Increasing Atlas performance using a nested OMOP model with Apache Spark

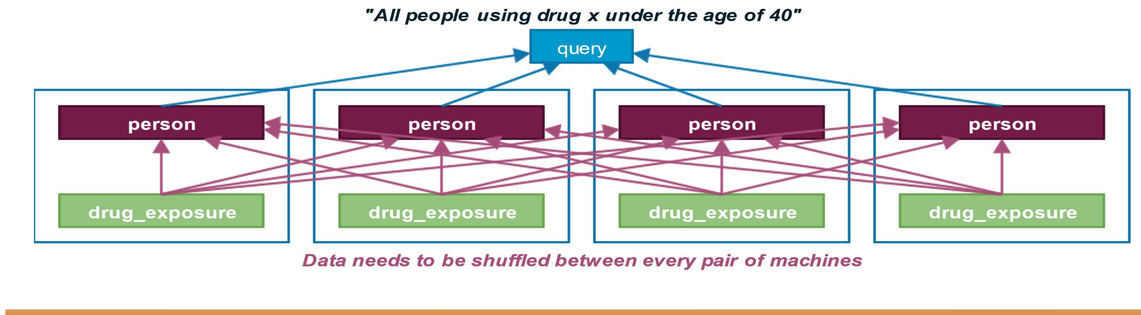
Michael Davies MSc, Alex Tawse BSc, Joana Cruz MSc, Katie Mannering
IQVIA, Brighton, UK

Introduction

The standard physical storage model for OMOP data is based on a set of relational tables. When using this model for large volumes of data, the query performance can be poor, even on Hadoop based systems. Analysis shows that most of the time taken during cohort specification is spent **scanning the raw data**. To minimize scans and shuffles to the data we propose a physical model that **co-locates person related data**, using a nested data structure, and **Apache Spark** as a distributed processing engine for the analysis generation.

Methodology

The existing Impala cohort definition queries result in multiple scans over the same data. This is partly due to how the SQL CTE used for primary events is evaluated, but is also due to separate scans for additional cohort inclusion criteria. As the data is not co-located, the results of each scan need to be shuffled between servers prior to joining, which can be an expensive operation. This is illustrated in the figure bellow:



In order **to improve system performance** we need to…

Ensure the data is **scanned only once** | **Minimize shuffling** by co-locating person data

Move away from the evaluation of cohorts in **pure SQL**, where cohort queries are inevitably expressed as a series of **joins**

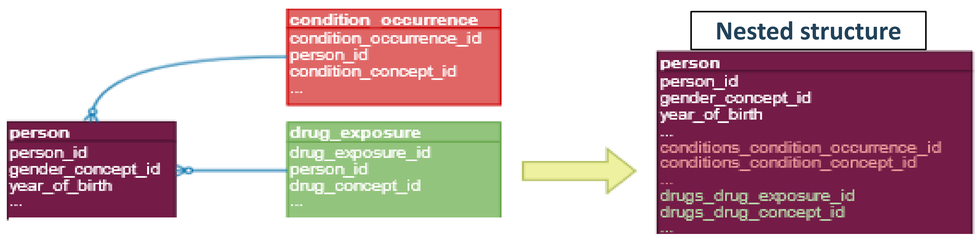
Use of a **big data** processing framework, **Apache Spark**, to perform cohort evaluation which provides the fine-grain control to…

Ensure that the two conditions of read-once and co-location are met

Do cohort evaluation using a programming language such as Java or Scala, which helps modularize and test the implementation

Nested data structure

The nested structure is composed of **arrays of person events** ordered by start date and identifier nested within the person table. An extract below shows part of the schema:

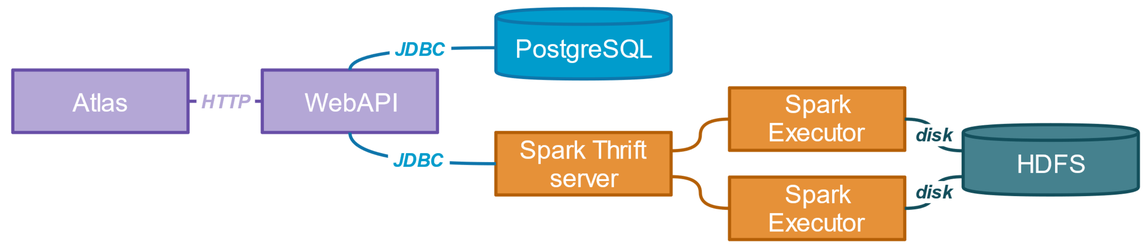


Integration with Atlas

A simple integration with Atlas has been achieved by i**ntercepting execution requests in the WebAPI Tasklets** and redirecting those aimed at a nested data source to Spark. With this approach we are supporting the following features from Atlas 2.6: **cohort generation, cohort characterizations, cohort pathways and profiles**.

To ensure the reliability of our solution, tests have been written to compare the obtained results between the existing and Spark based functionality. WebAPI tables that are not needed in Spark are being stored in a Postgres instance.

The setup of the Atlas integration is shown in the following figure:



Results

To evaluate performance of our solution we ran a selection of Atlas features on Spark and Impala and compared the execution time. The test cluster consisted of 3 physical machines with 24 cores (48 threads) and 256GB memory each. We evaluated the performance of the following Atlas features on Hospital dataset which has 88 million persons and 5 billion events:

Feature	*Scenario*	*Database*	*Time*
Cohort generation	Warfarin cohort	Spark	3s
		Impala	1min 30s
Cohort characterization	Warfarin cohort Criteria: Age > 70	Spark	11s
		Impala	3min 3s
Pathway analysis	Type 2 Diabetes Mellitus Medications	Spark	1min 32s
		Impala	3min 39s
Profiles	Person 99331892 Number of events: 473	Spark	4s
		Impala	10s

Conclusion

The approach described in this poster has proven to bring several benefits, most notably an improved and predictable query performance on Hadoop systems. Results showed that:

- cohort execution runs up to **25x faster** than on Impala,
- provides **improved QA** opportunities, which in the long term will translate in much more reliable code,
- enables additional **complex cohort specification** that cannot (easily) be expressed in SQL.

Spark backend is able to provide views on top of the nested tables to simulate the original CDM tables. This minimizes the development work required to operate Atlas (for example) on top of Spark, while still delivering performance benefits. To maximize performance, Atlas would need support for the nested data model.

Contact: michael.davies@iqvia.com, atawse@ iqvia.com, jcruz@iqvia.com, kmannering@iqvia.com