ADBMS to Apache Spark Auto Migration Framework

Edward Zhang,
Software Engineer Manager, Data Service & Solution (eBay)

#SAISDD7

Who We Are

- Data Service & Solution team in eBay
- Responsible for big data processing and data application development
- Focus on batch auto migration and Spark core optimization

Why Migrate to Spark

- More complex big data processing needs
- Streaming, Graph computation, Machine Learning use cases
- Extreme performance optimization need

What We Do

- ~90% batch workload auto migration
- Tool sets to enable manual migration

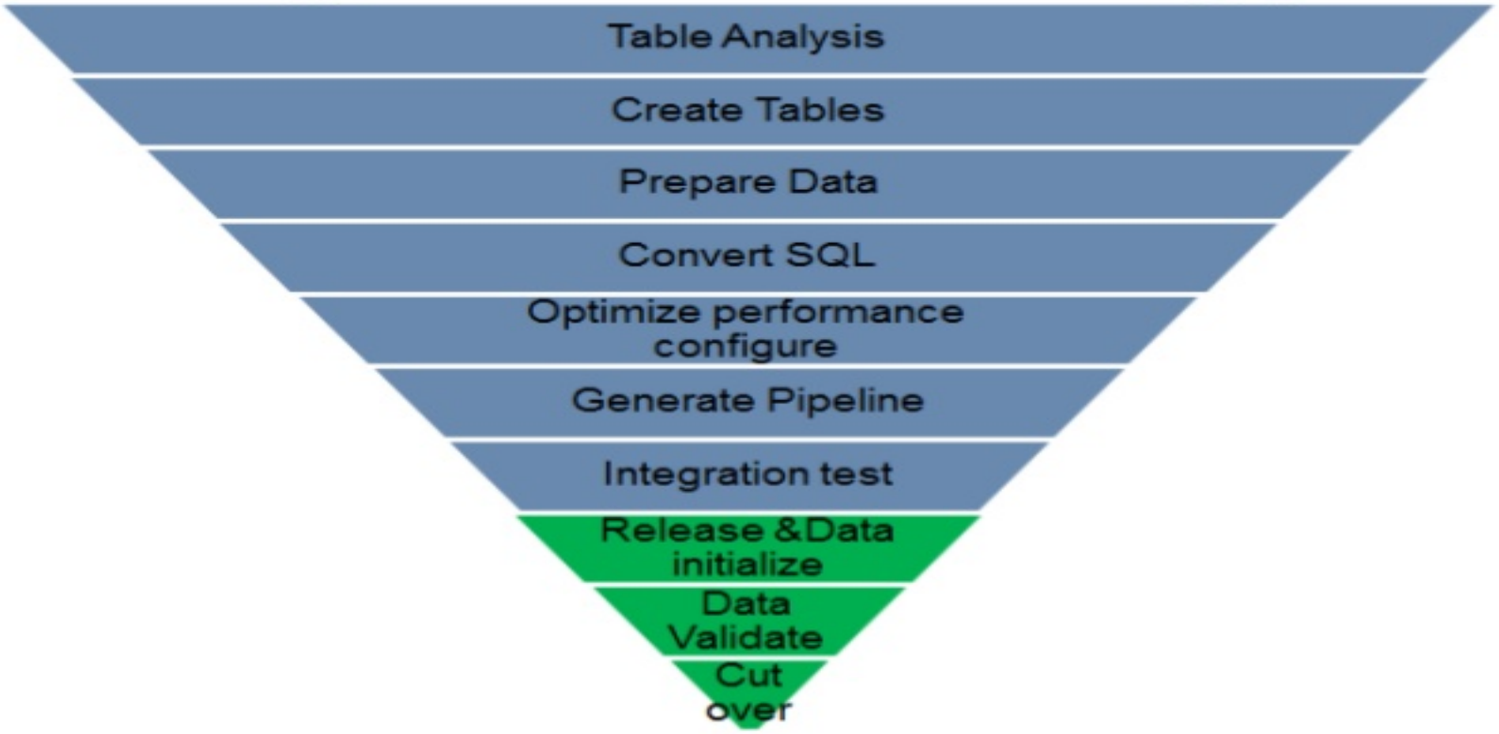
Agenda

- Auto Migration Scope
- Auto Migration Strategy
- Auto Migration Components
- Key Components
- Tool Sets
- Major Challenges
- Be part of community

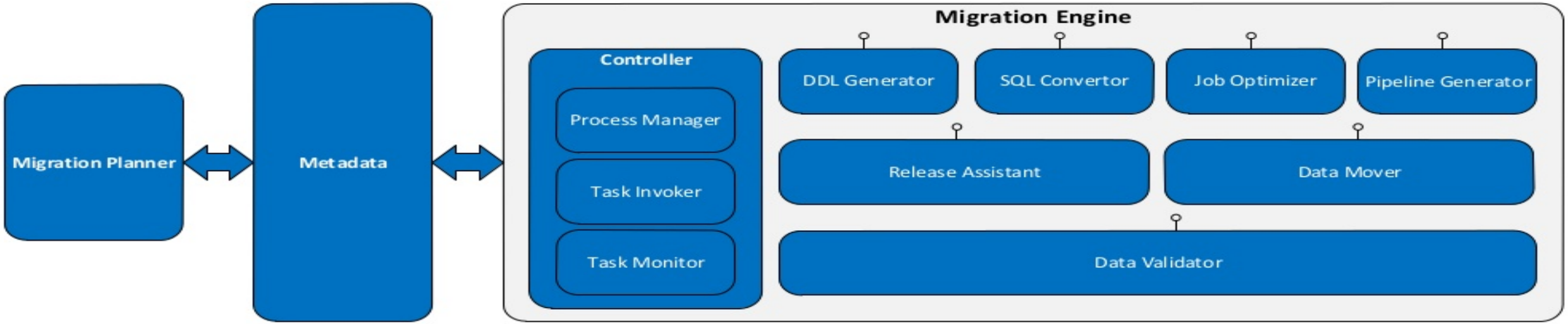
Auto Migration Scope

- ~5K Target tables
- ~20K intermediate/working tables
- ~22PB target tables
- ~40PB relational data processing every day
- ~ 1 year timeline

Auto Migration Strategy



Auto Migration Framework



Auto Migration Components

Migration Planner

- Analyze and identify auto migration candidates
- Determine the order of table migration

Metadata

- Define and collect metadata to enable the auto migration engine
- Include table profile, data linage, job linage, SQL file profile, pipeline profile

Auto Migration Components

Controller

- Manage the end to end migration process
- Include sub components like process manager, task invoker, task monitor

DDL Generator

- A data modeler to generate DDL on Spark for target table, working tables and views
- Also include setting the table format, bucket and partition

Auto Migration Components

SQL Convertor

- Split original SQL files into table transform + merge steps
- Parsing original ADBMS SQL into abstract syntax tree and assemble into Spark SQL
- Special rules to deal with SQL dialect and UDFs

Auto Migration Components

Job Optimizer

- Pre generate Spark job execution configurations based on table size and Spark cluster scale (typically spark.sql.shuffle.partions)
- Leverage Spark Adaptive Execution to optimize the execution plan online

Auto Migration Components

Pipeline Generator
– Generate workflow to set spark sql files execution steps and schedule

Release Assistant
- Push code to production environment and github repo, and table creation ..

Data Mover
- Move data across platforms, for snapshot data preparation on DEV and historical data initialize on PROD

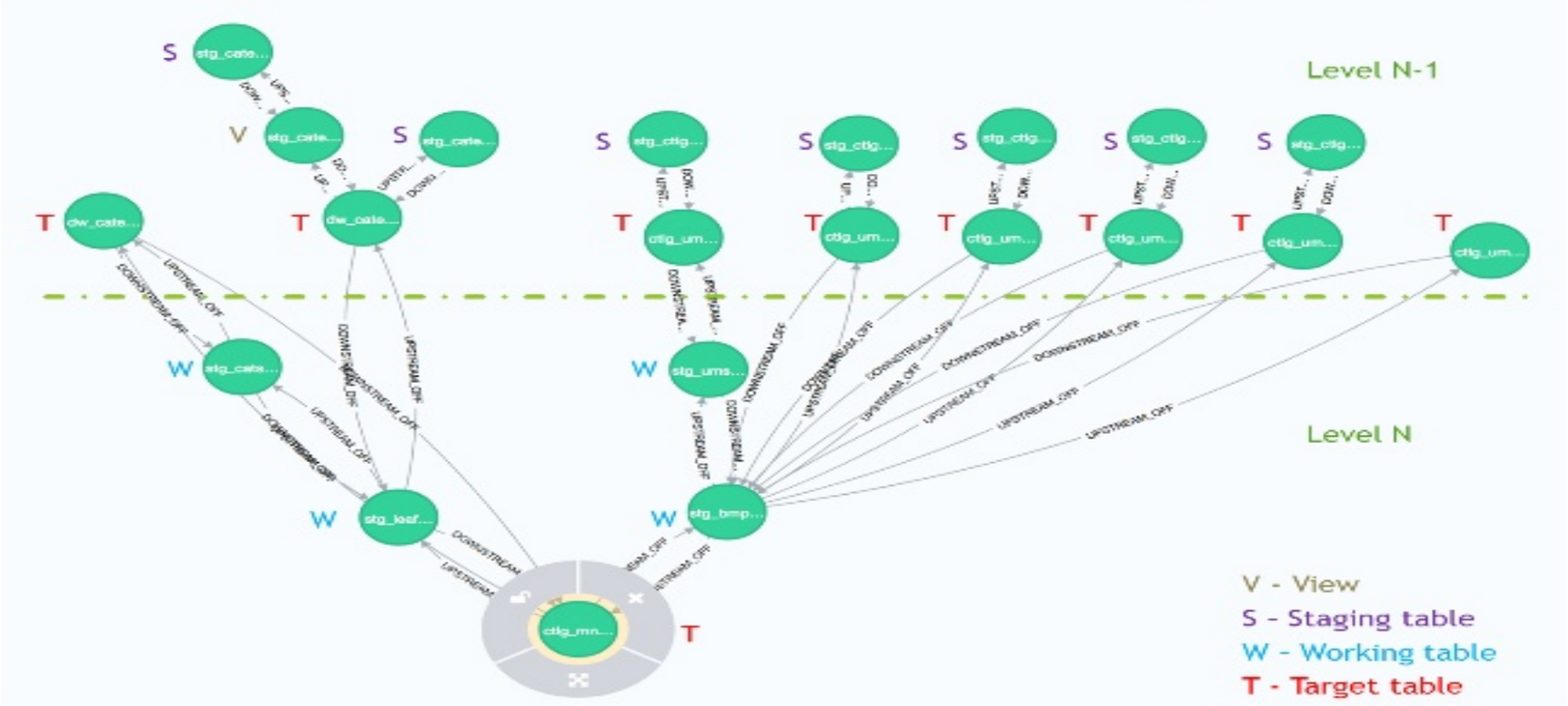
Data Validator
- Cross platform data checksum on both DEV and PROD

Key Components

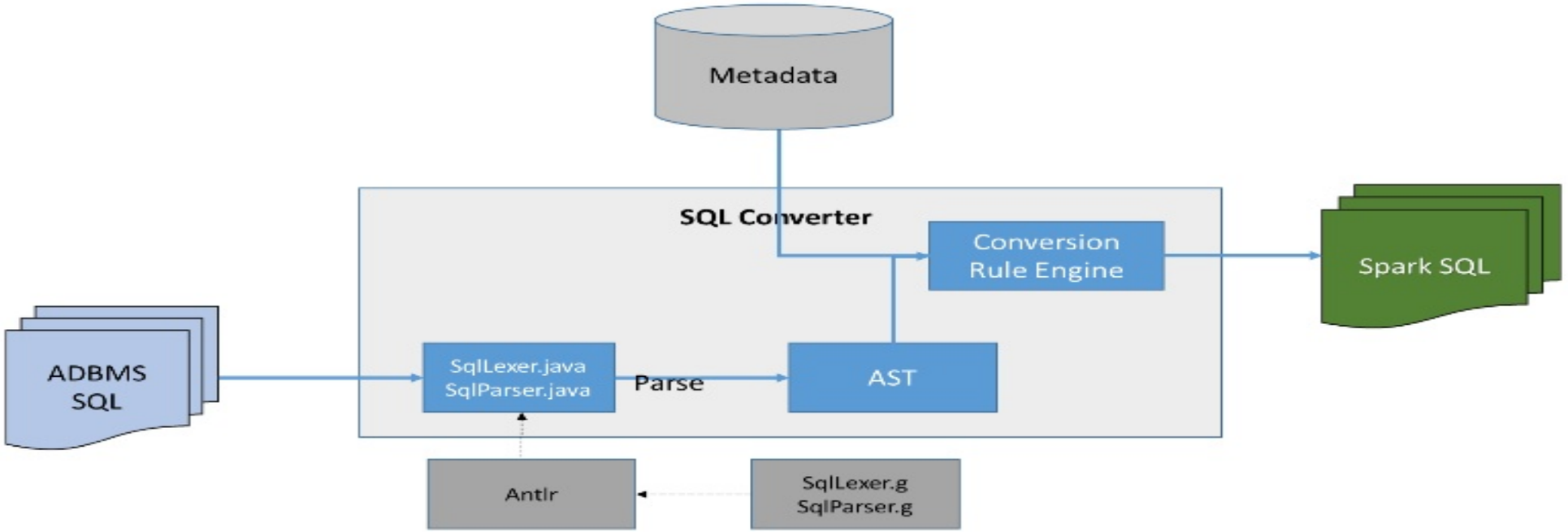
- **Metadata**
- **SQL Converter**

Metadata - Overview

Metadata – Data Linage



SQL Converter - Overview



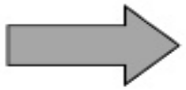
SQL Converter – Conversion Rules

- Split original SQL files into table transformation and final table merge
- Identify ACID steps (merge update/delete/insert into one insert-overwrite step)
- Multiple update/delete cases – store middle step result into temp view and do final single merge
- Special handling for cases like case sensitive, date/timestamp calculations, column name alias …
- Adaptive for Spark known issues
- Internal function & UDF translation

SQL Convertor – Sample

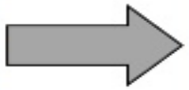
ADBMS SQL

```
UPDATE TGT
FROM
   DB.TARGET_TABLE TGT,
   DB.WORKING_TABLE SRC
SET
   COLUMN1 = SRC.COLUMN1
...
WHERE
   SRC.PK = TGT.PK;

INSERT INTO DB.TARGET_TABLE
(
PK,
COLUMN1,
...
)
SELECT
    PK,
    COLUMN1,
...
FROM
    DB.WORKING_TABLE SRC
LEFT JOIN
   DB.TARGET_TABLE TGT TGT
ON
      SRC.PK = TGT.PK;

WHERE
   TGT.PK IS NULL;
```



AST

```
<TOK_ROOT>
 <TOK_UPDATE_STATEMENT>
  <TOK_UPDATE_QUERY>
   <TOK_FROM_WRAP>
    <TOK_FROM>
     <TOK_TABREF>
      <TOK_TABNAME>
       <TOK_DB_TABNAME>
        DB
        TABLE
       </TOK_DB_TABNAME>
      </TOK_TABNAME>
      alias
     </TOK_TABREF>
    </TOK_FROM>
                                    ...
   </TOK_FROM_WRAP>
   ...
  </TOK_UPDATE_QUERY>
 </TOK_UPDATE_STATEMENT>
 <TOK_INSERT_STATEMENT>
  <TOK_INSERT_QUERY>
                ...
  </TOK_INSERT_QUERY>
 </TOK_INSERT_STATEMENT>
</TOK_ROOT>
```

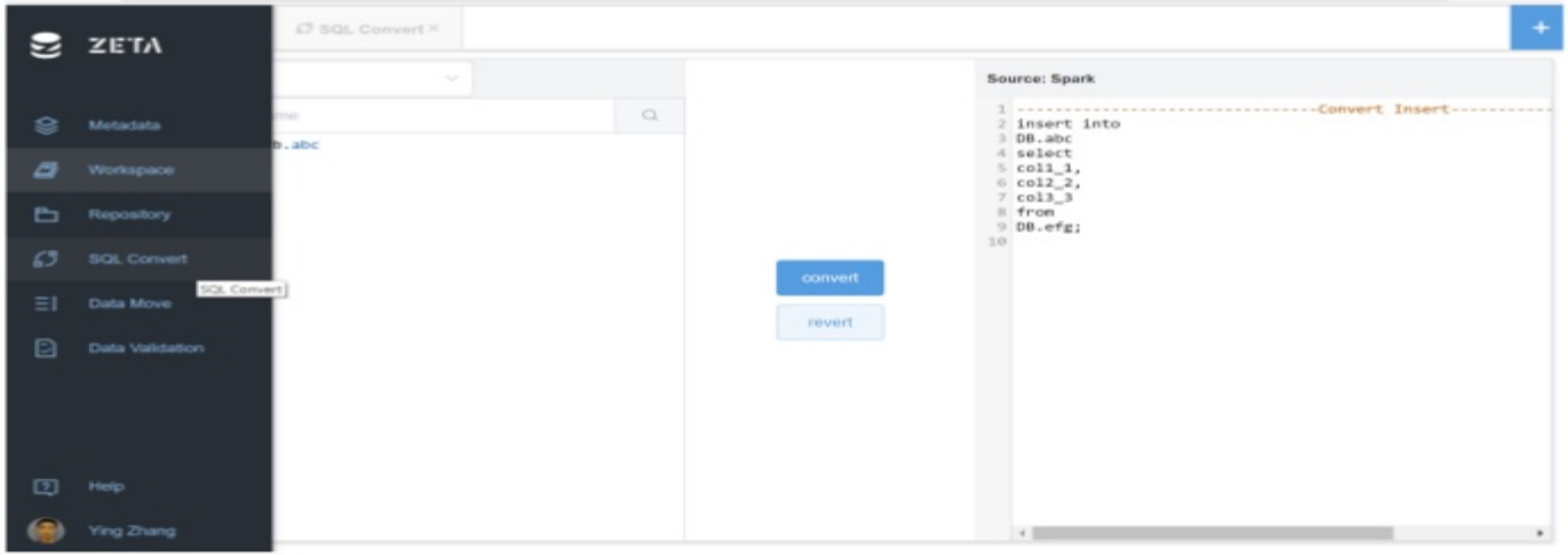


Spark SQL

```
-----------------Convert Update-----------------
insert overwrite table DB.TARGET_TABLE
partition(dt=${SNPSHT_DATE})
select
tgt.PK as PK
,case when src.PK is null then tgt.COLUMN1
else src.COLUMN1 end as COLUMN1,
...
from
DB.TARGET_TABLE as tgt
left join
DB.WORKING_TABLE as src
on
 (src.PK=tgt.PK)
-----------------Convert Insert-----------------
union all
select
src.PK,
src.COLUMN1
...
from
DB.WORKING_TABLE as src
left join
DB.TARGET_TABLE as tgt
on
 (src.PK=tgt.PK)
where
(tgt.PK) is null;
```

Tool Sets

- DDL Generator
- SQL Converter
- SQL Optimizer
- Pipeline Generator
- Release Assistant
- Data Mover
- Data Validator
- + Dev Suite



Major Challenges

- **Metadata Definition & Collection**
  - *You do not know what you do not know*
- **Data Validation**
  - *Upstream data quality issues*
  - *SQL behavior or data format difference on Spark*
- **No SQL Jobs**
  - *Cannot cover logic in shell scripts or command lines in pipeline*

Be part of community

~ 50 issues reported to community during migration

Case-insensitive field resolution

- SPARK-25132 Case-insensitive field resolution when reading from Parquet
- SPARK-25175 Field resolution should fail if there's ambiguity for ORC native reader
- SPARK-25207 Case-insensitive field resolution for filter pushdown when reading Parquet

Parquet filter pushdown

- SPARK-23727 Support DATE predict push down in parquet
- SPARK-24716 Refactor ParquetFilters
- SPARK-24706 Support ByteType and ShortType pushdown to parquet
- SPARK-24549 Support DecimalType push down to the parquet data sources
- SPARK-24718 Timestamp support pushdown to parquet data source
- SPARK-24638 StringStartsWith support push down
- SPARK-17091 Convert IN predicate to equivalent Parquet filter

UDF Improvement

- SPARK-23900 format_number udf should take user specifed format as argument
- SPARK-23903Add support for date extract
- SPARK-23905 Add UDF weekday

Bugs

- SPARK-24076 very bad performance when shuffle.partition = 8192
- SPARK-24556 ReusedExchange should rewrite output partitioning also when child's partitioning is RangePartitioning
- SPARK-25084 "distribute by" on multiple columns may lead to codegen issue
- SPARK-25368 Incorrect constraint inference returns wrong result

Q & A

Thank You!