Spark Optimization

Ricardo Vilaça

HASLab

University of Minho

rmvilaca@di.uminho.pt

Roadmap

- Spark workflow
- Shuffle
- Partitioning/Bucketing
- Caching
- Skew data
- Join Strategies
- Windows
- Adaptative query planning

Spark Workflow

- Every Spark Cluster has a Driver and one or more executors
- Work submitted to the Cluster is split into as many independent Jobs as needed
- Jobs are further subdivided into tasks
- The input to a job is partitioned into one or more partitions
- In between tasks, partitions may need to be re-organized and shared over the network
- As opposed to narrow transformations, wide transformations cause data to shuffle between executors

Pipelining/Stages

- Executing as many operations as possible on a single partition of data
- Combine as many narrow operations as it can into a single Task
- Wide operations force a shuffle, conclude a stage, and end a pipeline

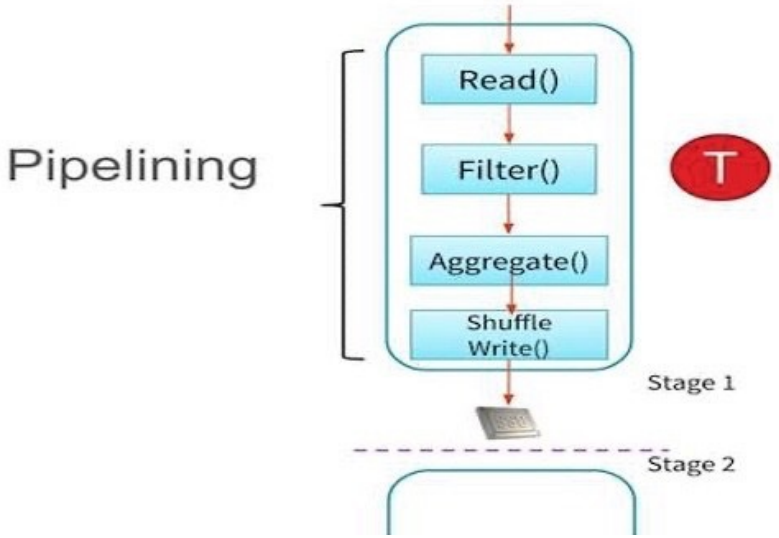
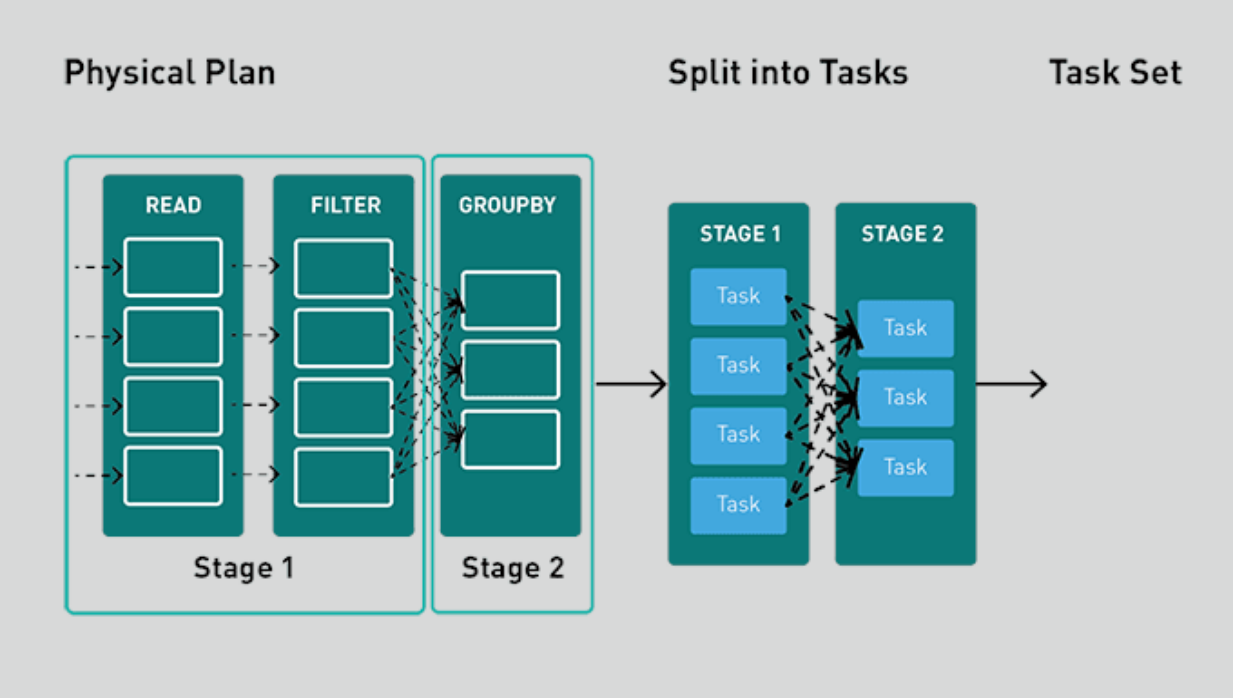


Image from https://www.linkedin.com/pulse/catalyst-tungsten-apache-sparks-speeding-engine-deepak-rajak/

Job Execution

- A set of tasks, namely TaskSet in Spark, is created for each Spark stage
- Each task in the task set computes the same logic/functions on partitions

Image from https://developer.hpe.com/blog/how-spark-runs-your-applications/



Shuffle

- Expensive operator
  - shuffle breaks pipeline
  - materialization points and triggers a new stage within the pipeline
  - needs to move data across the network
  - data is redistributed in a way required by downstream operators

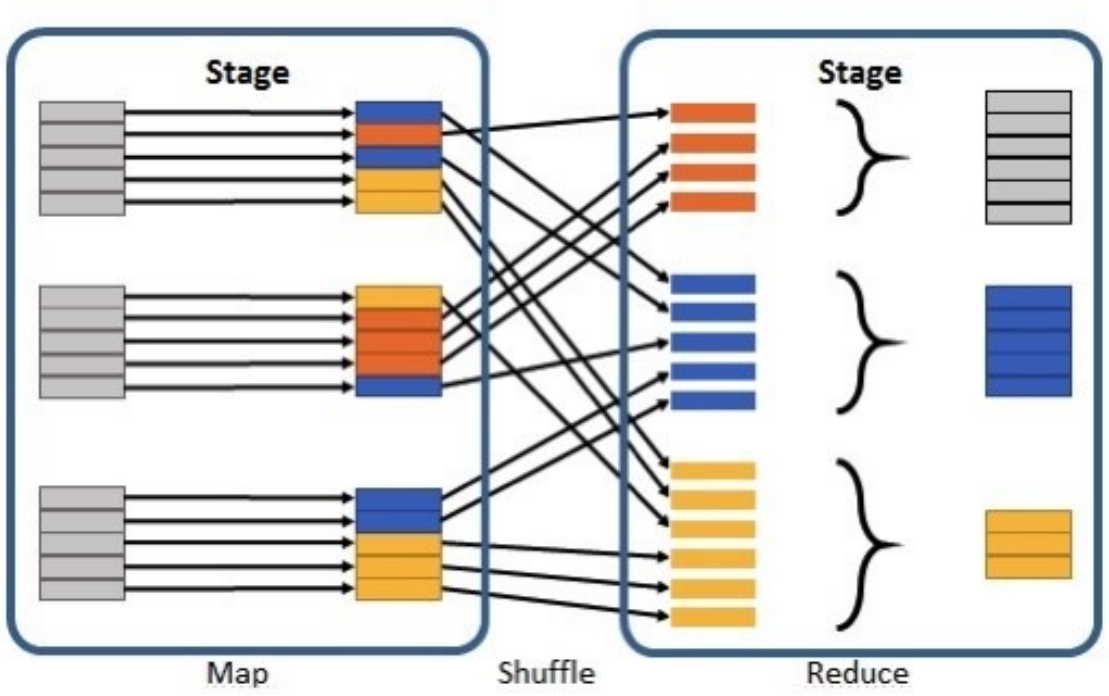


Image from https://www.linkedin.com/pulse/catalyst-tungsten-apache-sparks-speeding-engine-deepak-rajak/

Shuffle

- Best number of partitions is data dependent
  - may differ vastly from stage to stage, query to query
  - too few partitions
    - data size of each partition may be very large
    - may need to spill data to disk
  - too many partitions
    - data size of each partition may be very small
    - lot of small network data fetches to read the shuffle blocks
    - inefficient I/O pattern
    - Increase scheduler burden with large number of tasks

spark.sql.shuffle.partitions
spark.memory.offHeap.enable
spark.memory.ofHeap.size

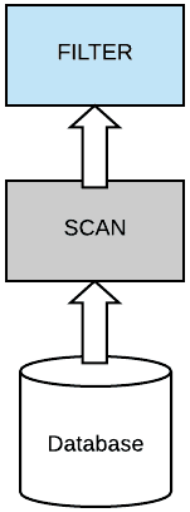
Partitioning

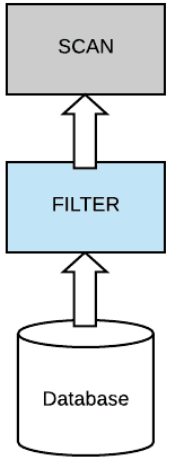
- Subsets of dataset in parallel on different computers
- Different distribution patterns
  - AllTuples - single partition
  - BroadcastDistribution – entire dataset is broadcasted to every node
  - ClusteredDistribution – rows sharing the same values for the clustering expression are co-located in the same partition
  - HashClusteredDistribution – rows are clustered according to the hash of the given expressions
  - OrderedDistribution – rows are ordered across partitions and not necessarily within a partition

```
COALESCE
REPARTITION REPARTITION_BY_RANGE
spark.sql.files.maxPartitionBytes
```

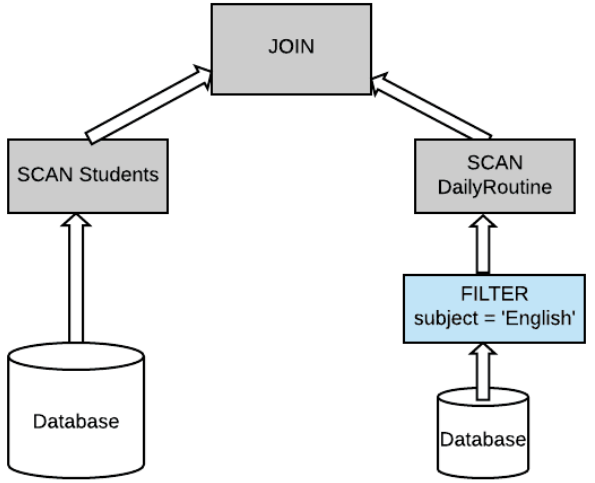
Partition Pruning

- Optimizer will avoid reading files that cannot contain the needed data





Filter Push Down



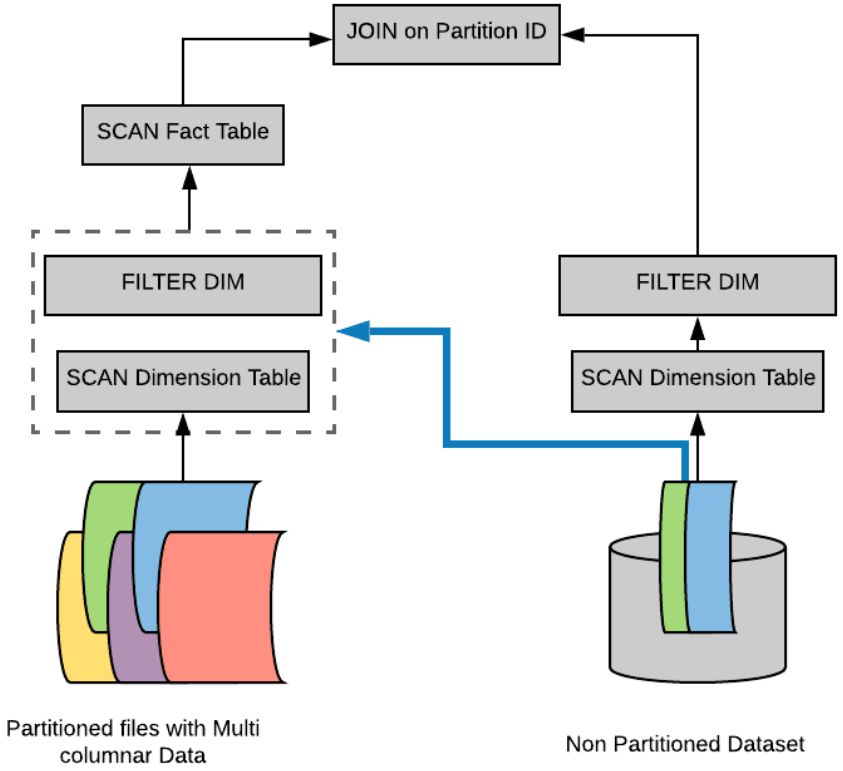
Static Pruning not possible

Images from https://dzone.com/articles/dynamic-partition-pruning-in-spark-30

Dynamic Partition Pruning (DPP)

- Predicate push down optimisation method
- Aims to minimise I/O costs of the data read from the data sources
- Optimization of JOIN batch queries of partitioned tables using partition columns in a join condition
- Best results are expected in JOIN queries between a large fact table and a much smaller dimension table (*star-schema queries*)
  - push filter conditions down to the large fact table and reduce the number of rows to scan

spark.sql.optimizer.dynamicPartitionPruning.enabled



Images from https://dzone.com/articles/dynamic-partition-pruning-in-spark-30

Sorting

- Ordering a dataset based on a sequence of giving ordering expressions
- Global sort of all partitions
- Supports spilling

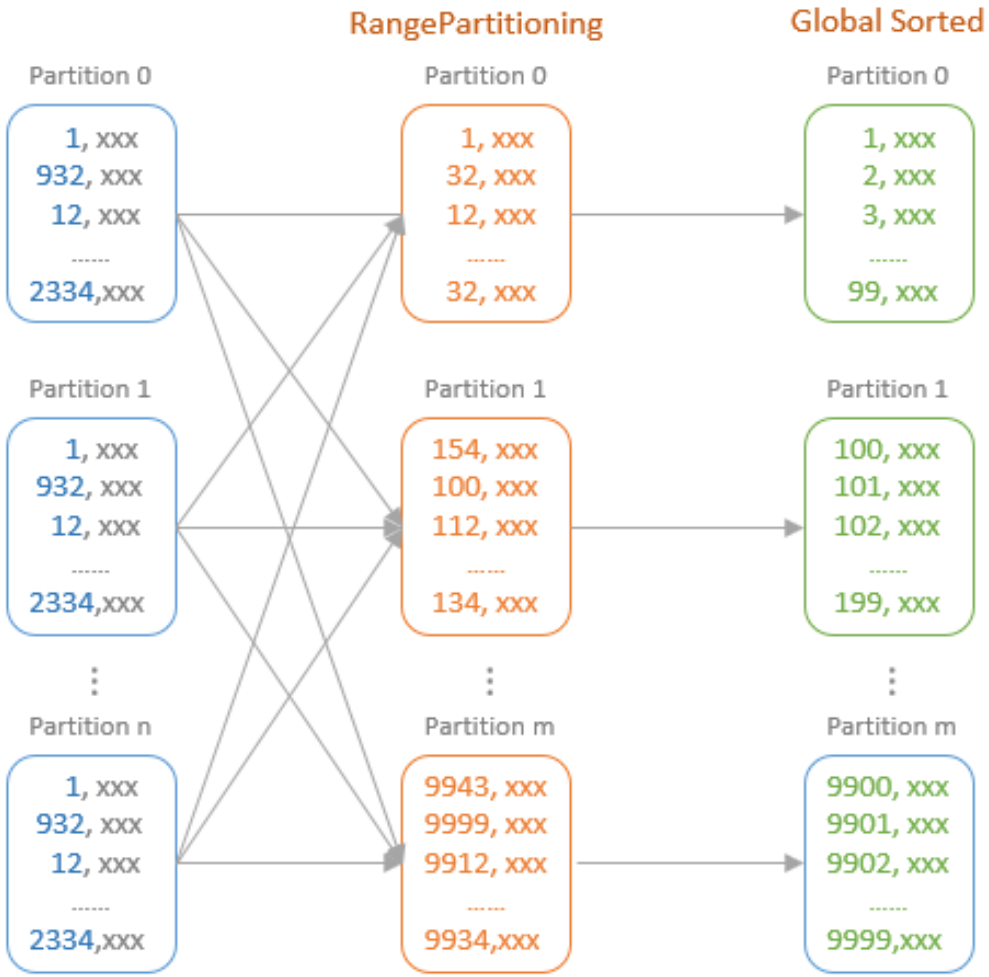


Image from: https://dataninjago.com/2022/01/23/spark-sql-query-engine-deep-dive-15-unsafeexternalsorter-sortexec/

Bucketing

- Groups data with the same bucket value
- Distributes data across a fixed number of buckets by a hash on the bucket value
- Avoid data shuffle in join queries
- Also performs well when the number of unique values is large
- Requires sorting on reading time
  - degrades the performance

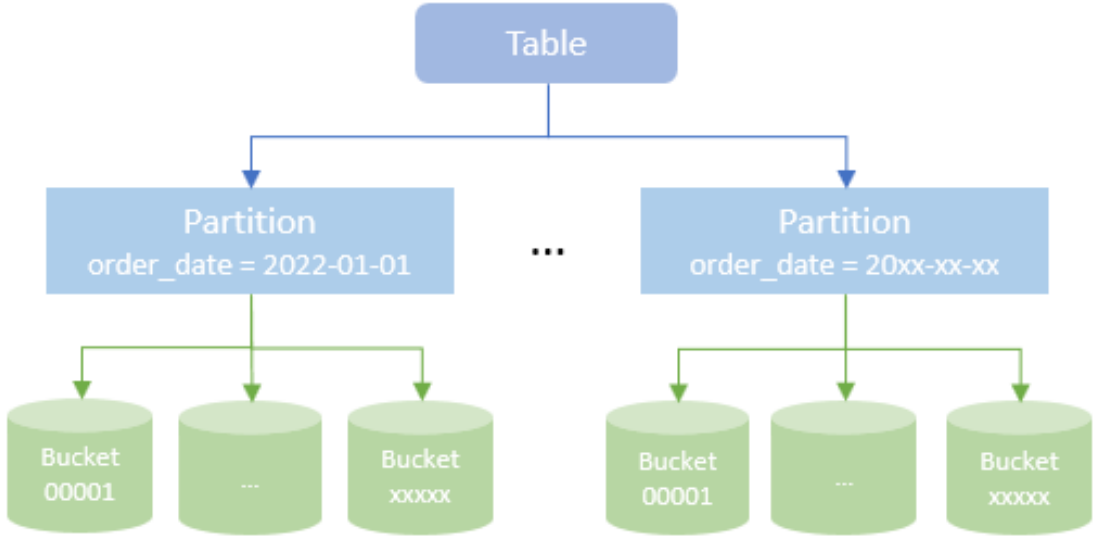


Image from: https://dataninjago.com/2022/02/07/spark-sql-query-engine-deep-dive-18-partitioning-bucketing/

`spark.sessionState.conf.bucketingEnabled`
DataFrameWriter.bucketBy

Caching

- The reuse of shuffle files

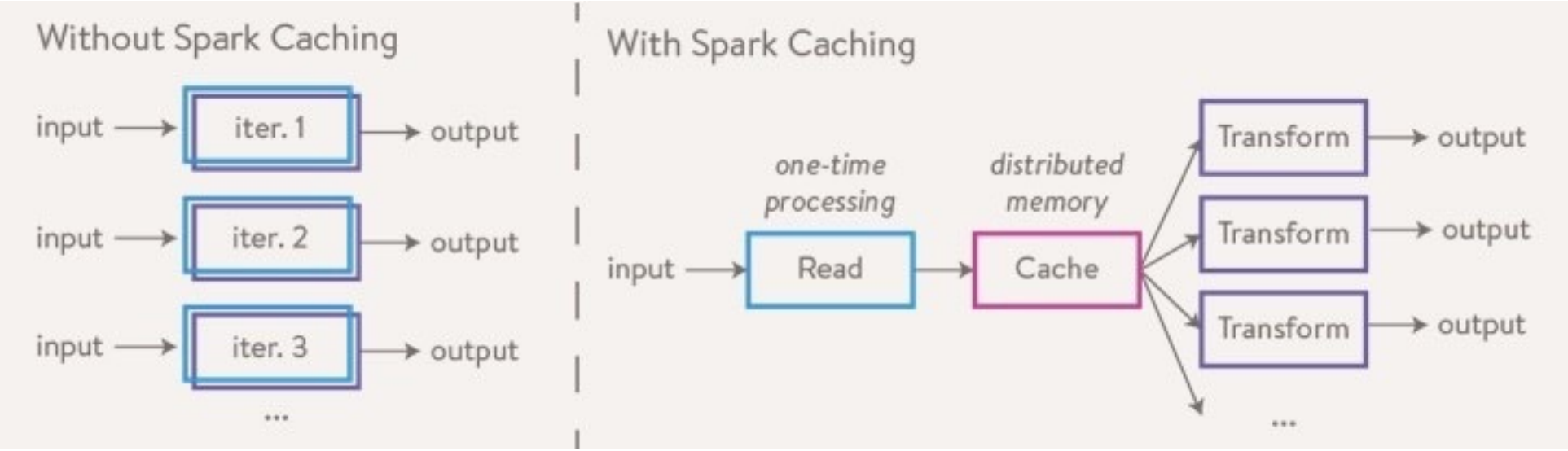


Image from https://www.linkedin.com/pulse/catalyst-tungsten-apache-sparks-speeding-engine-deepak-rajak/

Caching

- Caching can be done in memory or disk:
  - .cache() or .persist(MEMORY_ONLY)
  - .persist(MEMORY_ONLY_SER)
  - .persist(DISK_ONLY)
  - .persist(MEMORY_AND_DISK)
  - …
- and other combinations including memory, disk, replication, serialization, and off-heap

```
dataFrame.cache()
dataFrame.unpersist()
CACHE TABLE …
```

Caching – Block Manager

- Each RDD consists of several blocks and each block is cached independently
- LRU for block eviction

`spark.memory.storageFraction`

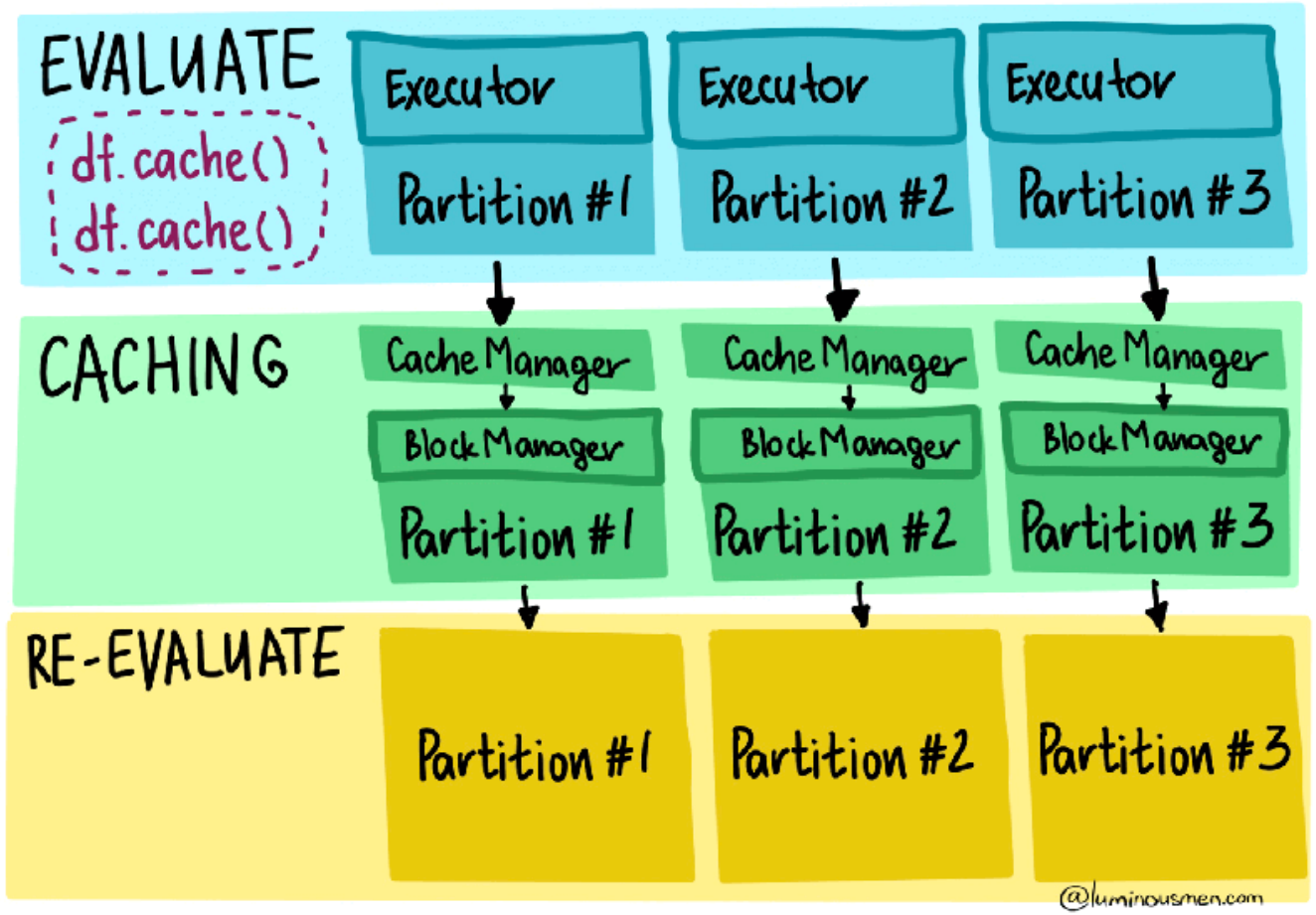


Image from https://luminousmen.com/post/explaining-the-mechanics-of-spark-caching

Caching Hints

- Tradeoffs
  - serialization, slower but smaller
- Execution memory and storage memory share a unified region
  - unnecessary caching may increase spill onto the disk
    - performance hit
- When reading the data from the cache, Spark will read the entire dataset

Skew data

- Working duration of the entire stage is directly dependent on the longest running time of the task
  - May cause a spill of the data from memory to disk
- Solutions
  - Salting
  - Repartition
  - Adaptive query execution (AQE)

Salting

- Adding randomization to the data to help it to be distributed more uniformly
- Is applied only to the skewed key

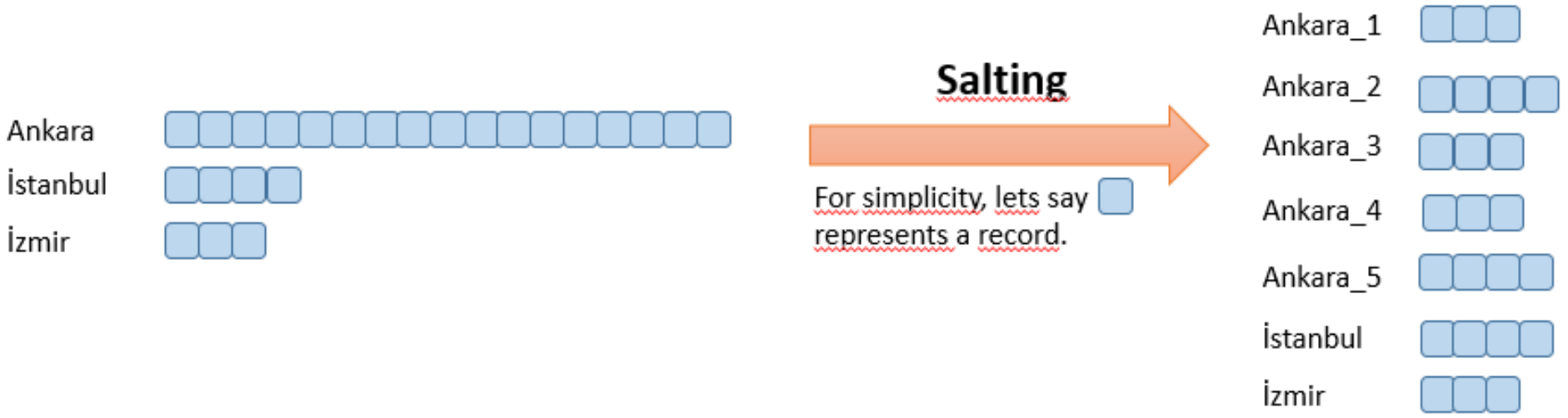


Image from https://towardsdatascience.com/apache-spark-performance-boosting-e072a3ec1179

Repartition

- Full shuffle, creates new partitions, and increases the level of parallelism in the application
  - extra cost
- Might be performed by specific columns
  - if there exists multiple joins or aggregations on these columns
- Coalesce
  - reduce the partition number without shuffling
  - may not solve the imbalance problem in the distribution of data

Join Strategies

- Joins are one of the fundamental operation and implies shuffle
- Performance depends upon the strategy used to tackle each scenario
- Five built-in Join physical operators:
  - **BroadcastHashJoinExec**
  - ShuffledHashJoinExec
  - SortMergeJoinExec
  - **CartesianProductExec**
  - BroadcastNestedLoopJoinExec
- Join strategy selection takes into account:
  - join type is equi-join or not
  - join strategy hint
  - size of Join relations

SQL Hints BROADCAST, MERGE, SHUFFLE_HASH, SHUFFLE_REPLICATE_NL
spark.sql.autoBroadcastJoinThreshold
spark.sql.join.preferSortMergeJoin

Broadcast Hash Join

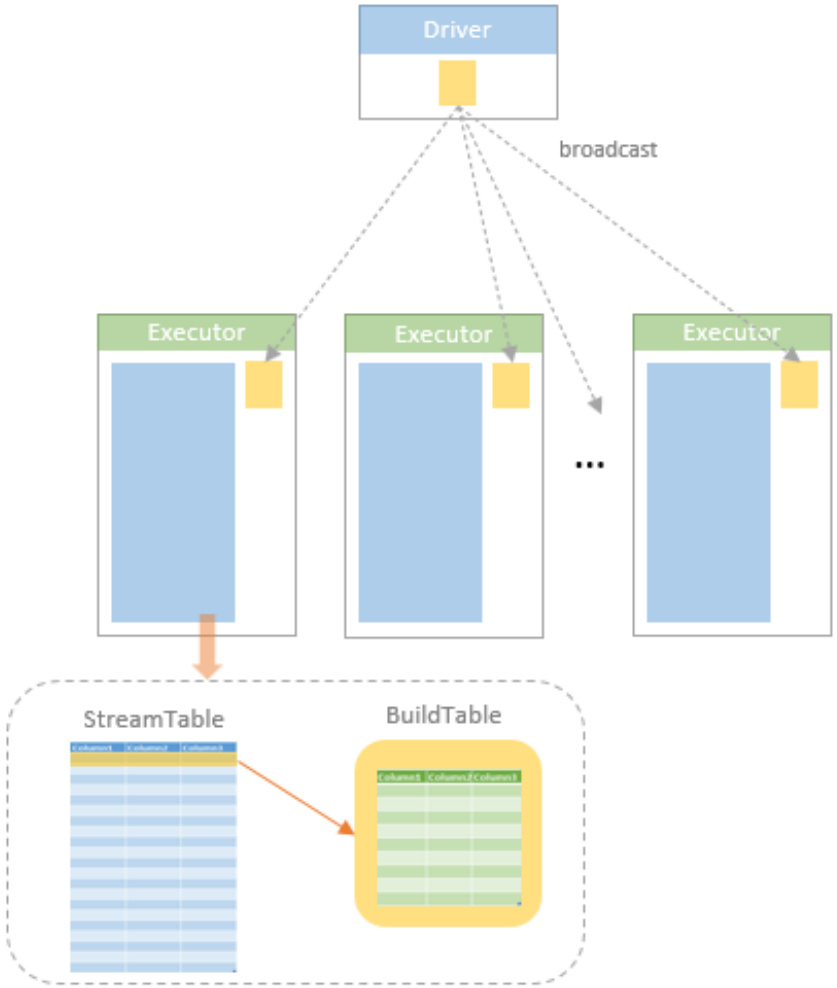


Image from:
https://dataninjago.com/2022/01/11/spark-sql-query-engine-deep-dive-11-join-strategies/

Shuffle Hash Join

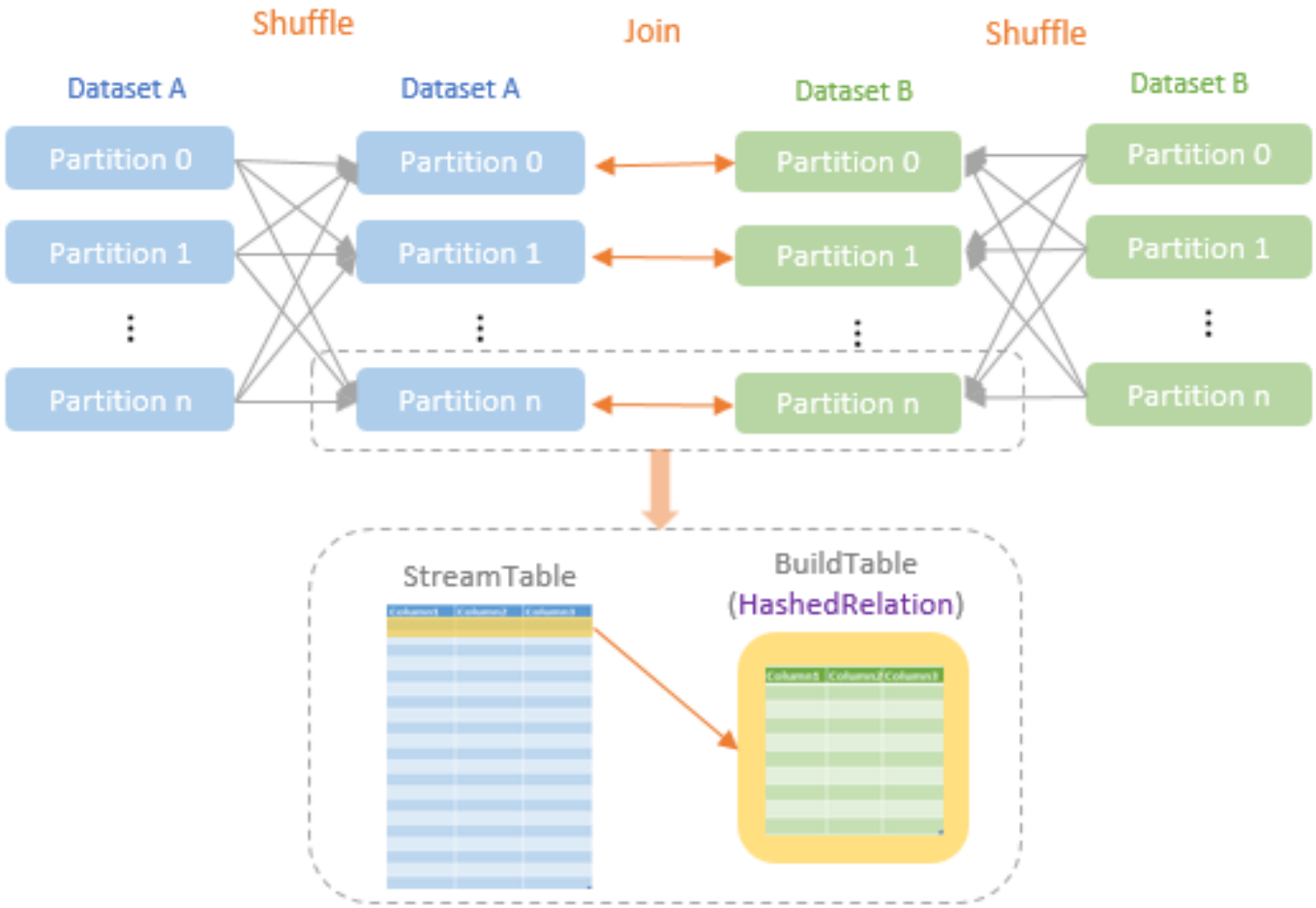


Image from: https://dataninjago.com/2022/01/11/spark-sql-query-engine-deep-dive-11-join-strategies/

Shuffle Sort Merge Join

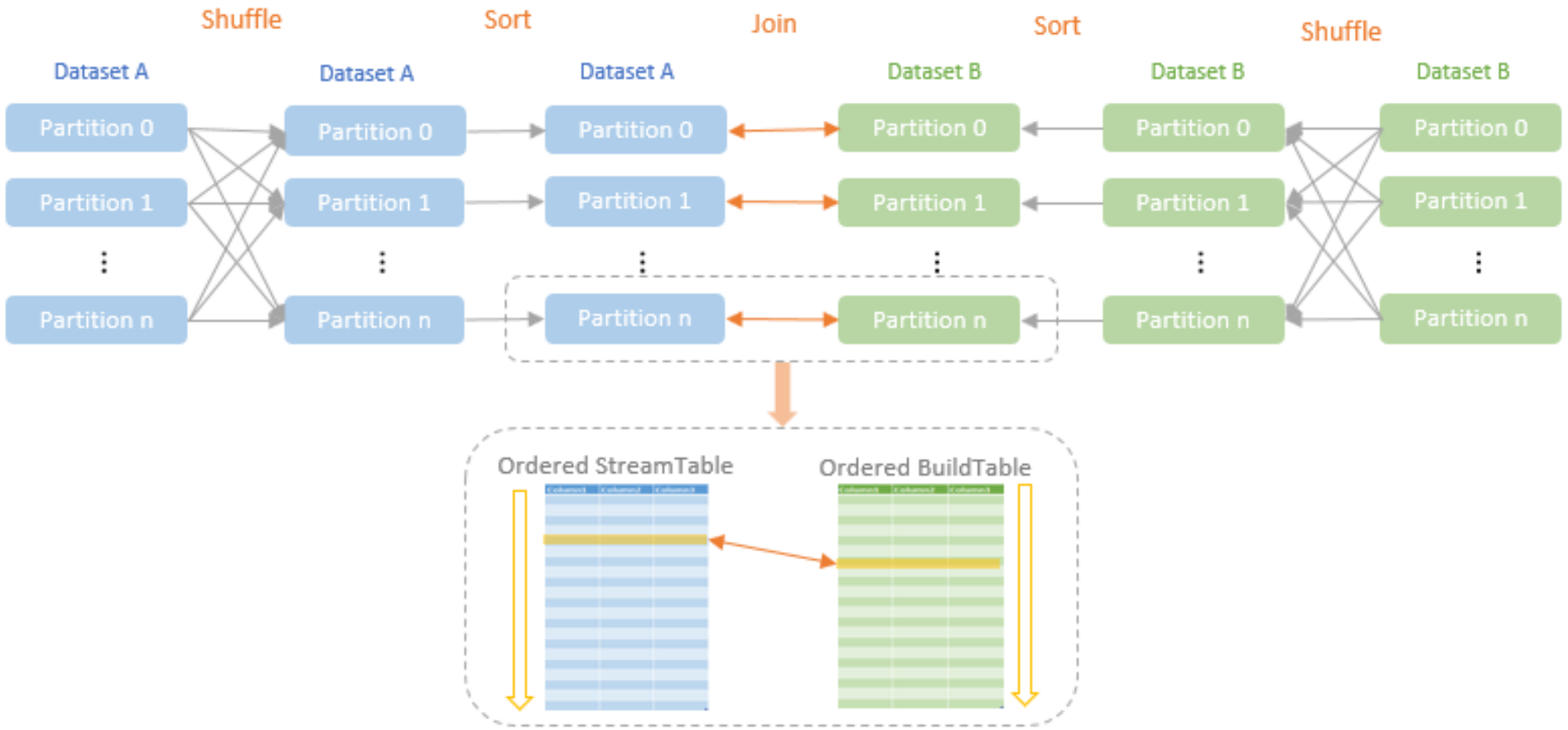
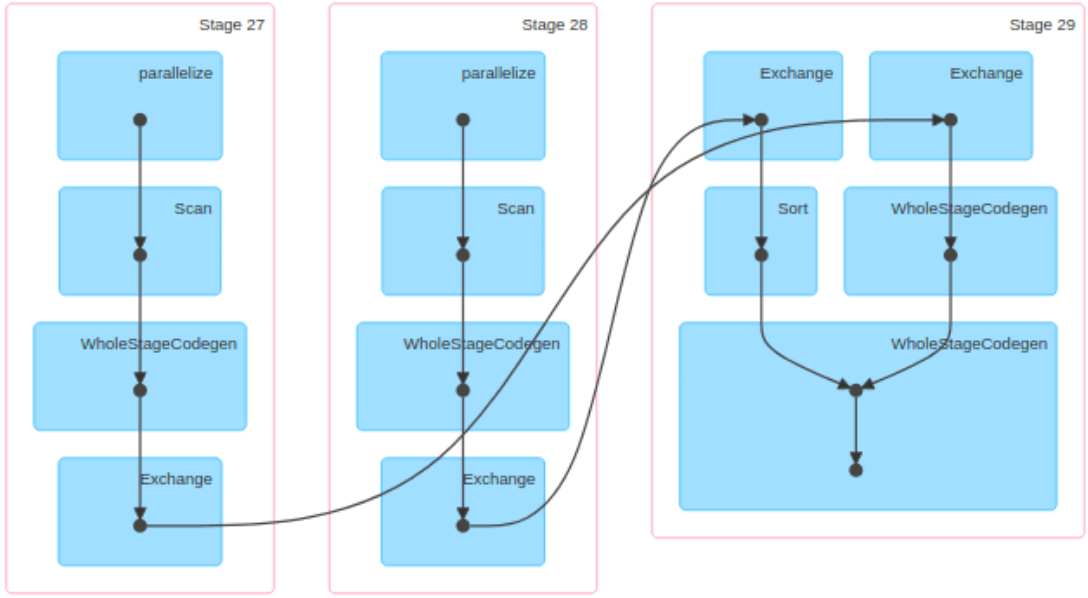
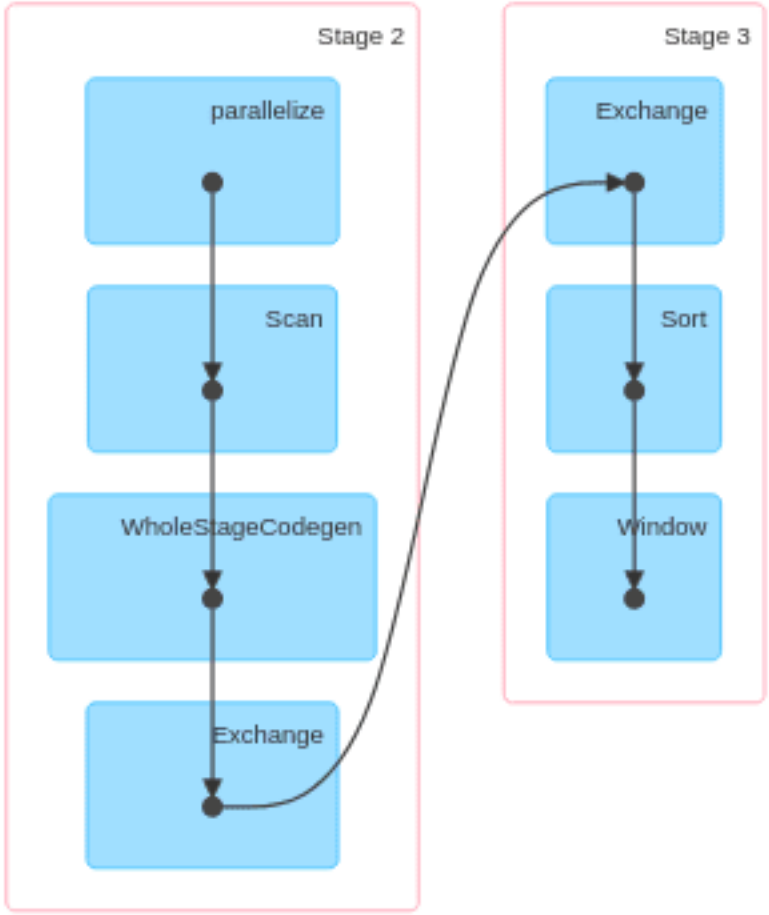


Image from: https://dataninjago.com/2022/01/11/spark-sql-query-engine-deep-dive-11-join-strategies/

Window

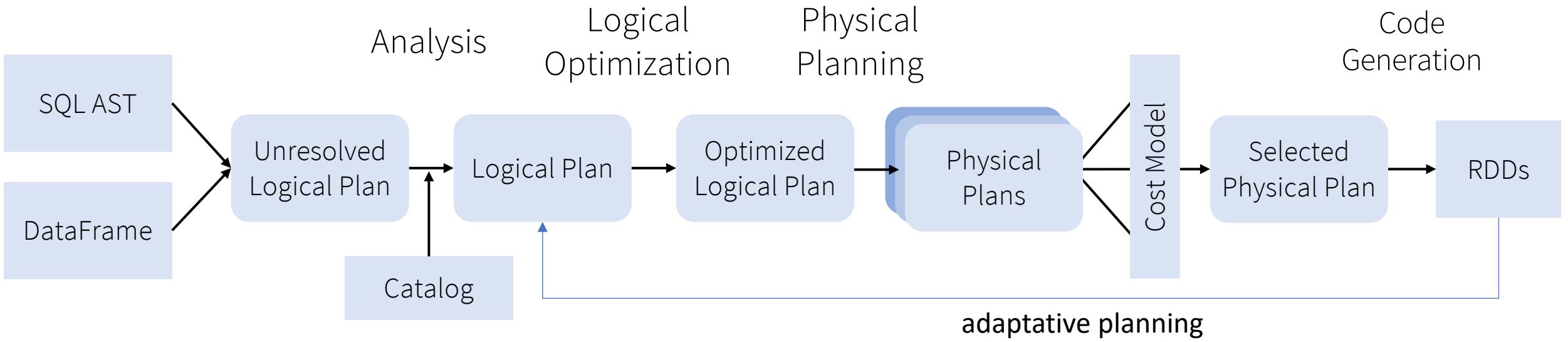
- Performing aggregation on specific columns and keep the results inside the original table as a new feature/column
  - Aggregation followed by a join
- Use of a window function eliminates the need for a join





Adaptive query execution (AQE)

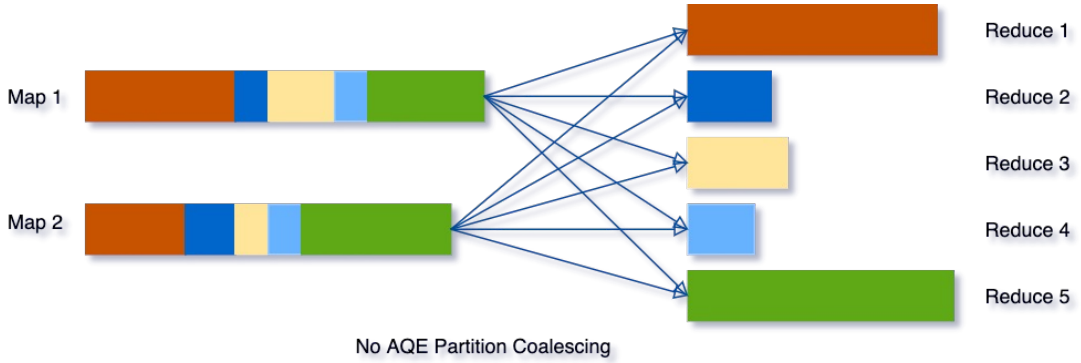
- Challenging and expensive to collect and maintain a set of accurate and up-to-date data statistics in distributed datasets
- Reoptimises based on more accurate runtime statistics



spark.sql.adaptive.enabled = true

AQE dynamically coalescing shuffle partitions

- Automatically balances out the skewness across the partitions



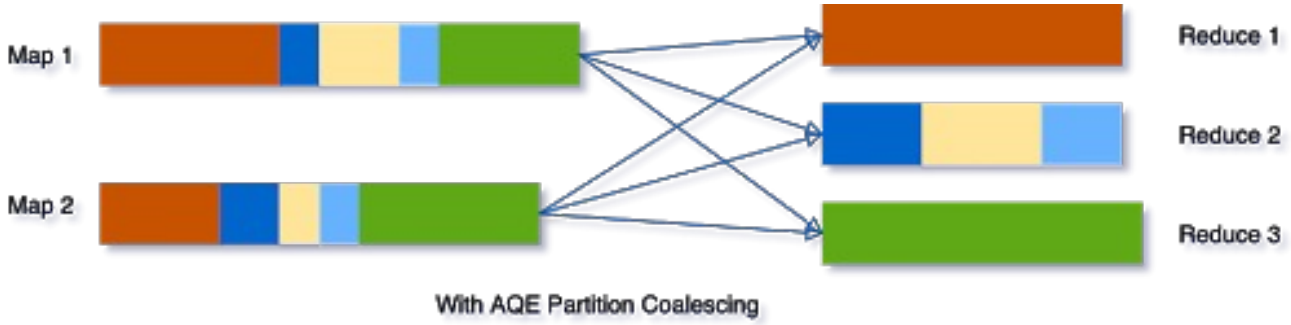


Image from https://www.databricks.com/blog/2020/05/29/adaptive-query-execution-speeding-up-spark-sql-at-runtime.html

spark.sql.adaptive.coalescePartitions.enabled

AQE Dynamically switching join strategies

- Size estimation can go wrong
  - very selective filter, series of complex operators other than just a scan

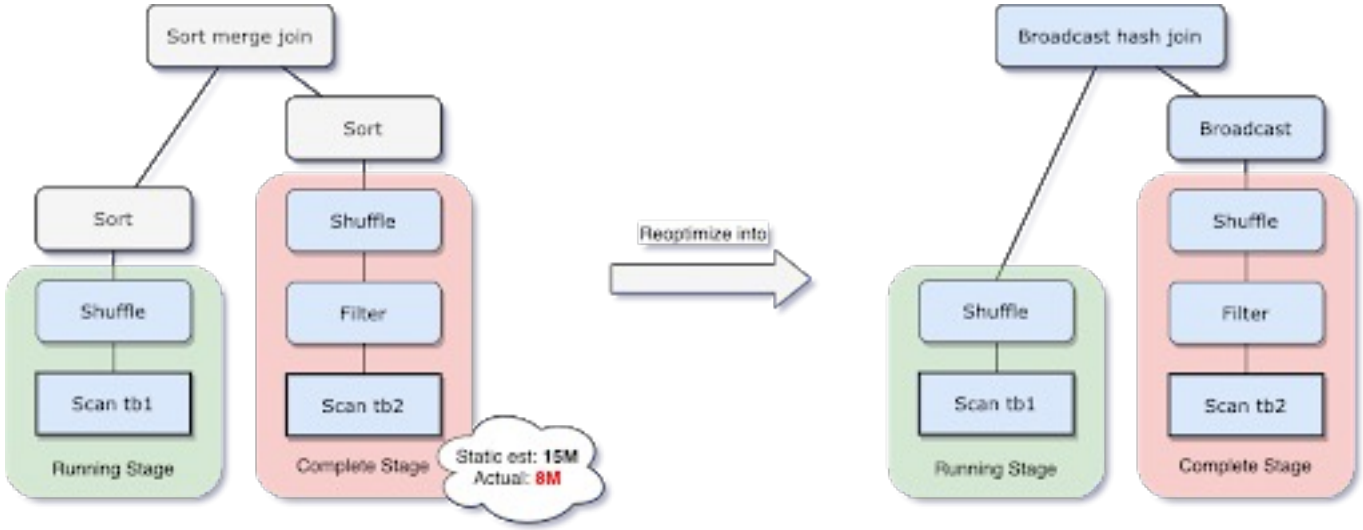
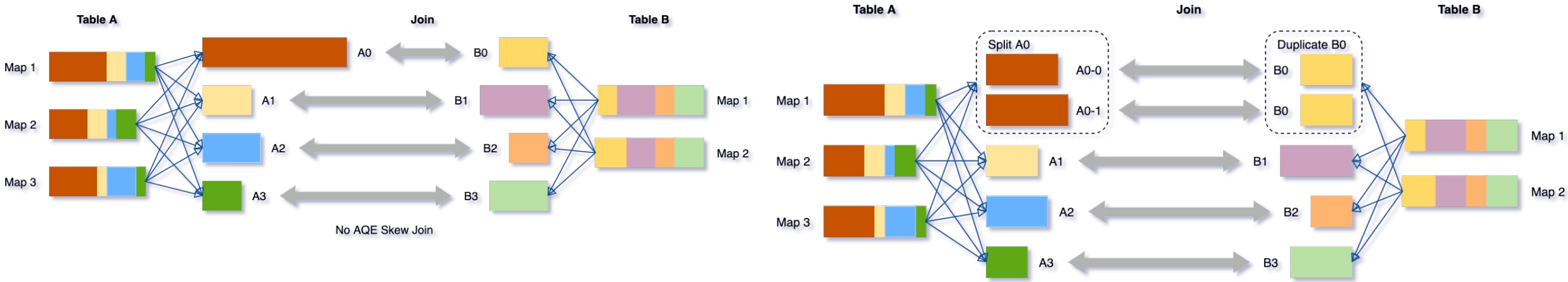


Image from https://www.databricks.com/blog/2020/05/29/adaptive-query-execution-speeding-up-spark-sql-at-runtime.html

spark.sql.adaptive.autoBroadcastJoinThreshold
spark.sql.adaptive.maxShuffledHashJoinLocalMapThreshold

AQE Optimizing Skew Join

- Detects such skew automatically from shuffle file statistics
- Splits the skewed partitions into smaller subpartitions



Images from https://www.databricks.com/blog/2020/05/29/adaptive-query-execution-speeding-up-spark-sql-at-runtime.html

spark.sql.adaptive.skewJoin.enabled

More information

- Spark Performance Tuning by Databricks
- Optimization recommendations on Databricks by DataBricks
- Spark SQL Hints by Databricks
- Adaptive Query Execution: Speeding Up Spark SQL at Runtime by Databricks
- Spark: The Definitive Guide by Bill Chambers, Matei Zaharia, Chapter 19. Performance Tuning
- The Internals of Spark SQL by Jacek Laskowski