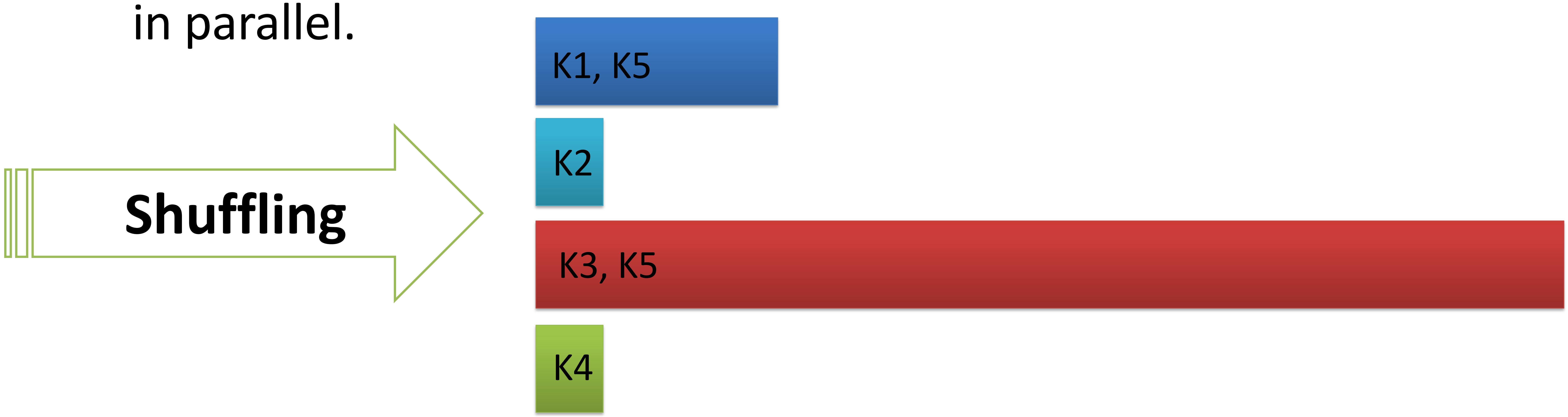
Skewness (uneven data distribution across the partitions)

- Data Skew is a very common problem with big data **after shuffling** and managing skew is very important for running the pipeline seamlessly.

  - Key distribution is not uniform (**highly skewed**), causing some partitions to be very large and not allowing Spark to process data in parallel.

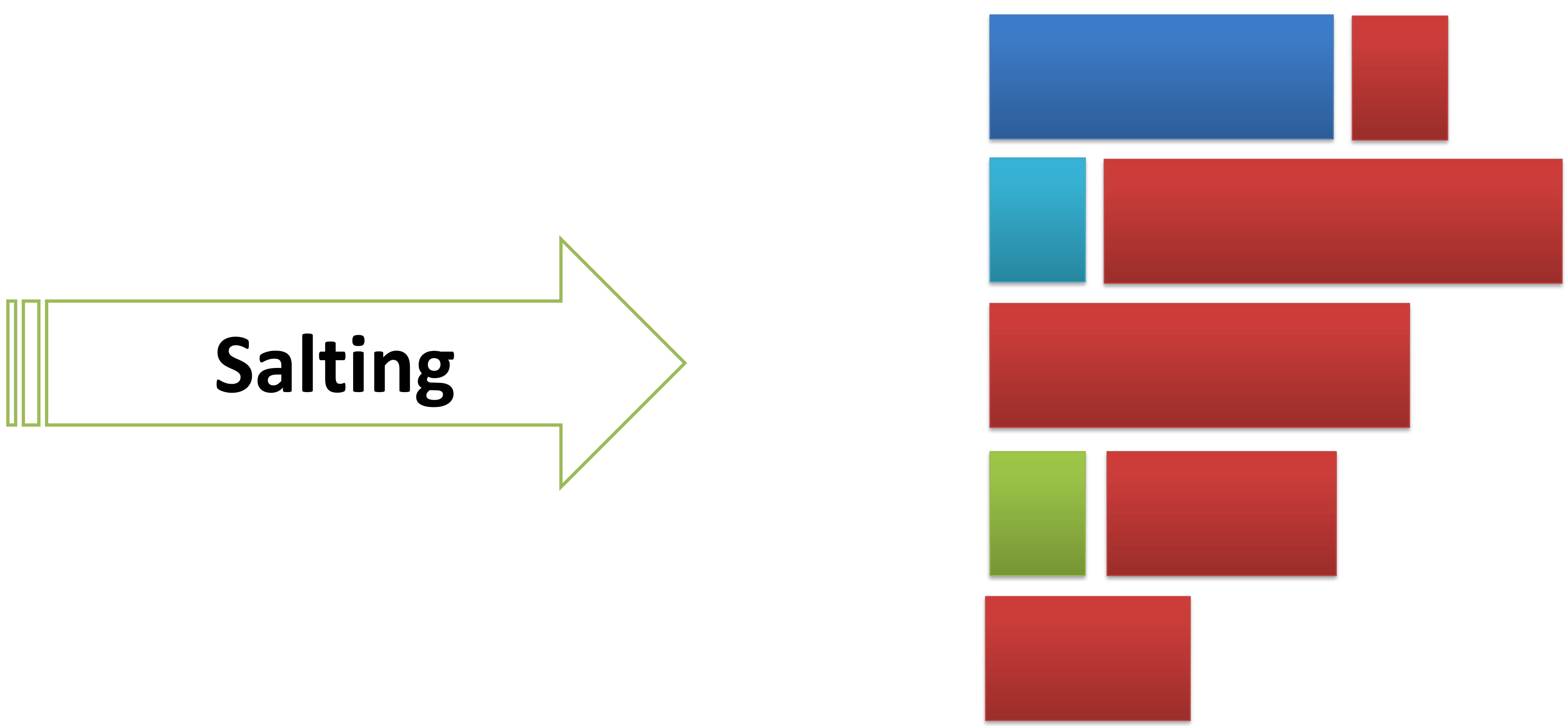


Distributed Processing Challenges: Handling Data Skew in RDD PySpark
Udemy Course: Best Hands-on Big Data Practices and Use Cases using PySpark

Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

How to mitigate skewed data?

- **SALTING** is a technique that adds random values to the join keys, then Spark can partition data evenly.



Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Spill

- If there is a large partition (such as skewed data) that cannot fit into RAM, we need to incorporate DISK read/write to avoid application/system crash.

- **Spill** refers to the moving an RDD from RAM to DISK, and later back it to the RAM again for processing.

- Spark will execute this expensive disk read/write to free up RAM and it avoids **Out of Memory** error.

Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Spill

- **Spill (memory)** is the size of the deserialized form of the shuffled data in memory. The size of the data that exists in memory before it is spilled.

- **Spill (disk)** is the size of the serialized form of the data on disk. This is the size of the data that gets spilled and written into the disk.

Source: https://spark.apache.org/docs/latest/web-ui.html

Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Common scenarios leading to SPILL

- `Aggregation/Shuffling` on **Skewed Data**
- `Join` or `Crossjoin` (Cartesian product) operations
- `Explode` operation (convert the array of arrays to a new column)
- `maxPartitionBytes` (default: 128 MB). This is the maximum number of bytes to pack into a single partition when reading files.

Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Mitigation of SPILL

- **Solve the Data Skew first**
- Use `repartitioning()` with the known number of partitions
- Increase the Memory of workers

```
spark.conf.set("spark.executor.memory",75g)
spark.conf.set("spark.driver.memory",100g)
```


Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Mitigation of SPILL (continue)

- [DataFrame] Manage `spark.sql.shuffle.partitions`: this configures the number of partitions that are used during data shuffling for joins/aggregations in DF (default 200).

```
spark.conf.set("spark.sql.shuffle.partitions",8)
```


Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Mitigation of SPILL (continue)

- [RDD] Manage `spark.default.parallelism`: The default value for this configuration set to the number of all cores on all nodes in a cluster, that are used during data shuffling for joins/aggregations.

```
spark.conf.set("spark.default.parallelism",4)
```


Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Mitigation of SPILL (continue)

- Manage `spark.sql.files.maxPartitionBytes`: this is the maximum number of bytes to pack into a single partition when reading files (default 128MB = 134,217,728 bytes).

  - example: 256MB = 256 * 1024 * 1024 = 268,435,456 bytes

```
spark.conf.set("spark.sql.files.maxPartitionBytes",maxSplit)
```


Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com

Wrapping up

- The performance of your Big Data application depends on

1. Dataset size
2. Number of cores and Memories size for in-parallel computations (hardware)
3. Spark shuffling and performance (manage Skew & Spill)

Author: Amin Karami (PhD, FHEA) amin.karami@ymail.com