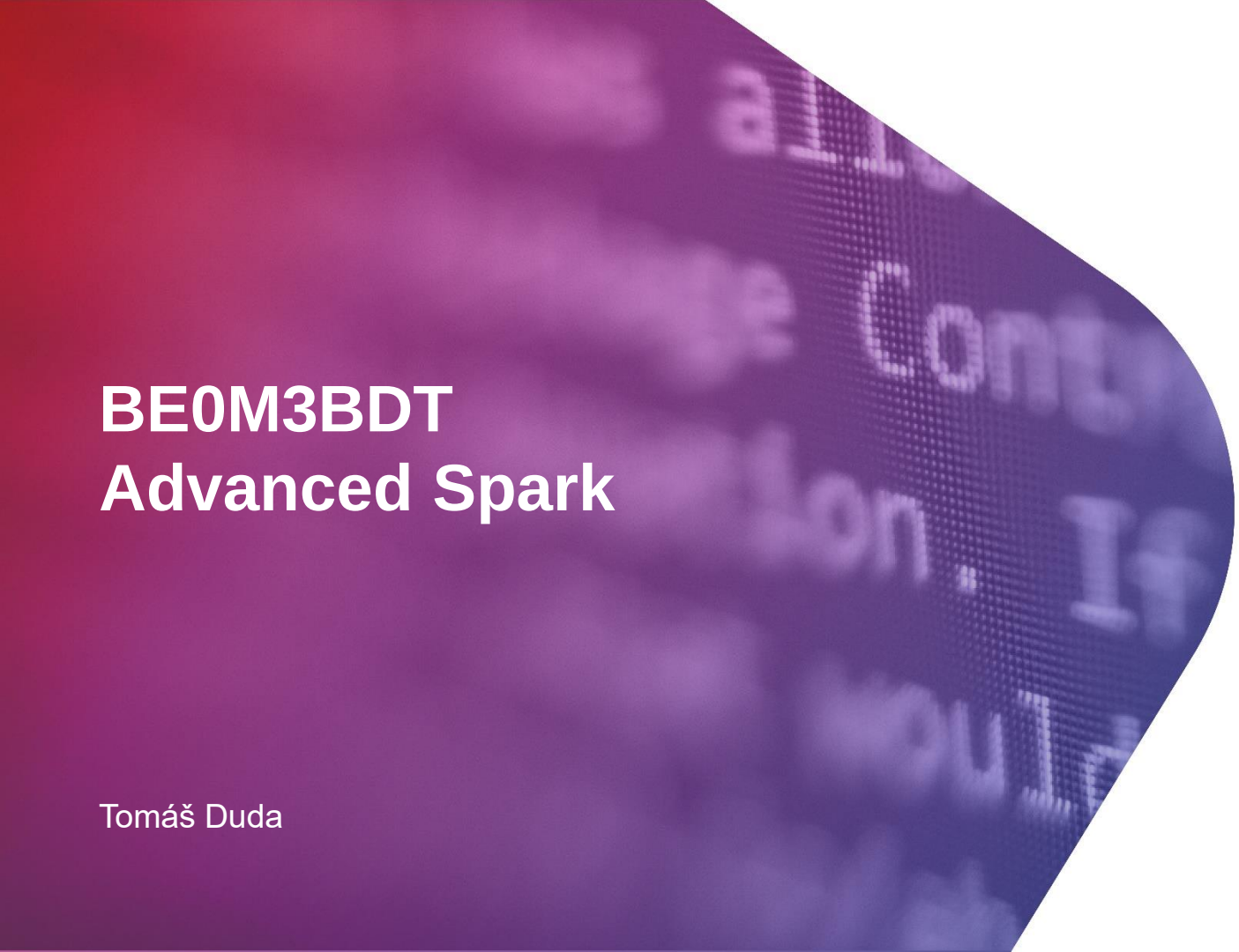


PROFINIT

BE0M3BDT
Advanced Spark

Tomáš Duda

20th November 2024

Outline

PROFINIT

- Revision
- Spark Catalyst
- Monitoring and Debugging Spark
- Spark Joins
- Adaptive Query Execution
- Conclusion

Revision

Revision: What is Spark?

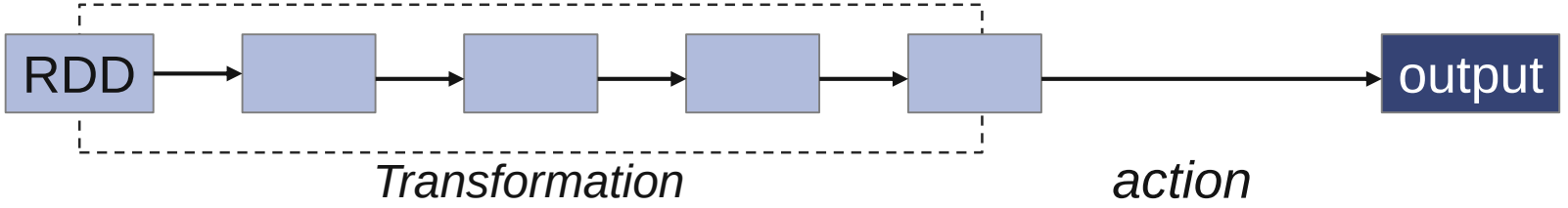
PROFINIT

- 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
  - Parallelizable
- Transformations – map, flatMap, filter, …
- Actions – take, count, collect, …
- Computation = DAG



Revision: Spark - Demo

PROFINIT

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

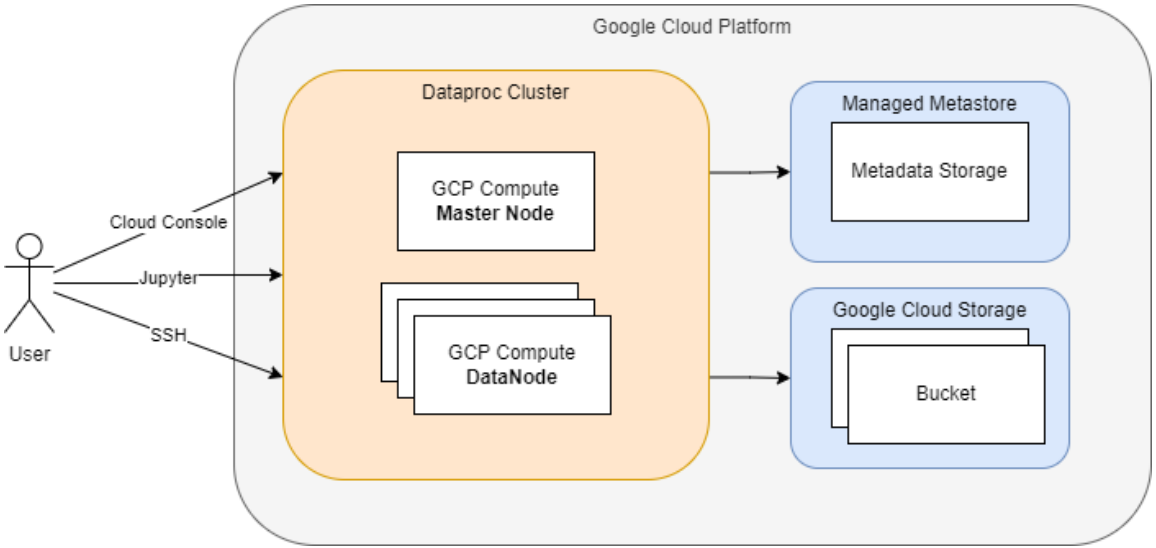
```
sc = SparkContext()
lines = sc.textFile('input.csv')
lines = lines.map(lambda r: r.split(',')).map(lambda r: (r[8], r[1], r[4]))
lines = lines.filter(lambda r: is_float(r[2]))
lines = lines.filter(lambda r: r[1] in ['07', '08', '09'])
lines = lines.map(lambda r: (r[0], (float(r[2]), 1)))
             .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

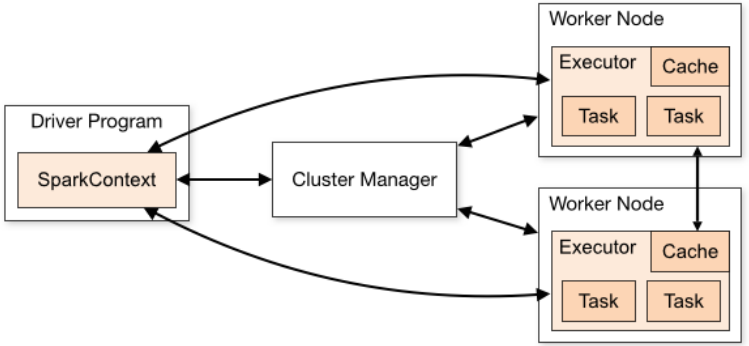
- Example: Country with the highest avg temperature in summer months

```
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

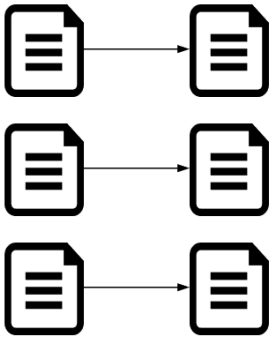
- Computational model
  - Application manager
  - Driver
  - Executors
- Spark program
  - Jobs, stages, tasks
- Deploy mode – local, client, cluster
- Spark partitions (too few x too many)
  - Repartition, coalesce, partitionBy
- Interactive x Batch mode (spark-shell and spark-submit)

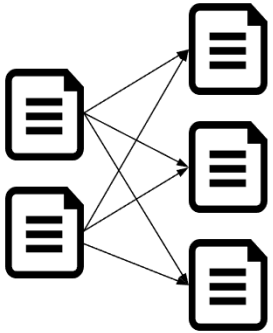


Narrow and Wide Transformations

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, …



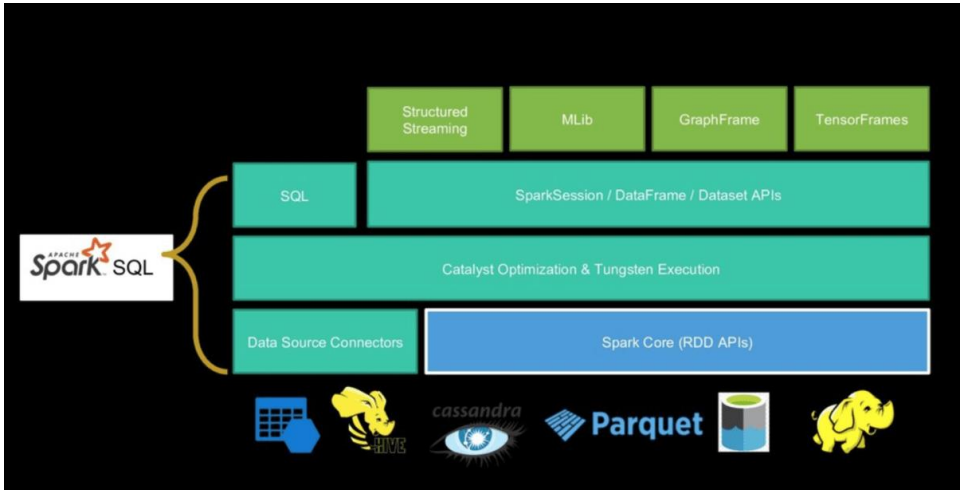


Spark Catalyst

Spark Catalyst – motivation

PROFINIT

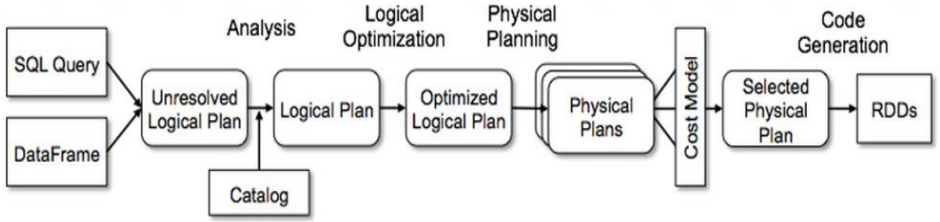
- **Spark SQL** unifies the access to data stored on various systems and in various formats
- Higher-level API enables further optimizations



Spark Catalyst – query planning

PROFINIT

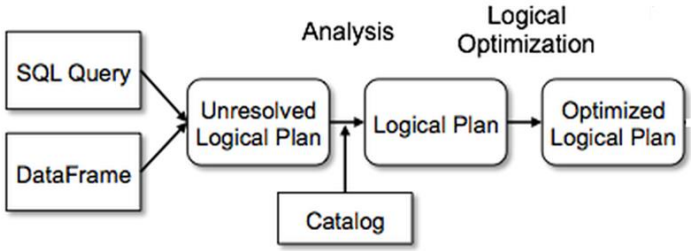
- Catalyst is the Spark SQL optimizer
- Execution plan is a translation of Spark statements (queries, transformations, actions, …) to a sequence of logical and physical operations (DAG)
- Function **explain()** shows the plan(s)



Spark Catalyst – step by step

PROFINIT

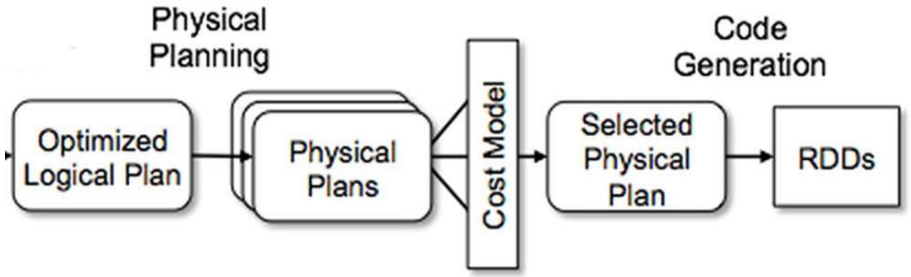
- Unresolved logical plan = Spark's interpretation of what we want to do
- Logical plan = metadata check, typing (resolution of tables, AnalysisException)
- Optimized logical plan = reordering of operations, simplification (rules executor)



Spark Catalyst – step by step

PROFINIT

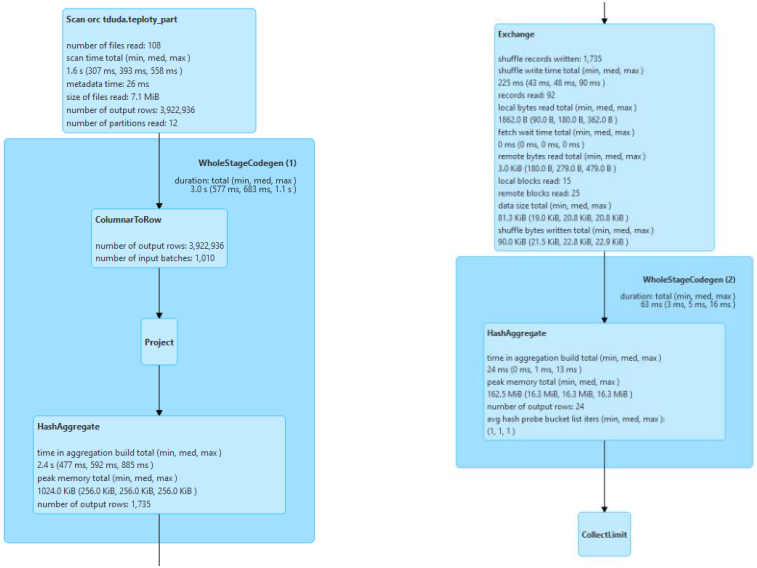
- Physical plan = different ways how to compute the result
- Cost-based optimization (CBO)
- Promoted physical plan = the plan selected for execution
- Since Spark 3 – Adaptive Query Execution



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

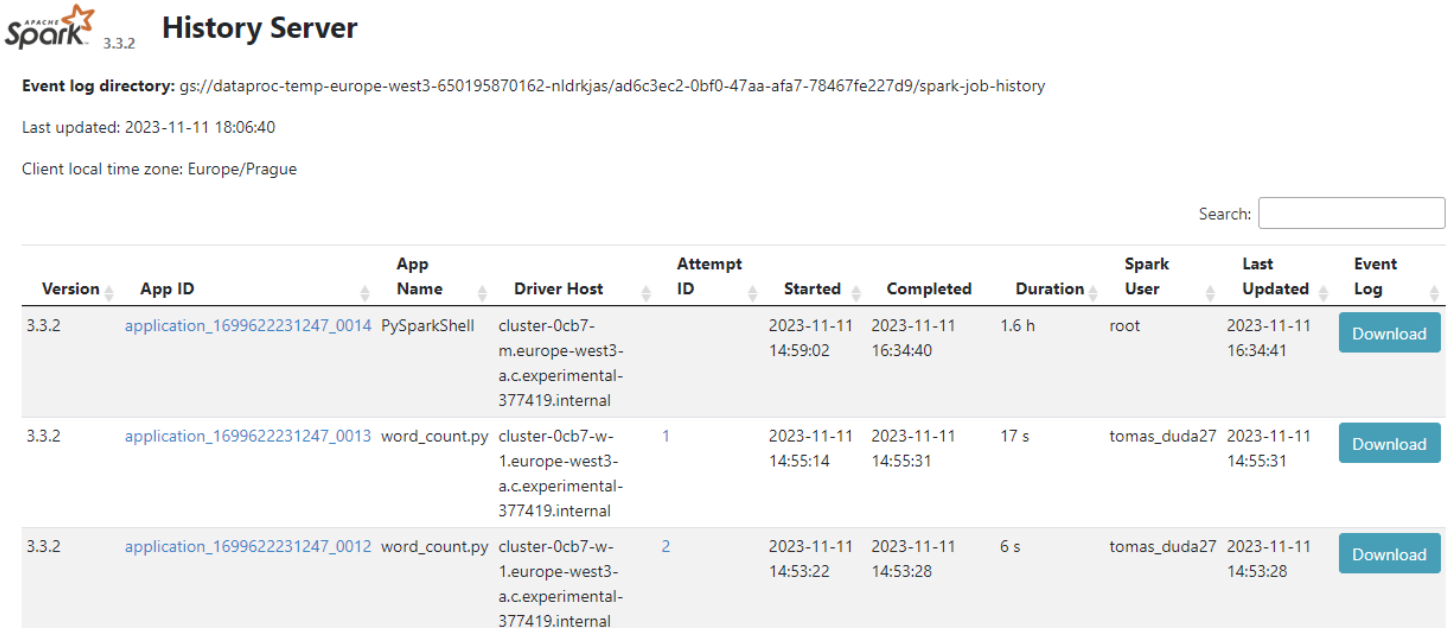
PROFINIT

- 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**



Spark 3.3.2 **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

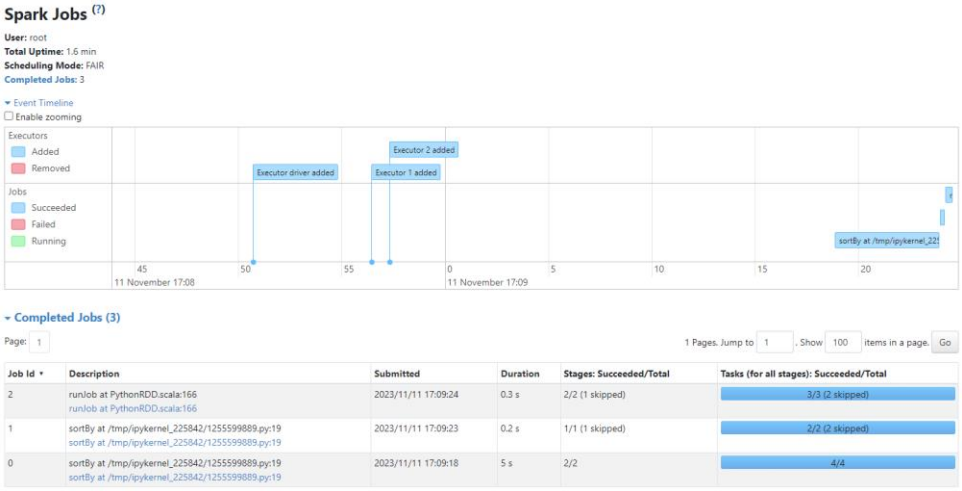
Search:

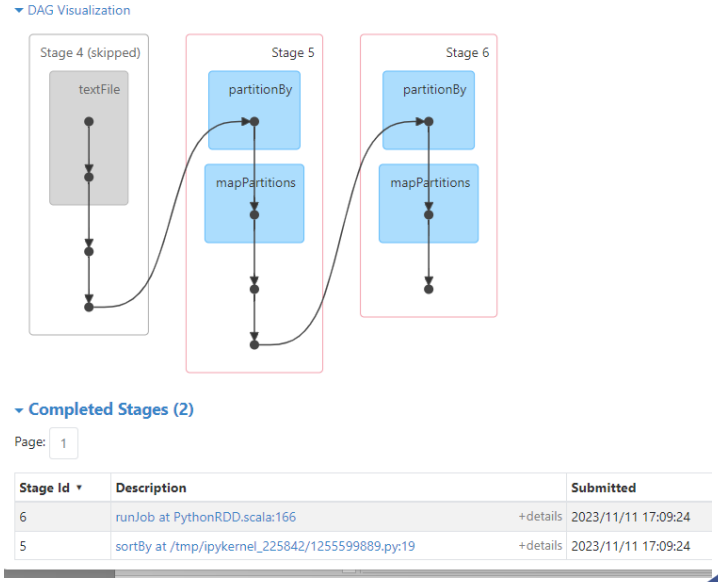
Version	App ID	App Name	Driver Host	Attempt ID	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.3.2	application_1699622231247_0014	PySparkShell	cluster-0cb7-m.europe-west3-a.c.experimental-377419.internal		2023-11-11 14:59:02	2023-11-11 16:34:40	1.6 h	root	2023-11-11 16:34:41	Download
3.3.2	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

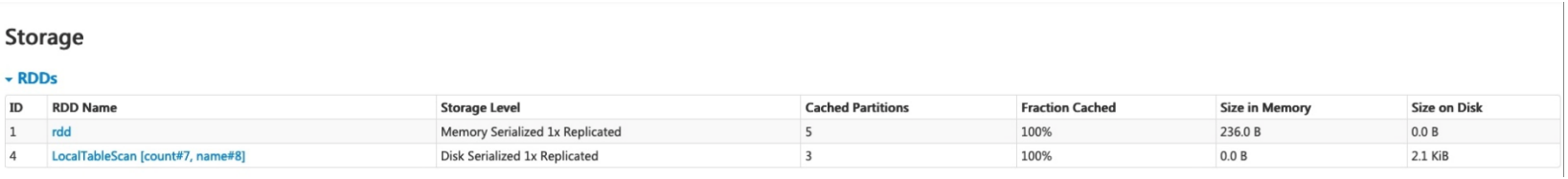




SparkUI – Storage, Environment, Executors

PROFINIT

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

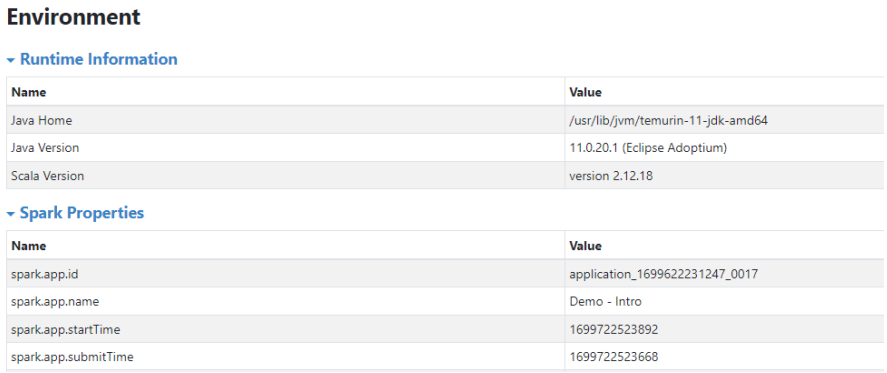


Storage

▾ 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

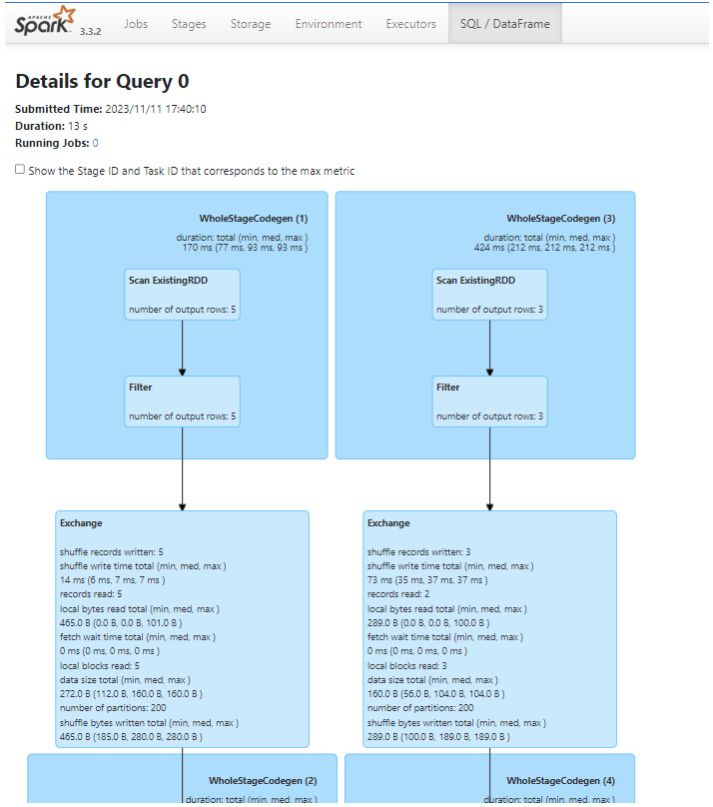
▾ Spark Properties

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

SparkUI – SQL / DataFrame

PROFINIT

- Execution plans and metrics



Debugging Spark applications

PROFINIT

- What to look for in Spark UI?
  - Failing tasks
  - Data spill – skew, large joins, explode on large array
  - 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 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

PROFINIT

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

Name	Company
Jack	Apple
John	Unknown
Lucy	Microsoft
Elisabeth	Google

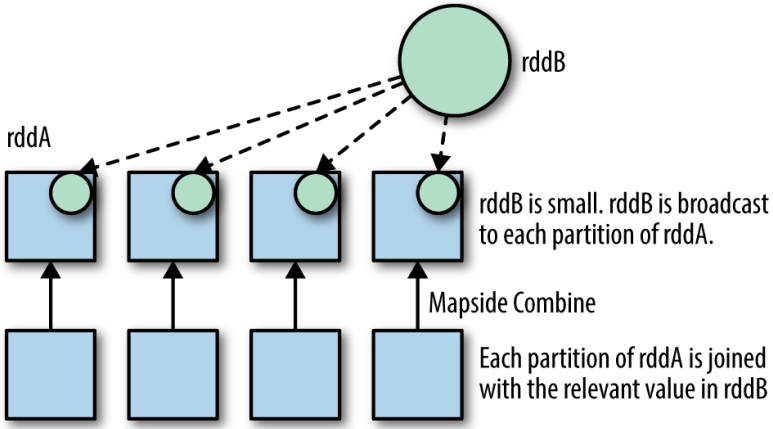
?

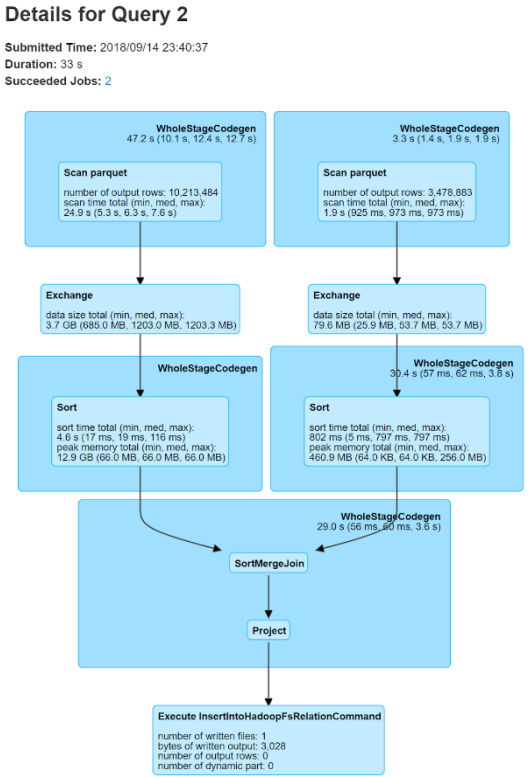
Company	Employees
Apple	161 000
Google	182 000
Microsoft	221 000

Spark joins – Strategies

PROFINIT

- 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



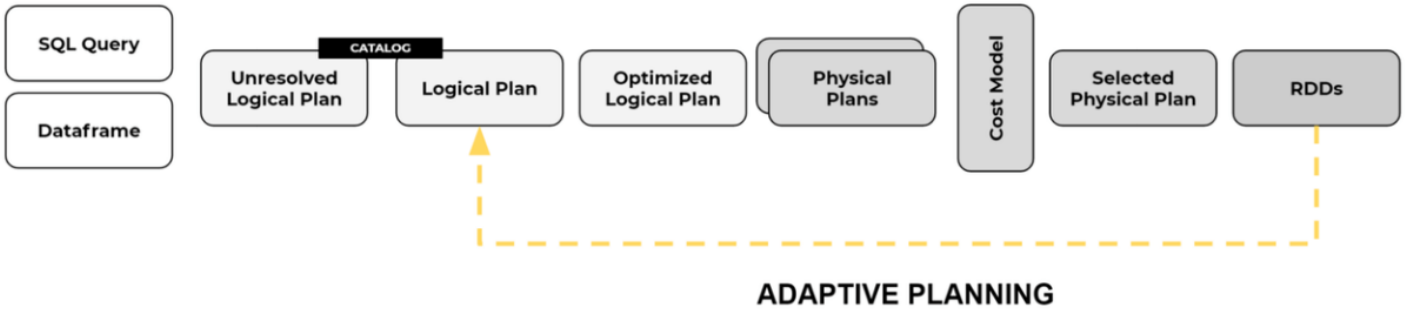


Adaptive Query Execution (AQE)

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

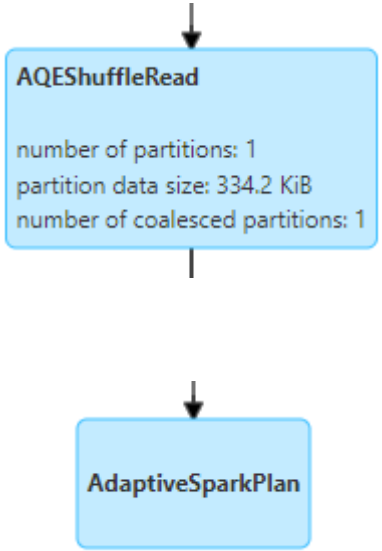
- Shuffle optimization
  - Coalesce post shuffle partitions – avoid too small partitions
- Join strategy optimization
  - Use faster strategy on small data
  - Sort-merge join to broadcast join conversion
  - Sort-merge join to shuffled hash join conversion
- Optimizing Skew Join
  - Split tasks in skewed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



Conclusion

Sometimes 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
- Spark in Cloud – still need to write effective jobs (~ cost optimization)
- Refer to documentation when in doubt

PROFINIT

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

LINKEDIN
linkedin.com/company/profinit

TWITTER
@profinit_EU

FACEBOOK
facebook.com/Profinit.EU

YOUTUBE
Profinit EU, s.r.o.