DIGGIBYTE

CREATING VALUE WITH DATA

DIGGIBYTE
CREATING VALUE WITH DATA

Spark Optimization

Presented by,
Amit Kundu

DIGGIBYTE
CREATING VALUE WITH DATA

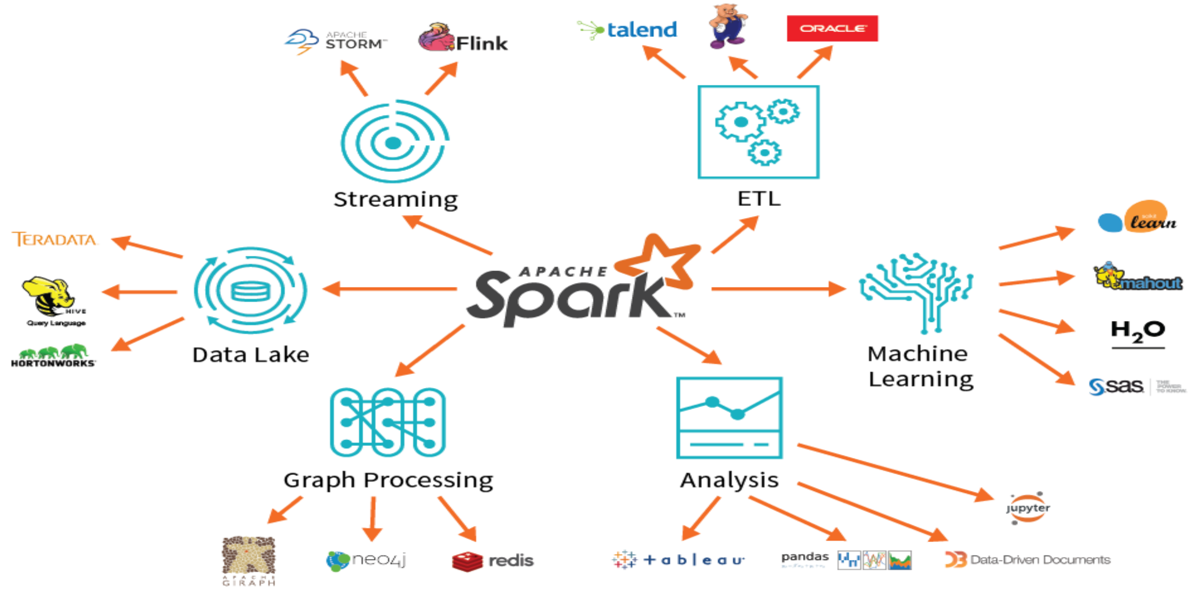
Agenda

- What is Spark?
- Impact of inefficient performance on Spark jobs
- Partitioning
- Bucketing
- Z-ordering

DIGGIBYTE
CREATING VALUE WITH DATA

What is Spark?

- Spark is a powerful distributed computing framework that processes large-scale data.
- Apache Spark has established itself as a robust solution for processing vast amounts of data, providing scalability and performance benefits. Within the domain of PySpark, effective data handling becomes essential, with three primary strategies standing out: partitioning, bucketing, and z-ordering.



DIGGIBYTE
CREATING VALUE WITH DATA

Why to Optimize Performance?

- Performance Boost: Optimizations enhance the speed and efficiency of data processing, ensuring faster job execution.
- Resource Efficiency: They maximize the utilization of CPU, memory, and network resources, preventing wastage in distributed computing.
- Cost Savings: By reducing computational overhead, optimizations lead to lower cloud computing costs.
- Scalability: Spark can handle larger datasets and more complex tasks efficiently with optimizations.
- Data Integrity: Optimizations help maintain data quality and consistency by addressing issues like skew and redundancy.
- These are the effective ways to optimize spark performance:
    - Partitioning
    - Bucketing
    - Z-ordering

DIGGIBYTE
CREATING VALUE WITH DATA

Partitioning

- Partitioning is a technique that involves dividing large datasets into smaller, more manageable parts. In Spark, partitions are the basic units of parallelism, and organizing data into partitions can significantly improve performance. By distributing data across multiple partitions, Spark can execute operations in parallel, leveraging the full power of a cluster.

DIGGIBYTE
CREATING VALUE WITH DATA

Why we need Partitioning?

•Load Balancing: Guaranteeing fair distribution of data across partitions optimizes cluster utilization, addressing data skew and averting uneven workload distribution among nodes.
•Network Optimization: Implementing a well-crafted partitioning strategy minimizes network data transfer during operations such as join or groupByKey, accelerating execution speed by minimizing unnecessary data shuffling.
•Parallelism: Fine-tuning partitions allows for parallel and autonomous processing on various nodes, maximizing the cluster's computational capabilities.

Types of Partitioning

- Hash Partitioning: Spark employs a hash function to ascertain the partition to which a given record is assigned. Assuming a suitable hash function, this method guarantees a uniform distribution of data across partitions. It is frequently employed in join-related procedures.
- Range Partitioning: Data is divided using a predetermined range of values in a process known as range partitioning. When your data, like dates or numerical values, has a natural ordering, this is helpful.
- Custom Partitioning: This feature lets you design a partitioning plan that is unique to your needs. Partitioning your data according to particular features or business logic may be necessary for this.

Partitioning by Department

Id	Name	Department
1	Smith	Sales
2	Mitchell	Sales
3	Brook	Finance
4	Klassen	Marketing
5	Miller	Marketing
6	Markram	Sales
7	Maxwell	Finance

Id	Name	Department
1	Smith	Sales
2	Mitchell	Sales
6	Markram	Sales

Id	Name	Department
3	Brook	Finance
7	Maxwell	Finance

Id	Name	Department
4	Klassen	Marketing
5	Miller	Marketing

DIGGIBYTE
CREATING VALUE WITH DATA

Bucketing

- Bucketing is a method that organizing data into predetermined-size buckets or files using hash functions. Each bucket functions as a file, and data within a given bucket possess the same hash value. This approach offers greater precision in data organization and can prove especially advantageous for specific query types.

DIGGIBYTE
CREATING VALUE WITH DATA

Why is Bucketing important?

- Enhancing Performance: Utilizing bucketing in Spark tasks like groupBy, join, and orderBy can significantly boost job effectiveness by reducing output sizes and minimizing network shuffling during shuffle operations.
- Mitigate Data Skew: Bucketing plays a crucial role in operations by helping alleviate data skew, optimizing resource allocation for increased processing efficiency.
- Minimize Data Redundancy: Through selective targeting of data subsets, bucketing in Spark eliminates the necessity for exhaustive scans, thereby reducing IO overhead and improving query execution efficiency for enhanced performance.

```
df.write.bucketBy(numBuckets, “column_name”).parquet(“output_path”)
```

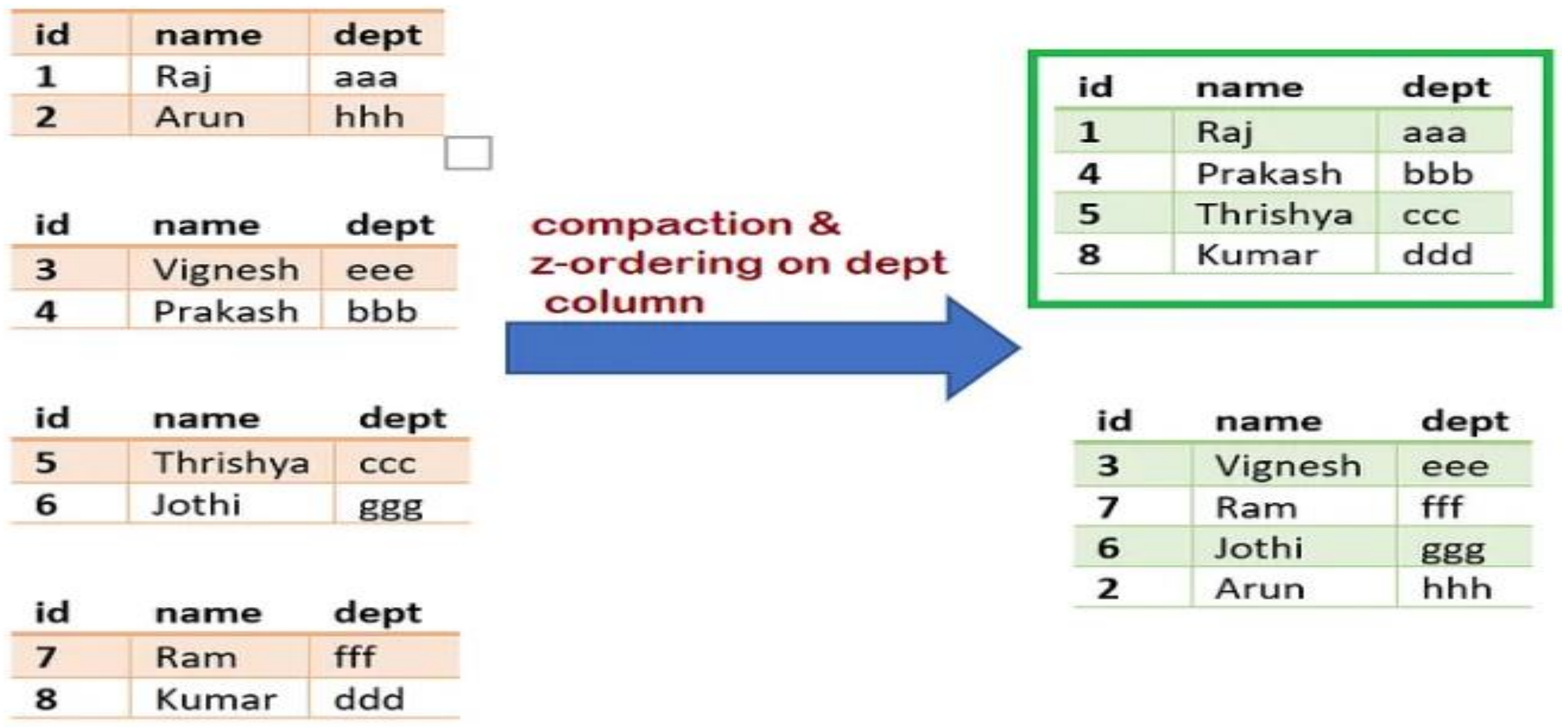
DIGGIBYTE
CREATING VALUE WITH DATA

Z-Ordering

- Z-ordering is a technique to colocate related information in the same set of files.
- In Spark implementation, Z-ordering serves to arrange columns to guarantee closeness of associated data. This deliberate organization proves particularly advantageous during spatial queries, as it amplifies the effectiveness of accessing and handling spatially related data.

```
df.write.option(“zOrderCol”, “column_name”).parquet(“output_path”)
```

Z-Ordering



id	name	dept
1	Raj	aaa
2	Arun	hhh

id	name	dept
3	Vignesh	eee
4	Prakash	bbb

id	name	dept
5	Thrishya	ccc
6	Jothi	ggg

id	name	dept
7	Ram	fff
8	Kumar	ddd

id	name	dept
1	Raj	aaa
4	Prakash	bbb
5	Thrishya	ccc
8	Kumar	ddd

id	name	dept
3	Vignesh	eee
7	Ram	fff
6	Jothi	ggg
2	Arun	hhh

DIGGIBYTE
CREATING VALUE WITH DATA

THANK YOU !!