DATA+AI
SUMMIT 2022

Advanced Migrations: From Hive to Spark SQL

How Pinterest migrated 10,000+ Hive jobs to Spark SQL

ORGANIZED BY databricks

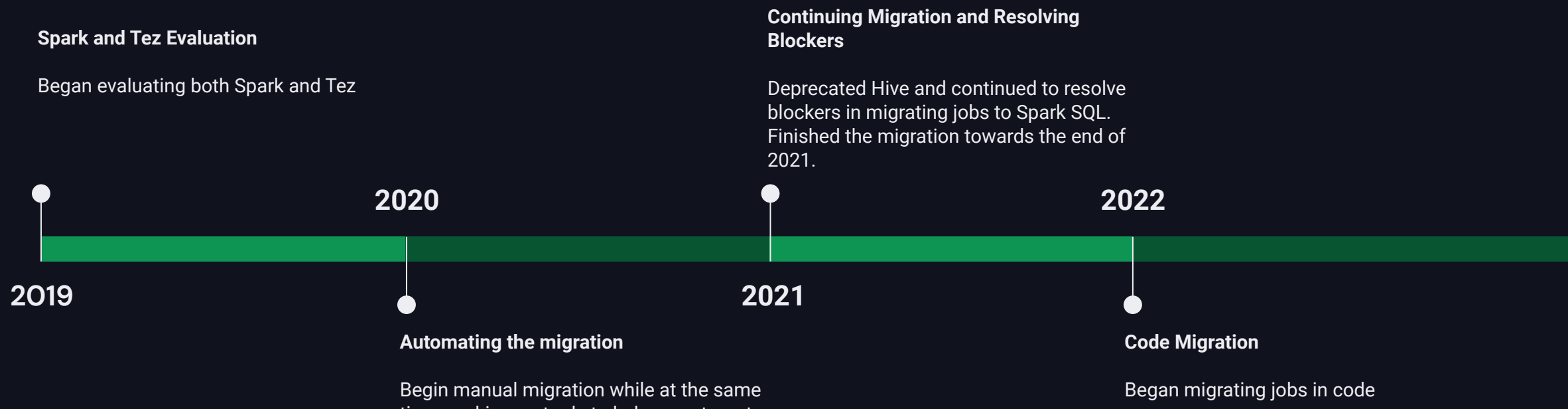
About Me

- Software Engineer at Pinterest working on the Query Platform Team
- Focused on improving Spark SQL usability and performance for engineers at Pinterest.

Agenda

- Migration Specifics
- Migration Challenges
- Automated Migration Service
- Code Migration
- Changes required to Spark
- Results of the migration

Migration Timeline



Migration Specifics

- Hive 1.2.1
- Spark 2.4
- 10,000+ Queries

Migration Challenges

Migration Challenges

Testing Queries

- No impact to production
- Syntax compatibility with Spark 2.4
- Validation

Migration Challenges

Making Queries Safe

- Tailor
  - In house query manipulation library
  - Based on sqlparse

Migration Challenges

Making Queries Safe

```
INSERT OVERWRITE TABLE mydb.num_user_events PARTITION (dt='2022-04-20')
SELECT id, count(*)
FROM mydb.user_events
GROUP BY 1
```

```
CREATE TABLE ams_sparksql.mydb__num_user_events_42069 LIKE mydb.num_user_events;
INSERT OVERWRITE TABLE ams_sparksql.mydb__num_user_events_42069 PARTITION (dt='2022-04-20')
SELECT id, count(*)
FROM mydb.user_events
GROUP BY 1
```

Migration Challenges

Syntax Compatibility

```
CREATE TEMPORARY TABLE temp_table …
```

```
CREATE TEMPORARY VIEW temp_table …
```

Migration Challenges

Validation - Output Validation

- Checksum UDF
  - Row order agnostic checksum function

- Computed and compared checksum of both Hive and Spark SQL outputs.

Migration Challenges

Validation - File Size and File Count

- Spark output file size <= Hive output file size + 1 gb
  - Compression Settings
  - Sorting (sorted datasets lead to better compression)

- Spark output file count <= Hive file output count + 1
  - Repartition is added to query

Migration Challenges

Validation - Runtime and Resource Usage

- Spark runtime > Hive runtime
  - Increase number of executors
  - Increase size of executors (Memory / Cores)

- Cost of Spark Query > Cost of Hive Query
  - Reduce number of executors
  - Reduce size of executors

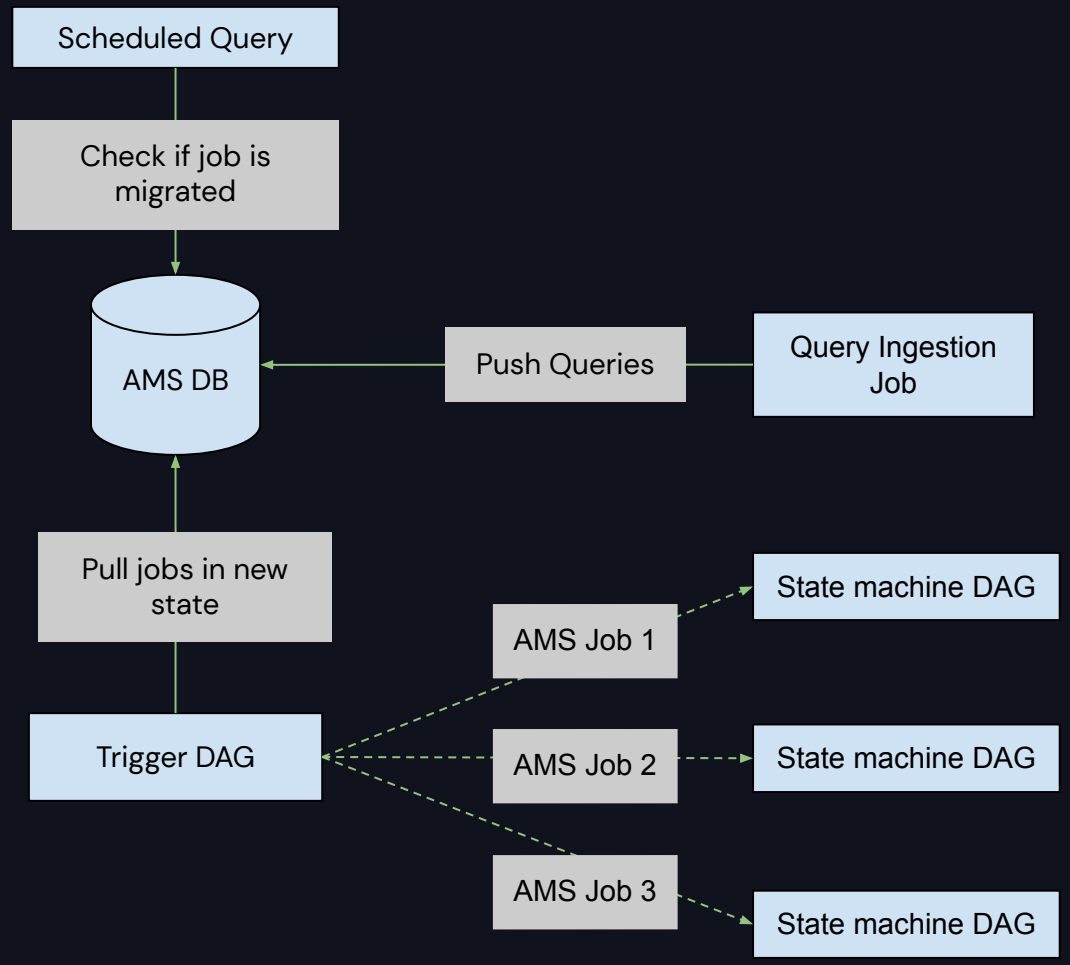
Auto Migration Service (AMS)

Architecture

- Help facilitate mass testing of Hive Jobs on Spark SQL
- State machine Pattern
- Built using Apache Airflow
- UI containing report on test results and migration status

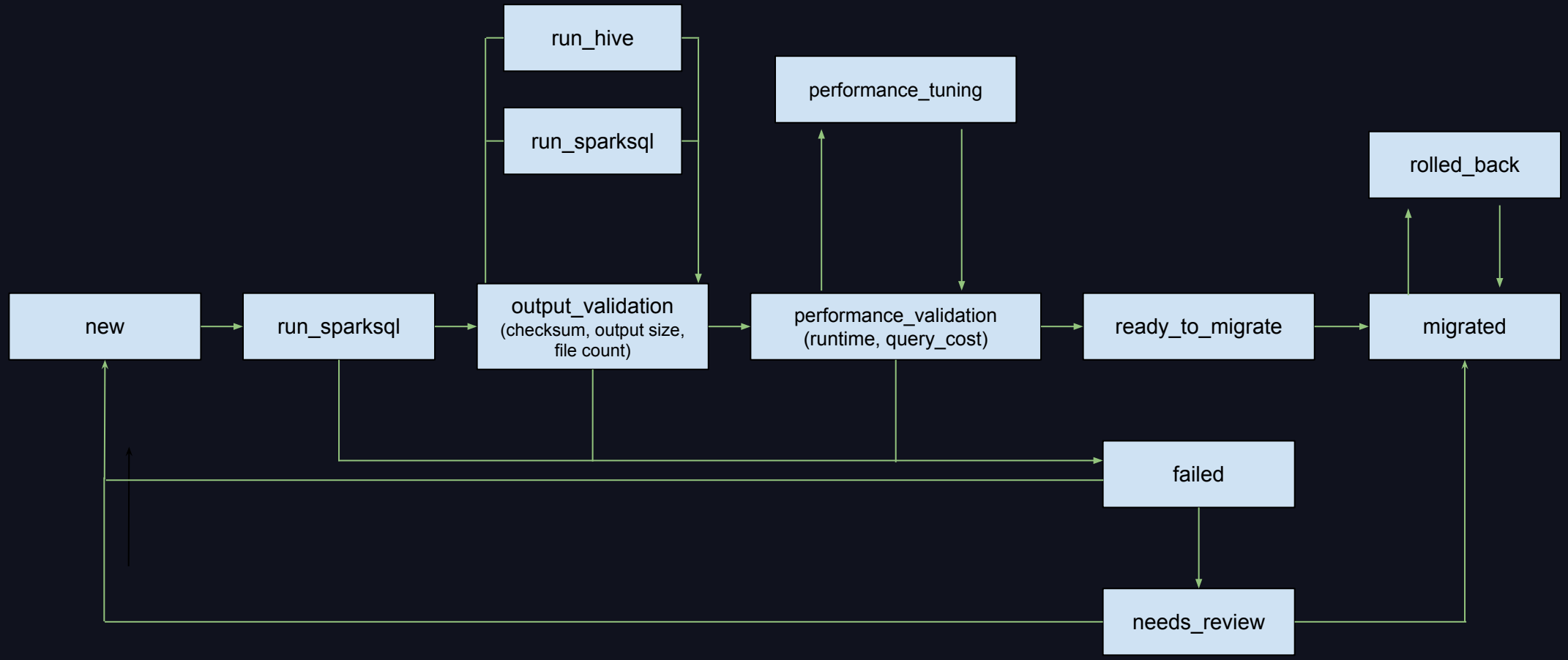
Auto Migration Service

Architecture - Overview



Auto Migration Service

Architecture - State Machine



Auto Migration Service (AMS)

Cluster Resources

- AMS test jobs were scheduled on an adhoc clusters as to not impact production
- An AMS specific queue is used
- Only 30 jobs were allowed to run simultaneously.

Migration Challenges

Failed Jobs

1. Identify error that is causing the most number failures
   - All errors were logged into a hive table for easy analysis
2. Fix error
   - (optional) Implement failure handler
3. Re-run jobs through state machine.
4. Repeat

Migration Challenges

DDL Queries

- AMS is only really useful for DML queries.
- It did not make sense for DDL queries to run through AMS
  - Many DDL queries are just metastore operations
  - DDL queries do not consume a lot of resource and the runtime is negligible
- Instead we made sure DDL statements behaved the same as Hive
  - Syntax compatibility
- Automatically migrated if syntax was compatible.
  - Took us two weeks to migrate all DDL queries from Hive to Spark SQL

Migration Challenges

Stopping the Inflow of Hive Jobs

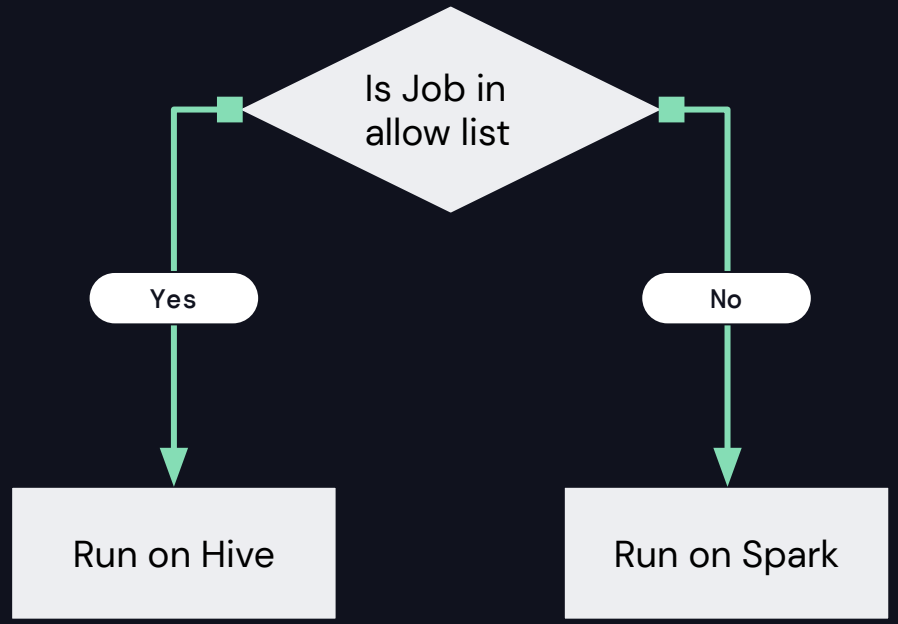
New jobs on Spark

Ad hoc Jobs

- Disabled ability to run Hive for most users

Scheduled Jobs

- Allow list that contains a workflow, job pair allowed to run on Hive
- Check is made during Hive execution



Migration Challenges

UDF Support

- Most UDFs work fine.
- For UDFs that didn’t work manual changes were required.
  - UDFs needed to work on both Hive and Spark SQL until migration was complete.
- Example of changes
  - Making UDFs thread safe
  - MapredContext

Code Migration

Code Migration

Why is code migration important?

- Source of Truth
- New engineers working on code may get confused to see a Hive job running as spark sql.
- Changes to queries overtime may lead to failures that the translator did not anticipate
- Maintenance overhead of AMS.

Code Migration

How scheduled queries are defined at Pinterest

Before

```
import HiveJob

class TestHiveJob(HiveJob):

    _QUERY_TEMPLATE = """

        CREATE TEMP TABLE …;

        SELECT * FROM …
    """
```

After

```
import SparkSQLJob

class TestHiveJob(SparkSQLJob):

    _QUERY_TEMPLATE = """

        CREATE TEMP VIEW …;

        SELECT

        /*+ REPARTITION(10) */ *

       FROM …
    """
```

Code Migration

How to make updates to code? - Python AST

- Python AST
  - Python AST is lossy
  - ASTs are good for tools like compilers and type checkers where the semantics of code is important, but the exact syntax isn’t.
  - Tooling for generating code is lacking.

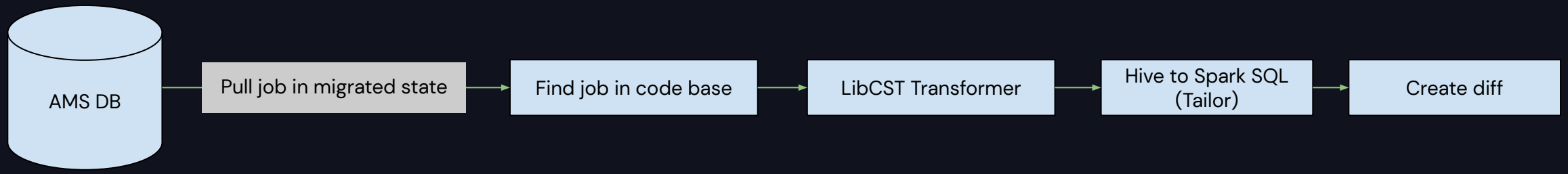
Code Migration

How to make updates to code? - LibCST

- LibCST
  - Parsing library developed by Instagram
  - Exact Representation of code
  - Allows for easy traversal and modification of code
  - Allows you to go to from tree to code and vice versa

Code Migration

Architecture



Changes Required to Spark

Changes Required to Spark

Thrift Table Support

Thrift Changes

- Pinterest has a number of tables backed by a thrift schema
  - Thrift files hold the schema definition
  - Single source of truth.
- Uses a custom serde
- Change was made to extract columns from the thrift serde object inspector.

Thrift Schema (source of truth)

```
struct Name {
  1: required string first_name,
  2: optional string last_name
}
```

Changes Required to Spark

S3 Committer

- AWS S3 Renames are slow
- Updated Spark to use direct committer
  - Based on https://github.com/rdblue/s3committer
  - On task commit a multi-part upload is started
  - On job commit completes each multi-part upload started by tasks
- Updated Hive insert code path
  - S3DirectoryOutputCommitter for unpartitioned data
  - S3PartitionedOutputCommitter for partitioned data

Changes Required to Spark

`CombineFileInputFormat`

- Problem with too many small files
  - Scanning files takes a long time
  - Creates many tasks which can slow down jobs
  - Driver needs to keep track of every tasks so many tasks can lead to driver OOMs
- Why not use coalesce?
  - Does not respect mapred.max.split.size and mapred.min.split.size
  - Coalesce doesn't balance data size across partitions so you can get skewed partitions
- Hive has a CombineFileInputFormat that can combine many small files
  - Respects mapred.max.split.size and mapred.min.split.size

Changes Required to Spark

Decompression Split

- Pinterest has some input formats that are not splittable
  - Changing source of data to be splittable was not an option
- Split large compressed files into chunks
  - Threshold determined by decompress.split.minsize
  - Chunk size determined by mapred.min.split.size
- Record reader forwards to the split start and reads until split end

Changes Required to Spark

MSCK REPAIR TABLE

- Hive has the ability to ignore invalid partitions
  - i.e part_name=""
- Spark would re-add partitions that already exist on the metastore
  - Overload HMS for tables with large number of partitions.
  - Exacerbated if many tables were repairing during the same time.

Results of the Migration

- Large cost savings
- Runtime weighted speed up of 70%
- Average of 38% reduction in cpu usage
- Average of 61% Increase in memory usage

DATA+AI
SUMMIT 2022

Thank you