PROFINIT

BE0M3BDT
Advanced Spark

Tomáš Duda

15th November 2023

Outline

PROFINIT

- Revision
- Catalyst
- Monitoring and Debugging Spark
- Spark Optimization
- Joins
- Adaptive Query Execution
- Common Issues
    - Reading and writing data
    - Common errors
- Conclusion and Trends

Revision

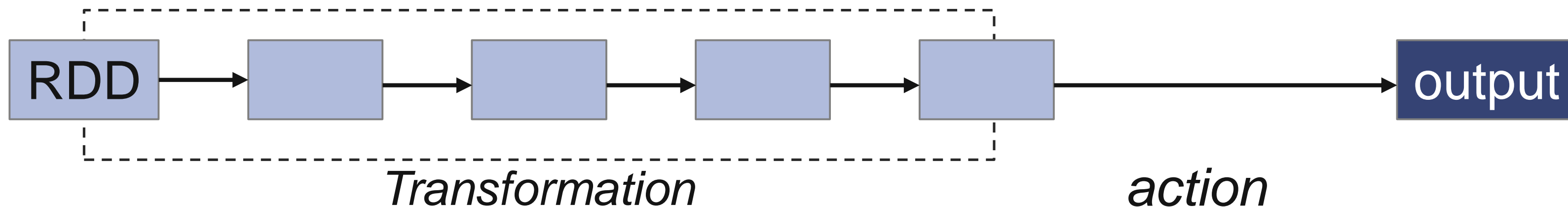
Revision: What is Spark?

- BigData compute engine
- In-memory processing, lazy evaluation
- Supports multiple languages: Scala, Java, Python, R
- Batch and stream processing, machine learning, graph analysis
- Spark program = Transformations + Actions
- Core APIs: RDD and DataFrame

Revision: Spark - RDD

PROFINIT

- RDD: Resilient Distributed Dataset
    - Collection of data
    - Immutable
- Transformations – map, flatMap, filter, …
- Actions – take, count, collect, …
- Computation = DAG



Revision: Spark - Demo

PROFINIT

- Example: Country with the highest avg temperature in summer months

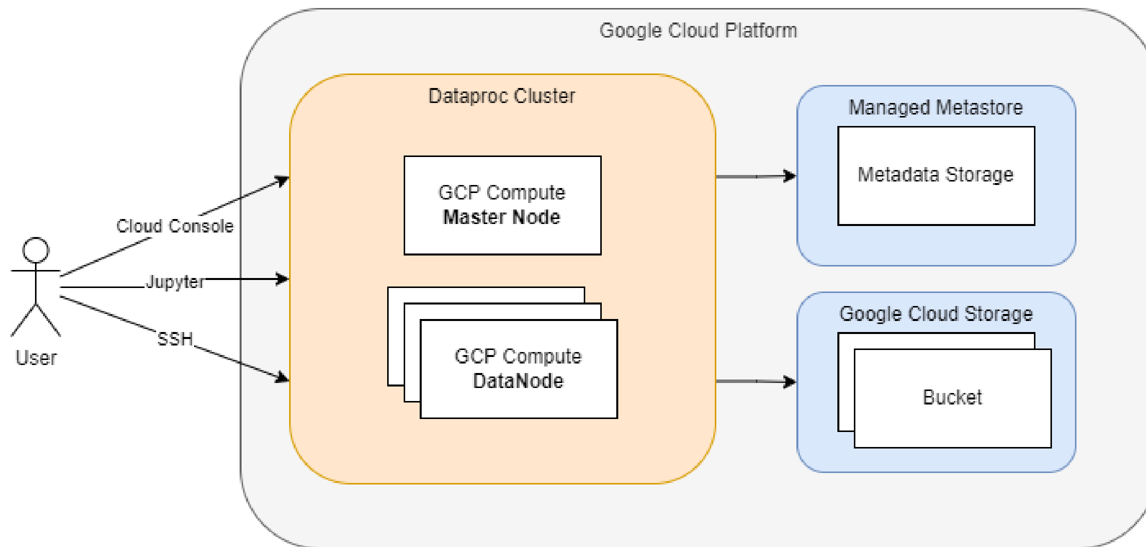
```
sc = SparkContext()
lines = sc.textFile('input.csv')
lines = lines.map(lambda r: r.split(',')).map(lambda r: (r[8], r[1], r[4]))
lines = lines.filter(lambda r: is_float(r[2]))
lines = lines.filter(lambda r: r[1] in ['07', '08', '09'])
lines = lines.map(lambda r: (r[0], (float(r[2]), 1)))
             .reduceByKey(lambda a, b: (a[0] + b[0], a[1] + b[1]))
             .map(lambda r: (r[0], r[1][0] / r[1][1])))
lines = lines.sortBy(lambda r: r[1], ascending=False)
lines = lines.take(1)
```

- **Can it be simplified?**

Revision: Step aside – Demo setup

PROFINIT

- Local testing – Docker image
- Distributed environment – I will be using GCP DataProc
  - Alternatives – DataBricks, AWS EMR, on-premise Hadoop, …



Revision: Spark SQL - DataFrame

PROFINIT

- DataFrame: „RDD with columns“
  - Table-like
  - Immutable
  - With metadata
  - Works with SQL
  - Strong typing (Scala, Java)
- Catalyst optimizer (more on that later)

Revision: Spark SQL - Demo

PROFINIT

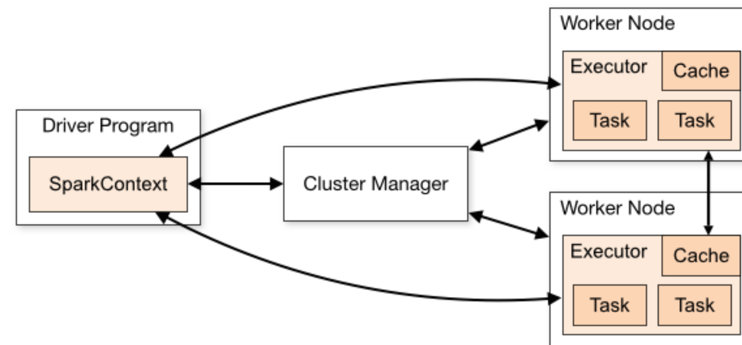
- Example: Country with the highest avg temperature in summer months

```
df.createOrReplaceTempView('temps')
df = spark.sql('SELECT `Country`, AVG(`Temperature`) AS avg_temp
                FROM `temps`
                WHERE `month` IN (7, 8, 9)
                GROUP BY `Country`
                ORDER BY `avg_temp` DESC')
```

Revision: Spark – Concepts

PROFINIT

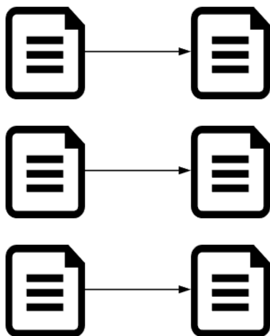
- Computational model
  - Application manager
  - Driver
  - Executors
- Spark program
  - Jobs, stages, tasks
- Deploy mode – local, client, cluster
- Spark partitions (too few x too many)
  - Repartition, coalesce, partitionBy
- Interactive x Batch mode (spark-shell and spark-submit)
- Spark configuration
  - Memory model
  - Cores allocation

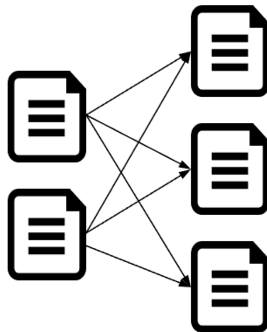


Narrow and Wide Transformations

PROFINIT

- Narrow
  - Does not incur shuffle
  - E.g. map, filter, flatMap, …
- Wide
  - Incurs shuffle and changes the number of partitions
  - E.g. reduceByKey, groupByKey, join, sortBy, …

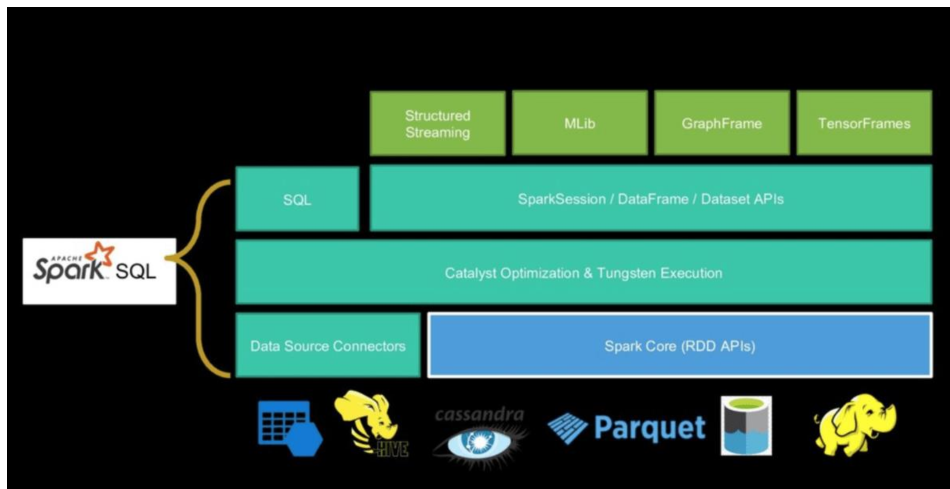




Spark Catalyst

Spark Catalyst – motivation

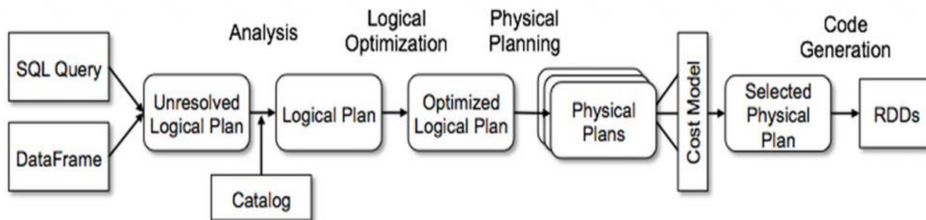
- **Spark SQL** unifies the access to data stored on various systems and in various formats
- Higher-level API enables further optimizations



Spark Catalyst – query planning

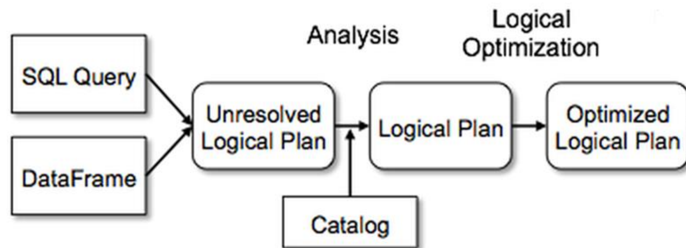
PROFINIT

- Catalyst is the Spark SQL optimizer
- Execution plan is a translation of Spark statements (queries, transformations, actions, …) to a sequence of logical and physical operations (DAG)
- Function **explain()** shows the plan(s)



Spark Catalyst – step by step

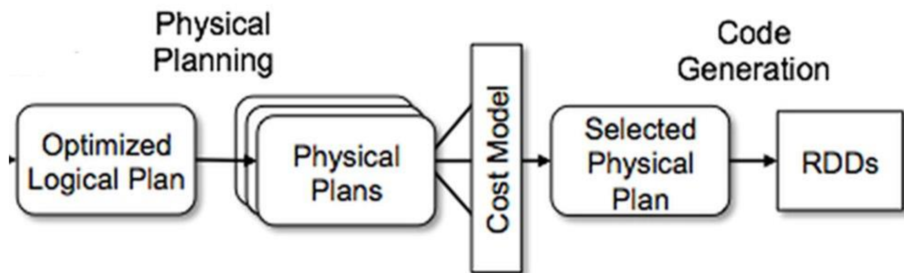
- Unresolved logical plan = Spark’s interpretation of what we want to do
- Logical plan = metadata check, typing (resolution of tables, AnalysisException)
- Optimized logical plan = reordering of operations, simplification (rules executor)



Spark Catalyst – step by step

PROFINIT

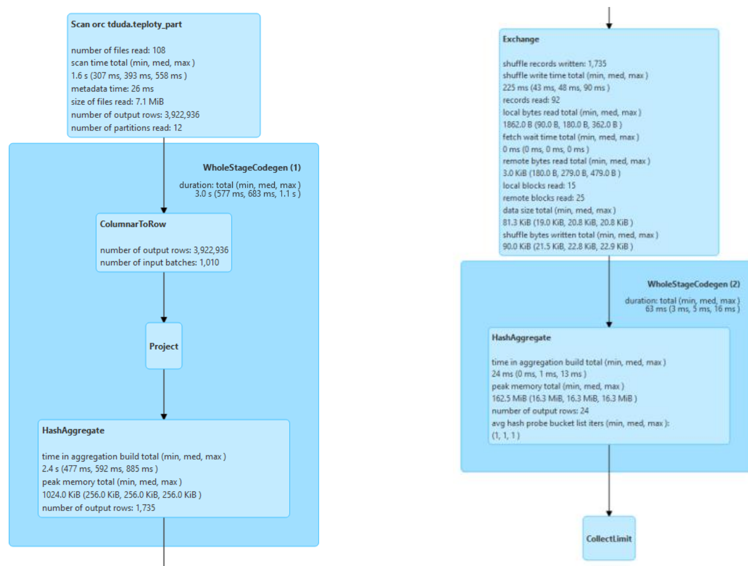
- Physical plan = different ways how to compute the result
- Cost-based optimization (CBO)
- Promoted physical plan = the plan selected for execution
- Since Spark 3 – Adaptive Query Execution



Spark Catalyst – execution plans in Spark HS

PROFINIT

- SELECT country, avg(temp) from temps group by country LIMIT 20



Monitoring and Debugging Spark

Debugging Spark applications - SparkUI

PROFINIT

- SparkUI vs Spark History Server
- Interactive vs Batch jobs
- Common ports – HistoryServer 18080, SparkUI 4040+
    - Beware: Using one node for too many drivers
    - Beware: Hanging interactive sessions
- Available information
    - **Current state**
    - **Statistics**
    - **Effective** configuration
    - Logical and physical **plans**

SparkUI – Applications

PROFINIT

- List of completed and Incomplete applications
- **Name your applications**

History Server

Event log directory: gs://dataproc-temp-europe-west3-650195870162-nldrkjas/ad6c3ec2-0bf0-47aa-afa7-78467fe227d9/spark-job-history

Last updated: 2023-11-11 18:06:40

Client local time zone: Europe/Prague

Search:

Version	App ID	App Name	Driver Host	Attempt ID	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.3.2	application_1699622231247_0014	PySparkShell	cluster-0cb7-m.europe-west3-a.c.experimental-377419.internal		2023-11-11 14:59:02	2023-11-11 16:34:40	1.6 h	root	2023-11-11 16:34:41	Download
3.3.2	application_1699622231247_0013	word_count.py	cluster-0cb7-w-1.europe-west3-a.c.experimental-377419.internal	1	2023-11-11 14:55:14	2023-11-11 14:55:31	17 s	tomas_duda27	2023-11-11 14:55:31	Download
3.3.2	application_1699622231247_0012	word_count.py	cluster-0cb7-w-1.europe-west3-a.c.experimental-377419.internal	2	2023-11-11 14:53:22	2023-11-11 14:53:28	6 s	tomas_duda27	2023-11-11 14:53:28	Download

SparkUI – Application and Job details

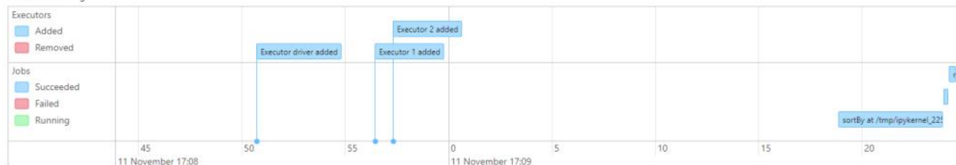
PROFINIT

- Application timeline – Jobs: ID, duration, stages, tasks
- **Drill down – Job – DAG – Stages – Tasks**
- Input and output size, amount of shuffled data

Spark Jobs (?)

User: root
Total Uptime: 1.6 min
Scheduling Mode: FAIR
Completed Jobs: 3

Event Timeline
Enable zooming

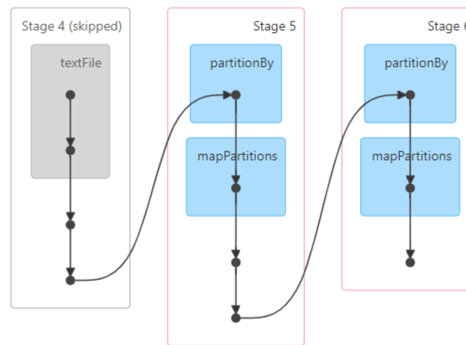


Completed Jobs (3)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
2	runJob at PythonRDD.scala:166 runJob at PythonRDD.scala:166	2023/11/11 17:09:24	0.3 s	2/2 (1 skipped)	3/3 (2 skipped)
1	sortBy at /tmp/ipykernel_225842/1255599889.py:19 sortBy at /tmp/ipykernel_225842/1255599889.py:19	2023/11/11 17:09:23	0.2 s	1/1 (1 skipped)	2/2 (2 skipped)
0	sortBy at /tmp/ipykernel_225842/1255599889.py:19 sortBy at /tmp/ipykernel_225842/1255599889.py:19	2023/11/11 17:09:18	5 s	2/2	4/4

DAG Visualization



Completed Stages (2)

Page: 1

Stage Id ▾	Description		Submitted
6	runJob at PythonRDD.scala:166	+details	2023/11/11 17:09:24
5	sortBy at /tmp/ipykernel_225842/1255599889.py:19	+details	2023/11/11 17:09:24

SparkUI – Storage, Environment, Executors

PROFINIT

- Amount of cached or persisted data
- Spilled data on disk
- Visible only for running jobs (SparkUI)

Storage

RDDs

ID	RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
1	rdd	Memory Serialized 1x Replicated	5	100%	236.0 B	0.0 B
4	LocalTableScan [count#7, name#8]	Disk Serialized 1x Replicated	3	100%	0.0 B	2.1 KiB

- Effective configuration

Environment

Runtime Information

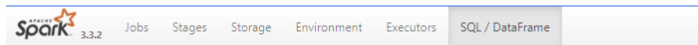
Name	Value
Java Home	/usr/lib/jvm/temurin-11-jdk-amd64
Java Version	11.0.20.1 (Eclipse Adoptium)
Scala Version	version 2.12.18

Spark Properties

Name	Value
spark.app.id	application_1699622231247_0017
spark.app.name	Demo - Intro
spark.app.startTime	1699722523892
spark.app.submitTime	1699722523668

SparkUI – SQL / DataFrame

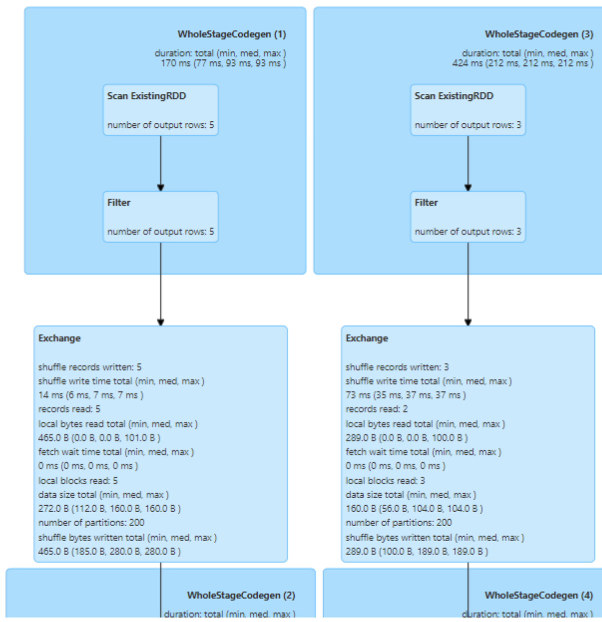
- Execution plans and metrics



Details for Query 0

Submitted Time: 2023/11/11 17:40:10
Duration: 13 s
Running Jobs: 0

☐ Show the Stage ID and Task ID that corresponds to the max metric



Debugging Spark applications

PROFINIT

- What to look for in Spark UI?
  - Failing tasks
  - Data spill
  - Pending tasks – possible skew
- Where are the logs?
  - Depends on Spark deploy mode
  - Interactive sessions
  - Batch jobs with local or client mode – standard output
  - Batch jobs with cluster mode – must be retrieved from executor

Spark Optimization

What to optimize?

- CPU, memory, storage, network
- What are the most expensive operations?
    - Data serialization and deserialization
    - Shuffles – wide vs narrow transformations
    - CPU is rarely the bottleneck
- How can we optimize?
    - Adjust configuration
    - Write more effective code
    - Optimize storage – inputs
- Be always aware how much we can benefit from optimization

General recommendations

PROFINIT

- Know your data
- Check for data quality
- Check for data stability
    - Version your interfaces
- Collect all useful information (monitoring, logging)

Critical Spark configuration

PROFINIT

- Many problems can be solved by adjusting the configuration
- Deploy mode of batch jobs
  - Avoid running too many drivers on a single node
- Parallelism
  - spark.default.parallelism, spark.sql.shuffle.partitions
  - spark.executor.instances
  - spark.executor.cores
- Memory configuration
  - spark.driver.memory
  - spark.executor.memory
- Dynamic allocation
  - spark.dynamicAllocation – long running jobs, shared cluster
  - spark.dynamicAllocation.minExecutors
  - spark.dynamicAllocation.maxExecutors

Spark Joins

Spark joins

PROFINIT

- What scenarios can occur?
  - Small + Small
  - Small + Large
  - Large + Large
- Does the input table grow over time?
- Do we really need to send all data into join?
- Join types
  - INNER, OUTER, SEMI, ANTI, CROSS
- Join strategies
  - Sort-Merge Join
  - Broadcast Join

Spark joins – Types

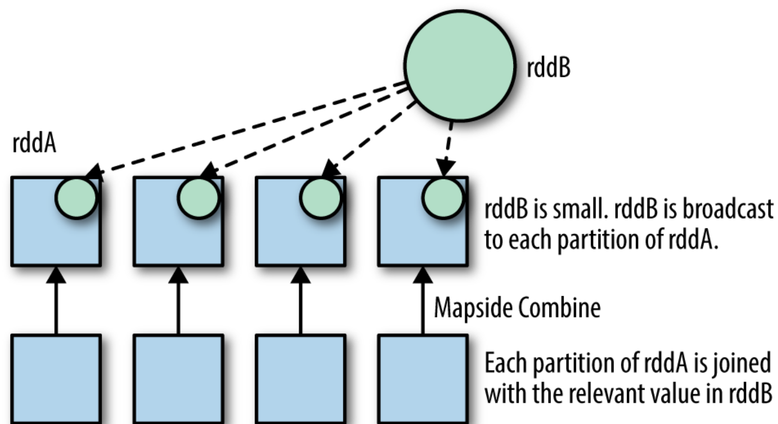
- INNER
- LEFT | RIGHT | FULL OUTER
- LEFT | RIGHT SEMI
- LEFT | RIGHT ANTI
- CROSS

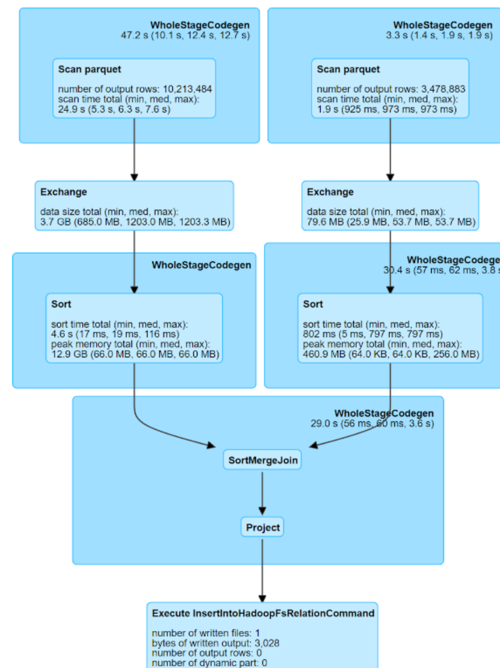
Name	Company
Jack	Apple
John	Unknown
Lucy	Microsoft
Elisabeth	Google

Company	Employees
Apple	161 000
Google	182 000
Microsoft	221 000

Spark joins – Strategies

- SortMerge Join
- Broadcast Join (Hash/Nested Loop)
- How to find out which algorithm Spark chose?
  - History server
- How to force Spark to use a different join strategy?
  - Join hints



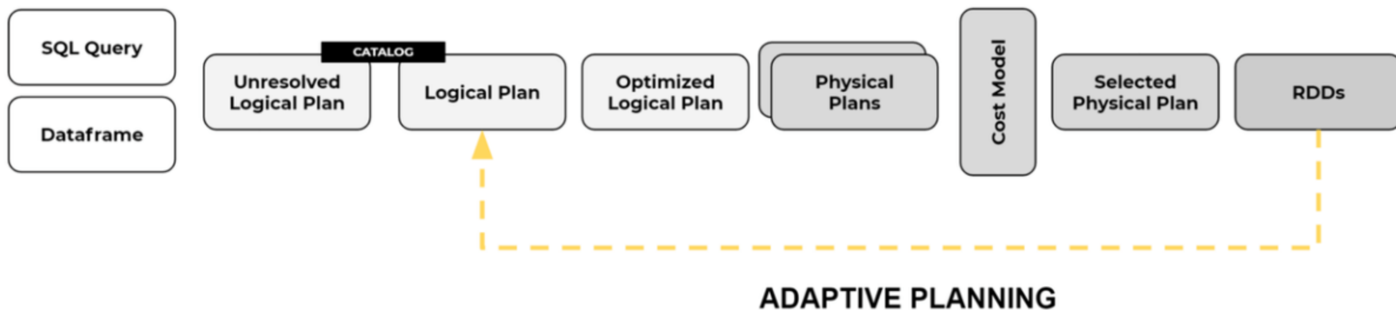


Adaptive Query Execution (AQE)

Adaptive Query Execution

PROFINIT

- New feature in Spark 3
  - spark.sql.adaptive.enabled
  - Enabled by default since Spark 3.2
- Execute on a part of data and recompute plan



Adaptive Query Execution – Optimizations

PROFINIT

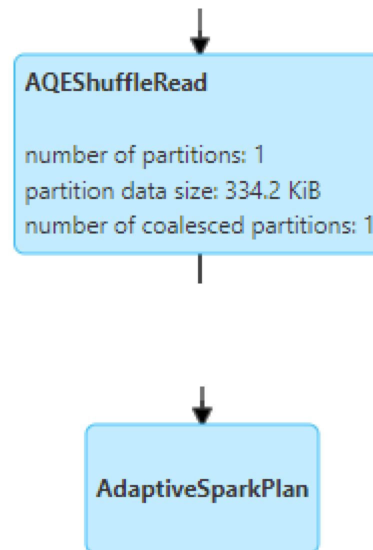
- Shuffle optimization
  - Coalesce post shuffle partitions – avoid too small partitions
- Join strategy optimization
  - Use faster strategy on small data
  - Sort-merge join to broadcast join conversion
  - Sort-merge join to shuffled hash join conversion
- Optimizing Skew Join
  - Split tasks in skewed merge-joins
  - Avoid pending tasks

Adaptive Query Execution – Demo

PROFINIT

Details

```
== Physical Plan ==
AdaptiveSparkPlan (24)
+- == Final Plan ==
   TakeOrderedAndProject (14)
   +- * HashAggregate (13)
      +- AQEShuffleRead (12)
         +- ShuffleQueryStage (11), Statistics(sizeInBytes=221.7 KiB, rowCount=7.10E+3)
            +- Exchange (10)
               +- * HashAggregate (9)
                  +- * Project (8)
                     +- * BroadcastHashJoin Inner BuildRight (7)
                        :- * Filter (2)
                        :  +- Scan hive default.customers (1)
                        +- BroadcastQueryStage (6), Statistics(sizeInBytes=32.0 MiB, rowCount=245)
                           +- BroadcastExchange (5)
                              +- * Filter (4)
                                 +- Scan hive default.countries (3)
+- == Initial Plan ==
   TakeOrderedAndProject (23)
   +- HashAggregate (22)
      +- Exchange (21)
         +- HashAggregate (20)
            +- Project (19)
               +- BroadcastHashJoin Inner BuildRight (18)
                  :- Filter (15)
                  :  +- Scan hive default.customers (1)
                  +- BroadcastExchange (17)
                     +- Filter (16)
                        +- Scan hive default.countries (3)
```



Common Issues:
Reading and Writing Data

Small files

PROFINIT

- Beware of small input files

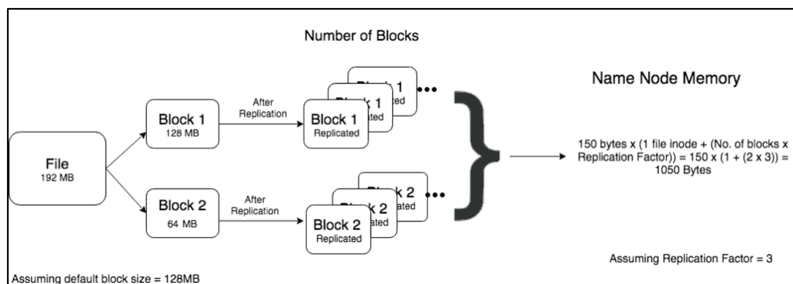
Number of files (parquet)	Size of file	Scan time (sum over executors)
10	50 MB	4.1 s
100	5 MB	5.0 s
1000	0.5 MB	39.3 s

Small files – mitigation

PROFINIT

- Remember when writing data from Spark job (default shuffle!)
- One Spark partition = 1 part-file written
- Repartition, coalesce
- Ideal part-file is between 128MB and 1GB

- **Avoid small files in Hadoop in general**



Small files – writing data

PROFINIT

- Especially when the job significantly reduces the amount of data
- Adaptive Query Execution: `spark.sql.adaptive.coalescePartitions.enabled`
- Use coalesce: `df.coalesce(n).write.mode(…).format(…)`

Buckets > spark-experiments > data > small_files

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER

Filter by name prefix only ▾ Filter Filter objects and f

Name	Size
_SUCCESS	0 B
part-00000-418fdae4-7999-444e-...	770.6 KB
part-00001-418fdae4-7999-444e-...	770.5 KB
part-00002-418fdae4-7999-444e-...	770.3 KB
part-00003-418fdae4-7999-444e-...	770.4 KB
part-00004-418fdae4-7999-444e-...	770.3 KB
part-00005-418fdae4-7999-444e-...	770.3 KB
part-00006-418fdae4-7999-444e-...	770.4 KB

Buckets > spark-experiments > data > small_files

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER TRANSFER DATA ▾

Filter by name prefix only ▾ Filter Filter objects and folders

Name	Size
_SUCCESS	0 B
part-00000-e768d9db-0666-4d2b-ba3c-dc552ae4a2aa-c000.csv	75.2 MB

Small files – compaction

PROFINIT

- Algorithm
- Determine the size of your dataset on disk
- Decide what your ideal part-file size is
- Compute the number of spark-partitions required (size-on-disk / ideal-size)
- Read data, repartition by N and write to disk

```
sourceDF = (spark
  .read                      # Get the DataFrameReader
  .parquet(srcPath)          # Read in the parquet file
  .repartition(partitions) # One spark-partition per part-file on disk
  .write                     # Get the DataFrameWriter
  .mode("overwrite")         # In case the file already exists
  .parquet(dstPath)          # Write out the parquet file
)
```

Reading data, writing data

PROFINIT

- Data format and compression
    - Choosing the most effective format – columnar, row-based
    - Compression – snappy by default for AVRO and PARQUET
    - https://spark.apache.org/docs/latest/sql-data-sources.html

Common Issues: Frequent errors

Broadcast join

PROFINIT

- Typical error
  - `Caused by: org.apache.spark.SparkException: Could not execute broadcast in 300 secs. You can increase the timeout for broadcasts via spark.sql.broadcastTimeout or disable broadcast join by setting spark.sql.autoBroadcastJoinThreshold to -1`

- How to solve it
  - Switch off broadcasting
    - `spark.sql.autoBroadcastJoinThreshold=-1`
  - Increase timeout from 300
    - `spark.sql.broadcastTimeout=1000`
  - Refresh table stats:
    - `ANALYZE TABLE <tableName> COMPUTE STATISTICS`

Driver out of memory

PROFINIT

- Typical error
  - `org.apache.spark.SparkException: Job aborted due to stage failure: Total size of serialized results of 3800 tasks (1024.2 MB) is bigger than spark.driver.maxResultSize (1024.0 MB)`

- How to solve it
  - Increase driver memory
    - `spark.driver.memory=2G`
  - Increase maxResultSize
    - `spark.driver.maxResultSize=2G`
  - Avoid .collect() on large results
    - Write data to file
    - Use .show() or .take(n) for data exploration

Executor out of memory

PROFINIT

- Typical error
  - `org.apache.spark.SparkException: Job aborted due to stage failure: Task 251 in stage 10.0 failed 4 times, most recent failure: Lost task 251.3 in stage 10.0: org.apache.spark.memory.SparkOutOfMemoryError: Unable to acquire 16384 bytes of memory, got 0`
- How to solve it
  - Check Spark UI
    - Increase executor memory size
      - `spark.executor.memory`
    - Increase number of executors
      - `spark.dynamicAllocation.maxExecutors`
    - Repartition data
      - `repartition(numPartitions, *cols)`
  - Rewrite code

Conclusion

When the problem is outside Spark

PROFINIT

- Take a step back
- Input validation
- Are the assumptions about our data coded into our programs?
- Did the input/output system change? Is it versioned?
- Is there reliable monitoring?
- Do we know how to reprocess the data on failure?

Summary

PROFINIT

- Monitoring Spark applications
    - Spark UI
    - Logs
- Spark optimization – focus on expensive operations
- Joins in Spark – know types, execution strategies
- Adaptive Query Execution – use Spark 3.2+ when possible
- Common issues
    - Small files
    - Memory errors
- Spark in Cloud – still need to write effective jobs (~ cost optimization)
- Refer to documentation when in doubt

Any questions?

PROFINIT

Thank you for your attention

Profinit EU, s.r.o.
Tychonova 2, 160 00 Praha 6

Tel.: + 420 224 316 016, web: www.profinit.eu