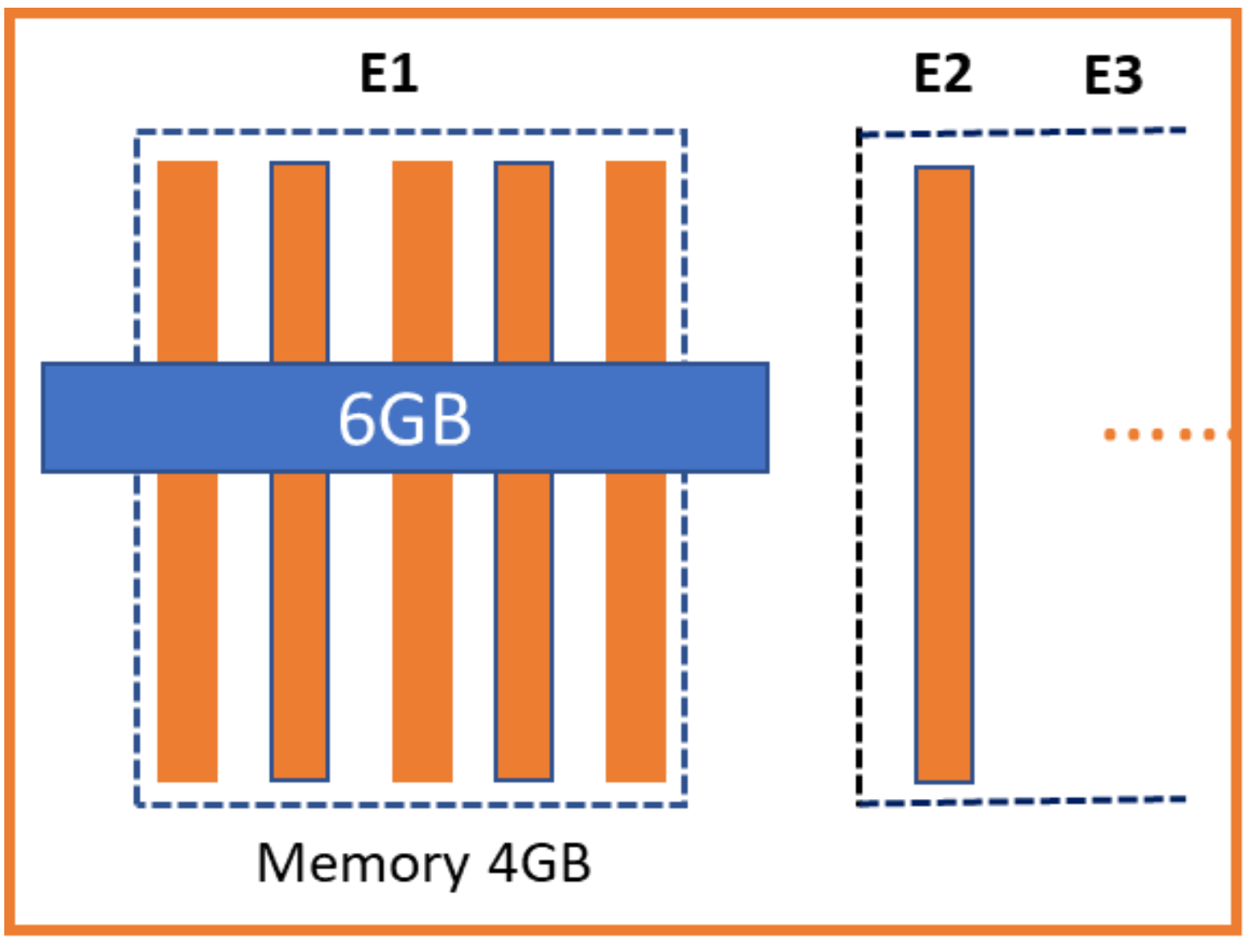
@shuvajithazra

Data Spill in Apache Spark

- Spill is the term used to refer to the act of moving a Partition from RAM to disk, and later back into RAM again

- This occurs when a given partition is simply too large to fit into RAM in order to avoid OOM errors

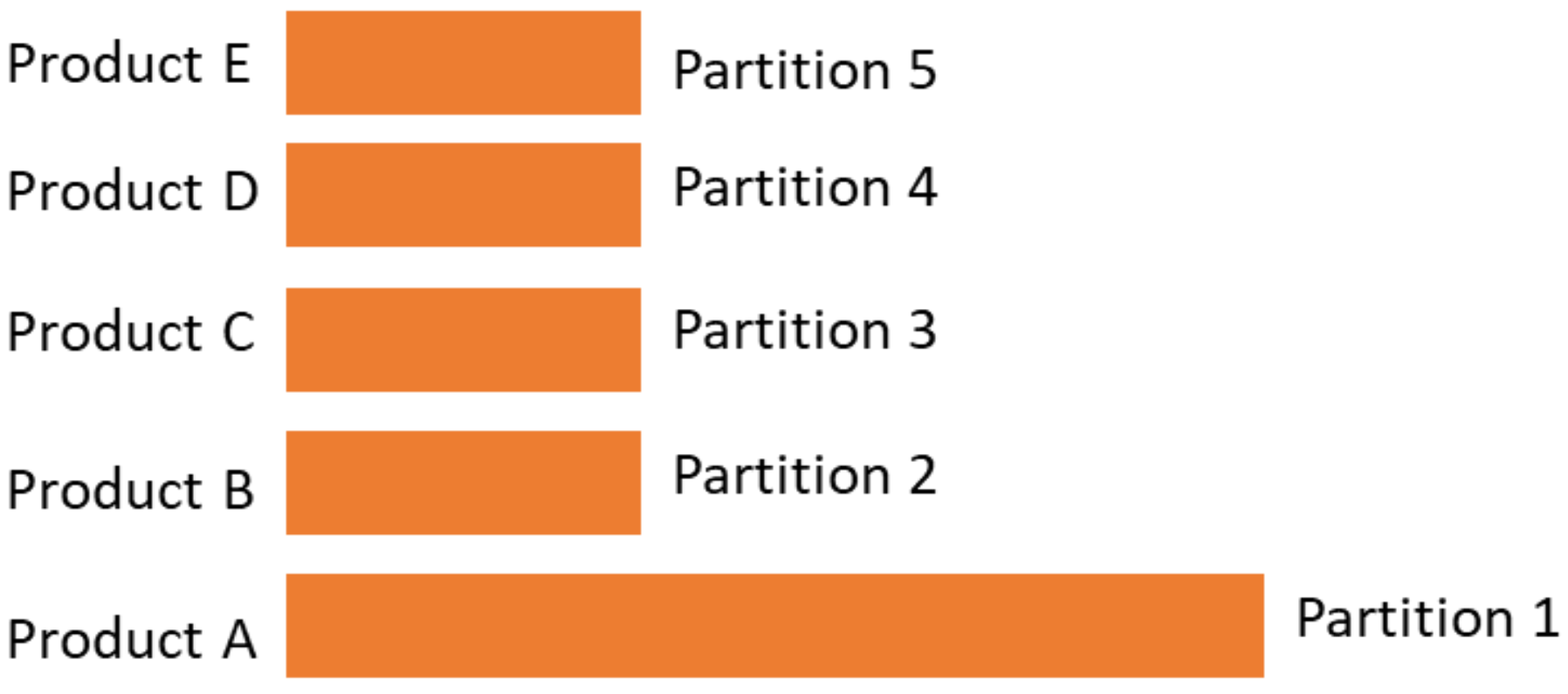
- In this case, Spark is forced into [potentially] expensive disk reads and writes to free up local RAM



@shuvajithazra

Mitigation Plan 1: Avoid Data Skew

Sometimes due to data skew one partition size becomes way higher than others resulting in data spill in larger partition. In such scenario we need to adress the data skew



Product Sales dataframe after partitioned on Product

Steps to avoid Data Skew:

1) Salting

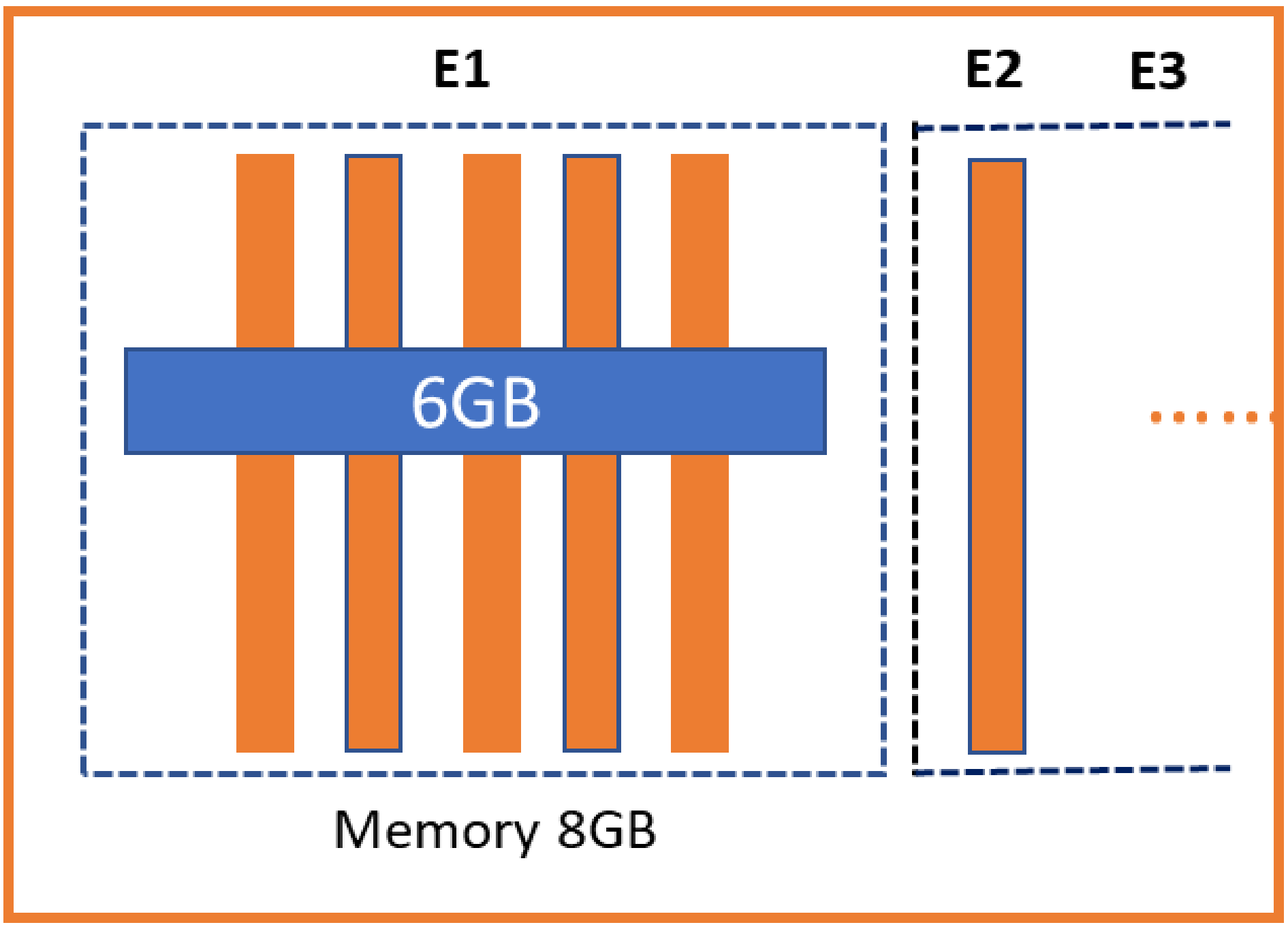
2) Adaptive Query Language in Spark 3

3) Databricks specific Skew Hint

@shuvajithazra

Mitigation Plan 2: Upgrade Cluster Configuration

It is quick but definitely expensive. You can allocate a cluster with more memory per worker

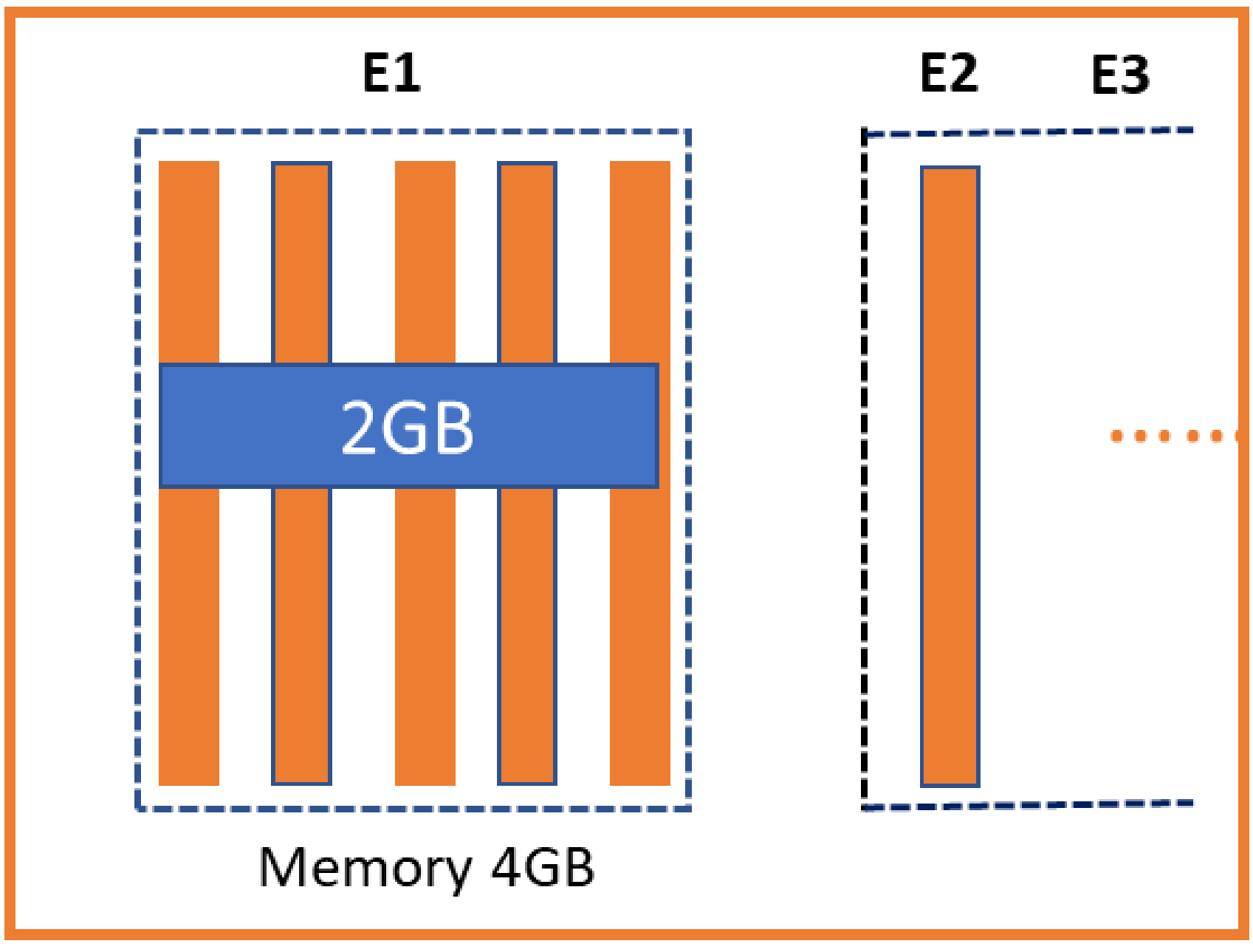


@shuvajithazra

Mitigation Plan 3: Reduce Partition Size

Decrease the size of each partition by increasing the number of partitions.

- By managing spark.sql.shuffle.partition
- By explicitly repartitioning
- By managing spark.sql.files.maxPartitionBytes



@shuvajithazra

spark.sql.shuffle.partitions

- The spark.sql.shuffle.partitions parameter allows you to control the number of partitions used during shuffling.

- By default, Spark sets this parameter to 200. However, you can adjust this value to optimize performance based on the characteristics of your data and the available resources.

main.py module1.py module2.py module3.py partitio

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, concat, lit, rand
from pyspark.sql.types import StructType,StructField, StringType

spark = SparkSession.builder\
    .appName("ReducePartitionSizeExample") \
    .config("spark.sql.shuffle.partitions", "10")\
    .getOrCreate()

```

@shuvajithazra

Repartitioning

With Repartitioning we can increase the number of partitions which in turn reduce the data size of each partition.

However, it's important to note that repartitioning involves shuffling data, which can incur a performance cost.

```
inputRDD = spark.sparkContext.textFile("path_to_input_file.txt")
repartitionedRDD = inputRDD.repartition(20)
```

spark.sql.files.maxPartitionBytes

The spark.sql.files.maxPartitionBytes parameter allows you to set the maximum size of each partition in bytes. By default, Spark sets this parameter to 128MB.

```
spark = SparkSession.builder\
    .appName("ReducePartitionSizeExample") \
    .config("spark.sql.files.maxPartitionBytes", "134217728")\
    .getOrCreate()
```

@shuvajithazra