it-novum
Open Business Solutions. Delivered.

Research Paper

Big Data: Using Hive on Spark with Jedox

Authors:
Alexander Keidel, Michael Deuchert und Phillip Musch

Tested with:
Jedox: 6.0, 6.0 SR1
Cloudera CDH: 5.4.8

Content

1. Big Data data sets with Hive on Spark and Jedox

With the release of Hive 1.1 in January 2015, Apache has now combined Spark, an open-source, cluster-computing project for processing large amounts of data, with Hive.

Unlike MapReduce (via Hadoop), Spark tries to perform many data processing operations in memory while keeping access to the HDFS as low as possible. As a result, for some applications it can perform up to 100 times faster than Map-Reduce. This applies in particular to applications that use many Reduce steps - such as when translating complex queries - or in business intelligence, which also routinely uses a lot of Reduce steps.

Until now, it was not possible to use Spark in the Jedox business intelligence platform and thus realise these performance gains. This was why we began our testing and investigations: We would like to show how Spark can be used in conjunction with Hive and, with some restrictions, with Jedox without having to make changes either to the queries or Jedox.

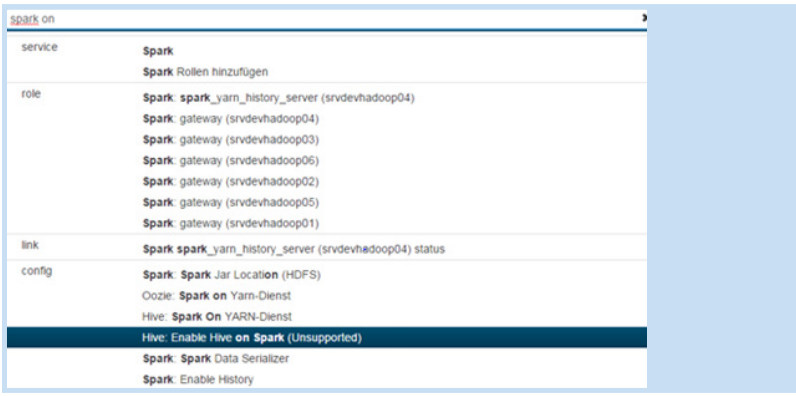
2. Configuring Hive on Spark: The Cloudera Cluster Manager

The most convenient way to provide Hive on Spark to users is via the Cloudera software stack, which is also intensively used at it-novum. This can be easily configured via the Cloudera Cluster Manager interface. Please note, however, that Hive on Spark is only available as of version 5.4.x and has not yet been released by Cloudera for live environments. This means that Cloudera does not yet provide support for using Hive on Spark.

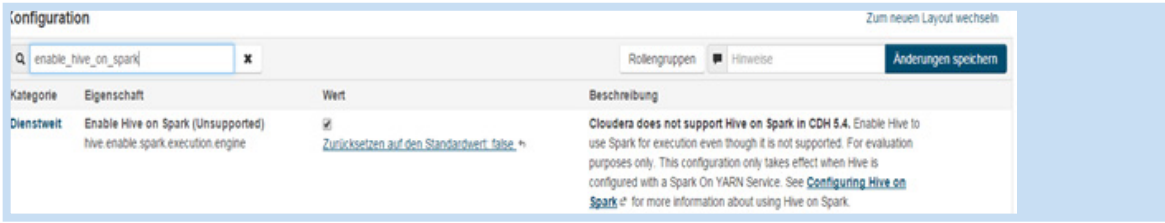
For our tests, we used the latest Cloudera version (CDH 5.4.8). To configure Cloudera correctly, the following steps must be carried out:

1) Click on the Hive Service:

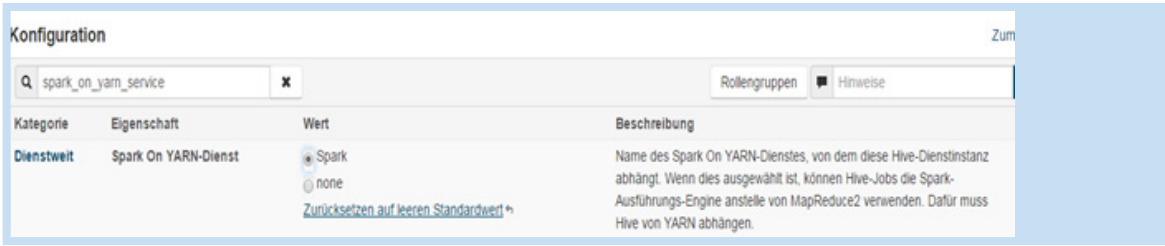
2) In the Cloudera Manager search window, double click the configuration setting „Hive: Enable Hive on Spark“:



3) By selecting the checkbox and then clicking on „Save changes“, Cloudera will create the corresponding configuration settings:

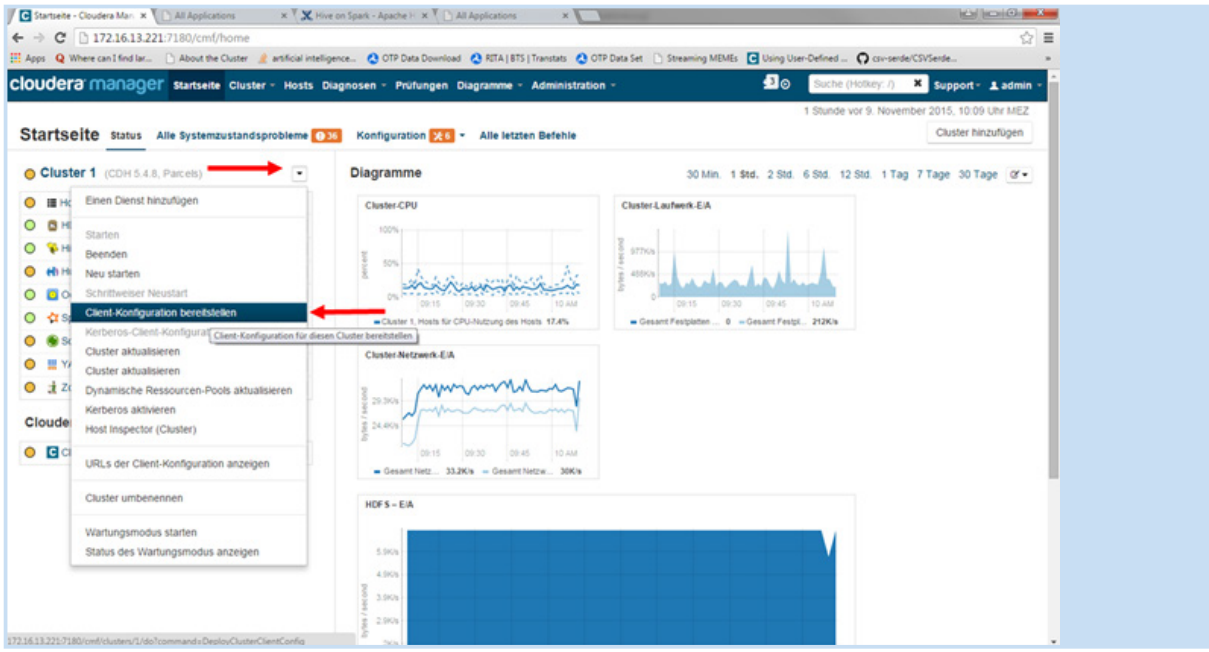


4) It should be noted that in working with Cloudera, Hive on Spark can only be used in conjunction with YARN. This presents an initial restriction, because it results in Spark having an additional dependence on Hadoop. However, it carries with it the advantage that both Spark and Hadoop MapReduce jobs can be managed via a standardised, consistent interface.

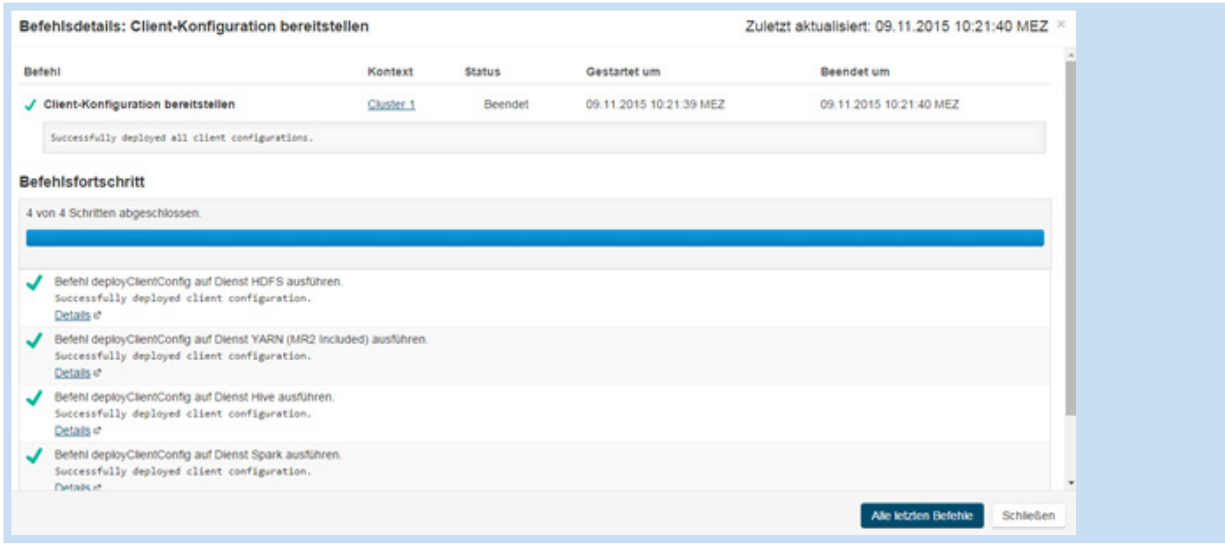


To configure Spark on YARN in Cloudera, you must first search for the spark_on_yarn service in Cloudera Cluster Manager. Once this has been done, click on Spark and then „Save changes“, so that Cloudera creates the appropriate configuration settings.

5) Finally, the configuration settings generated in steps 1 to 4 must be pushed to the individual cluster instances. This is achieved by clicking on „Deploy client configuration“ in the cluster context menu:



6) Once the client configuration has been updated, Hive on Spark can be used with Cloudera:



3. Testing Hive on Spark

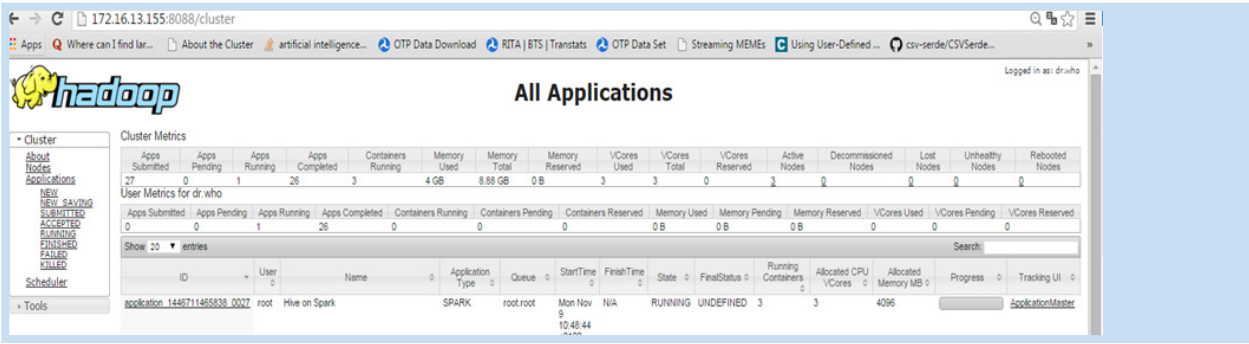
The simplest way to test the configuration created under Section 1 is via the Hive shell. To do this, the Hive Execution Engine has to be changed first. This is set up by default on MapReduce via Hadoop. The command „set hive.execution.engine=spark“ in the shell changes the hive. execution engine for the respective session to Spark:

hive>set hive.execution.engine=spark;

For example, a query for displaying the number of lines:

hive> select count (*) as num_row from mytable;

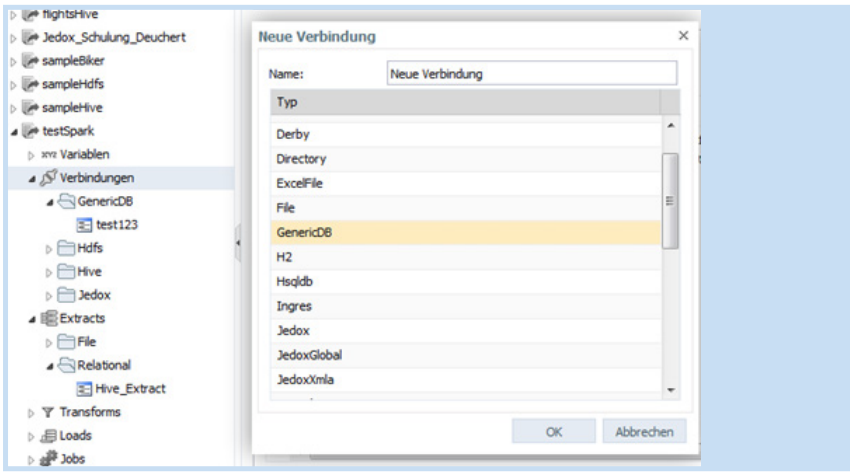
will create a Spark job that is then displayed in the YARN job tracker.



Note: Queries such as Select * From my_table do not generate a MapReduce or SparkJob in Hive because technically they are only reading the HDFS file that is stored for the table.

4. Setting up Hive on Spark in Jedox

To use Hive on Spark with Jedox, we will also need the Jedox Hadoop Connector. In Jedox Version 6.0 this must be installed as an additional package. Once this has been done, a standard Hive connection (via MapReduce) can be set up using the „Hive“ connection option. First, we need to establish a connection to Hive in Jedox and then set the Hive execution engine to be Spark. Currently, this can only be done using a few workarounds in Jedox.
To do this, we first need to create a new connection using the generic database connection:



…and then enter the following values:

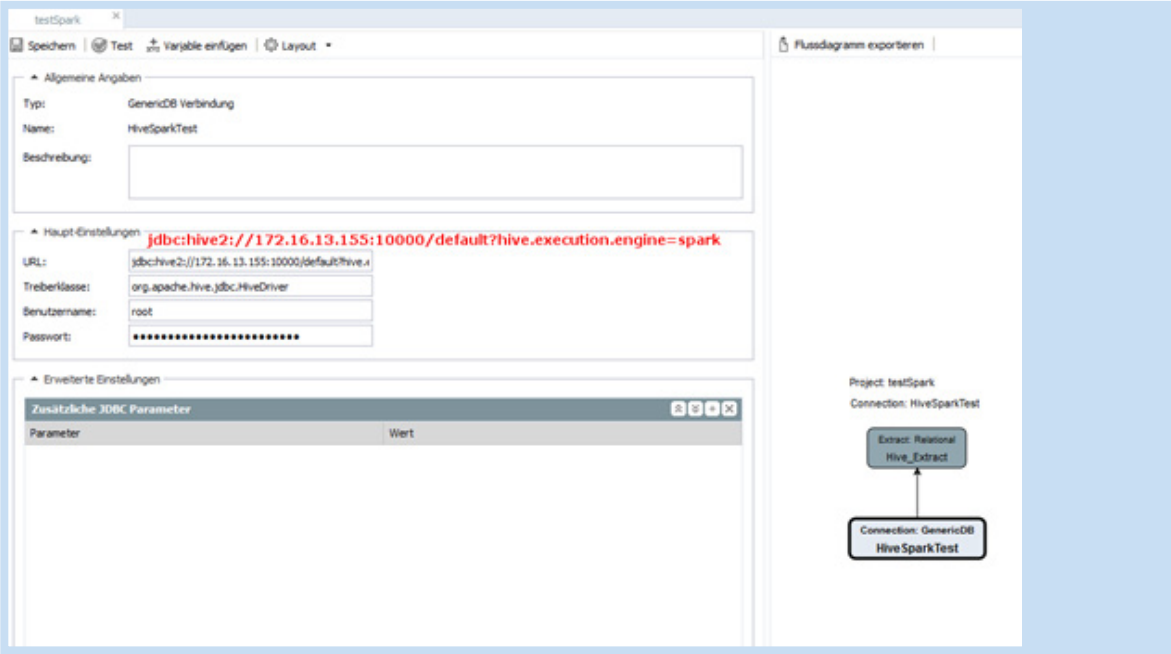
- URL
- Driver class
- Username
- Password

The URL is a JDBC string and should be set as follows:

Jdbc:hive2://IP_HIVE2_SERVER:HIVE_TCP_THRIFT_PORT/HIVE_DB

To be able to use Hive on Spark, a GET parameter has to be added that will be executed automatically when the Hive server is connected. The generic JDBC string for using Hive on Spark is:

Jdbc:hive2://IP_HIVE2SERVER:HIVE_TCP_THRIFT_PORT/HIVE_DB?hive.execution.engine=spark



The driver class we use is the official Hive JDBC driver:

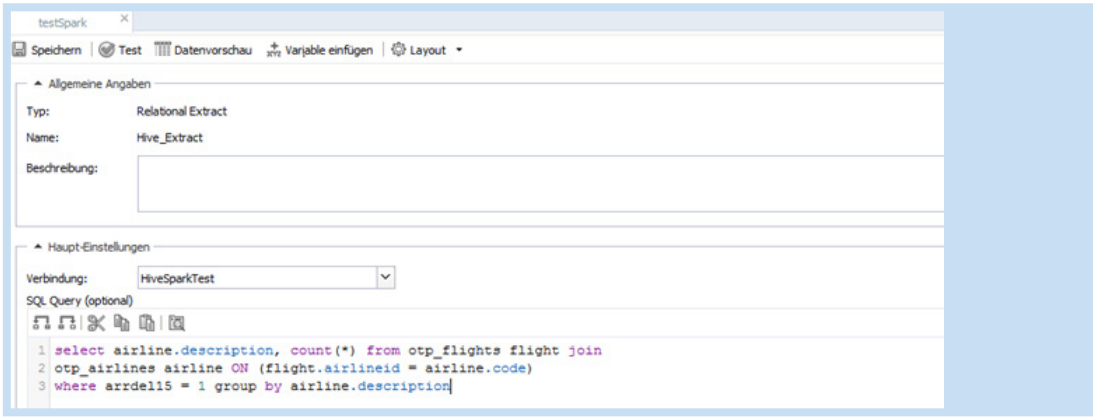
org.apache.hive.jdbc.HiveDriver.

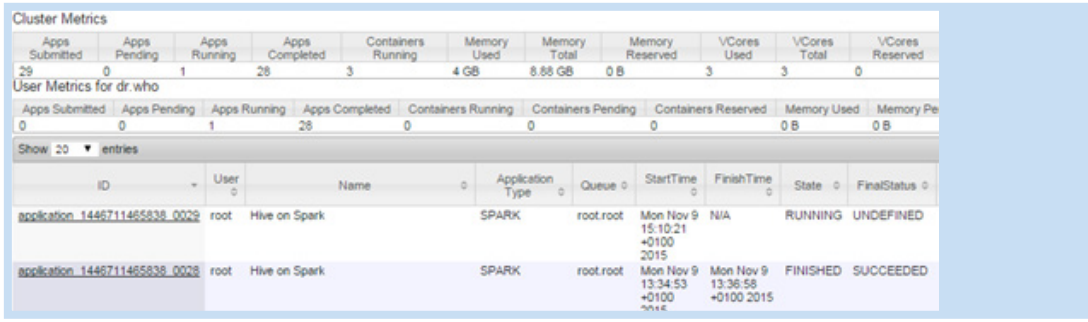
The connection can now be tested by clicking on the Test button.

Note: In Jedox 6.0 SR2, this configuration can be created directly via the GUI using the Hive connection type, which greatly simplifies the procedure. Detailed documentation for Hive configuration parameters and transferring these using a JDBC URL can be found under Hive Configuration Properties and Hive2 Connection URL Format.

5. Hive on Spark with Jedox

Using the connection described in the previous section, queries executed against the Hive table are no longer executed as MapReduce jobs via Hadoop, they‘re now run via Spark. So, for example, the query now creates a Spark job that, as described in Section 1, can be viewed in the YARN cluster manager.



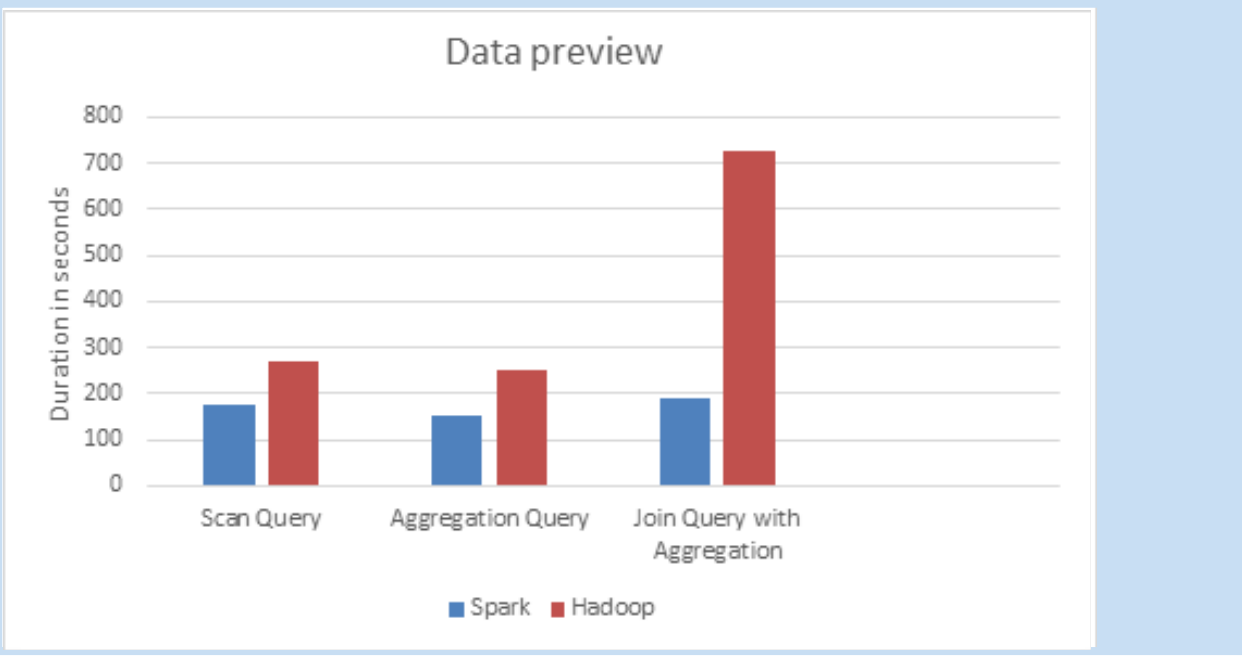


Once the Spark job has completed, the results are transferred to Jedox. These can now be further processed as usual.

6. Measuring performances

In order to measure the performance of Hive on Spark in conjunction with Jedox, we carried out measurements using three different queries. As test data, we used flight data from the U.S. from January 2014 to August 2015, a freely available data record with approximately 8.2 million lines that is available at http://www.transtats.bts.gov.

The first measurement, a simple scan query, was used to filter the data set according to the March 2014 data. The second measurement consisted of an aggregation query and was intended to output the number of rows in the database. Finally, we generated a BI-relevant query that outputs the number of delayed flights, grouping them by airline. This required several join and aggregation operations. The following queries were each executed using the data preview function. By setting a high limit it was also noted that the entire results for the queries were displayed in the data preview.

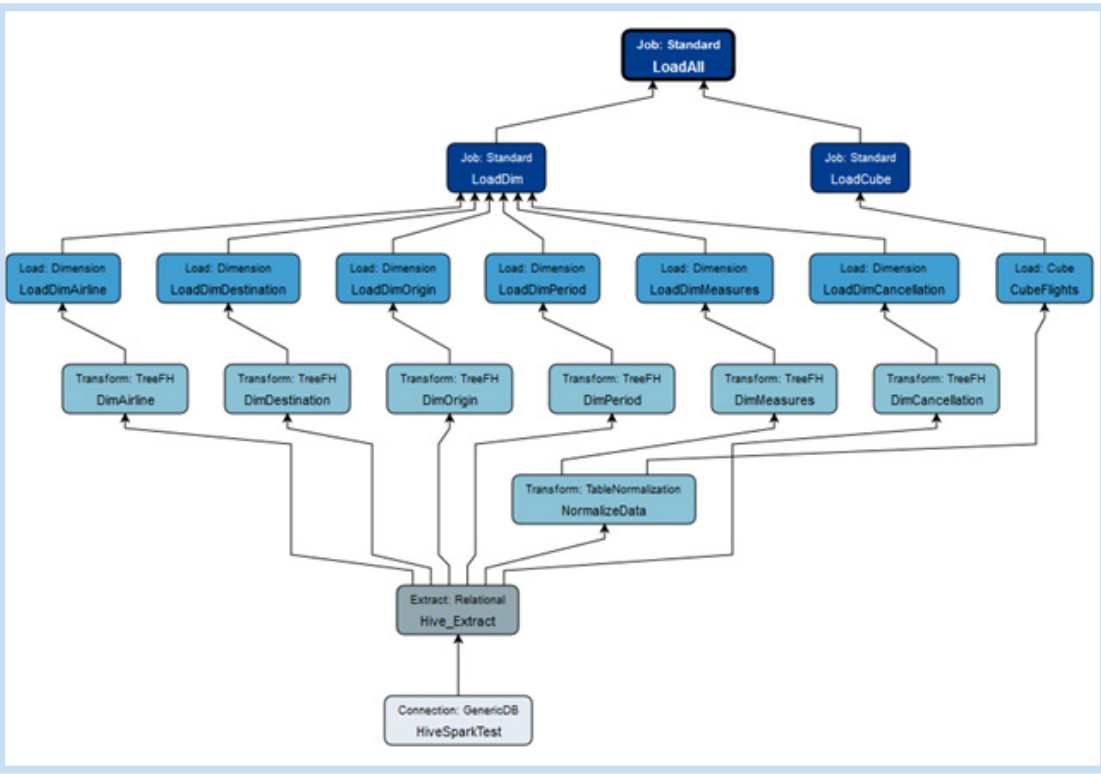


Using this more realistic query, we found a pronounced difference between Hive on Spark and Hadoop MapReduce, with the performance for Hive on Spark being almost 400% higher than that with the default setting! In short, by using Hive on Spark, data analyses can be performed much faster.

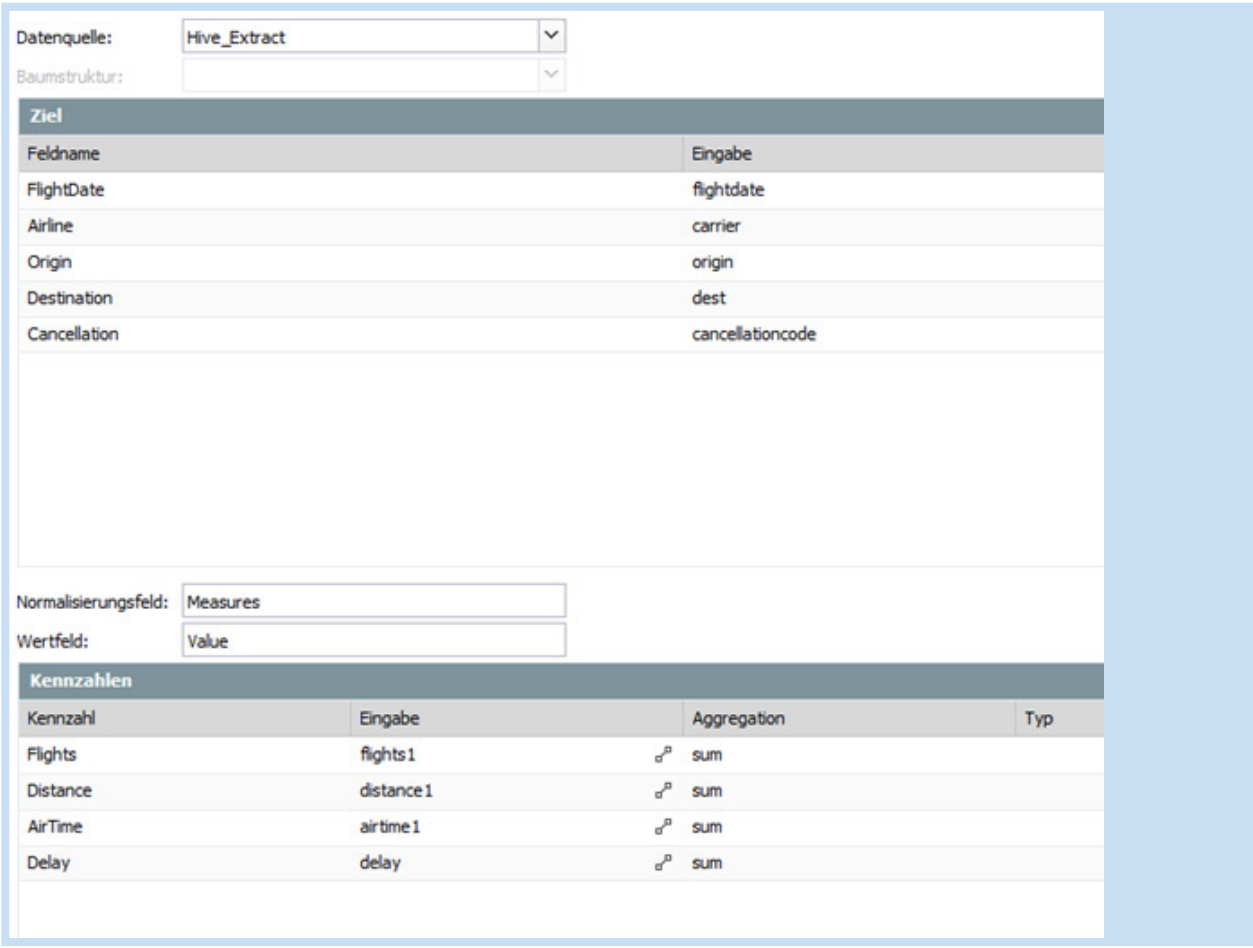
However, the queries with the longest running times in a BI environment occur during the loading of the data cubes and dimension tables. We therefore decided to examine Spark‘s suitability for this in our next test step.

To this end, we examined various load types such as dimension load, cube load and a combination of these two variants. We also looked at the processing speed when writing to a file. This excluded any bottlenecks, for example, caused by the loading process in the in-memory DB.

To be able to better understand / reproduce the measurements, the structure of the cube used should be examined:



The flow diagram illustrates the individual sub-steps of the ETL process. The first two areas, connection and extract, have already been mentioned. As our query we used query 3 as a Hive extract source. The dimension loads can be considered exceptions as we used our own queries for each dimension (see queries 1 and 2).



The TableNormalization transformation shows the fact table structure. It consists of the dimensions FlightDate, Airline, Origin, Destination, Cancellation and Measures. The latter includes the values Flights, Distances, AirTime and Delay.

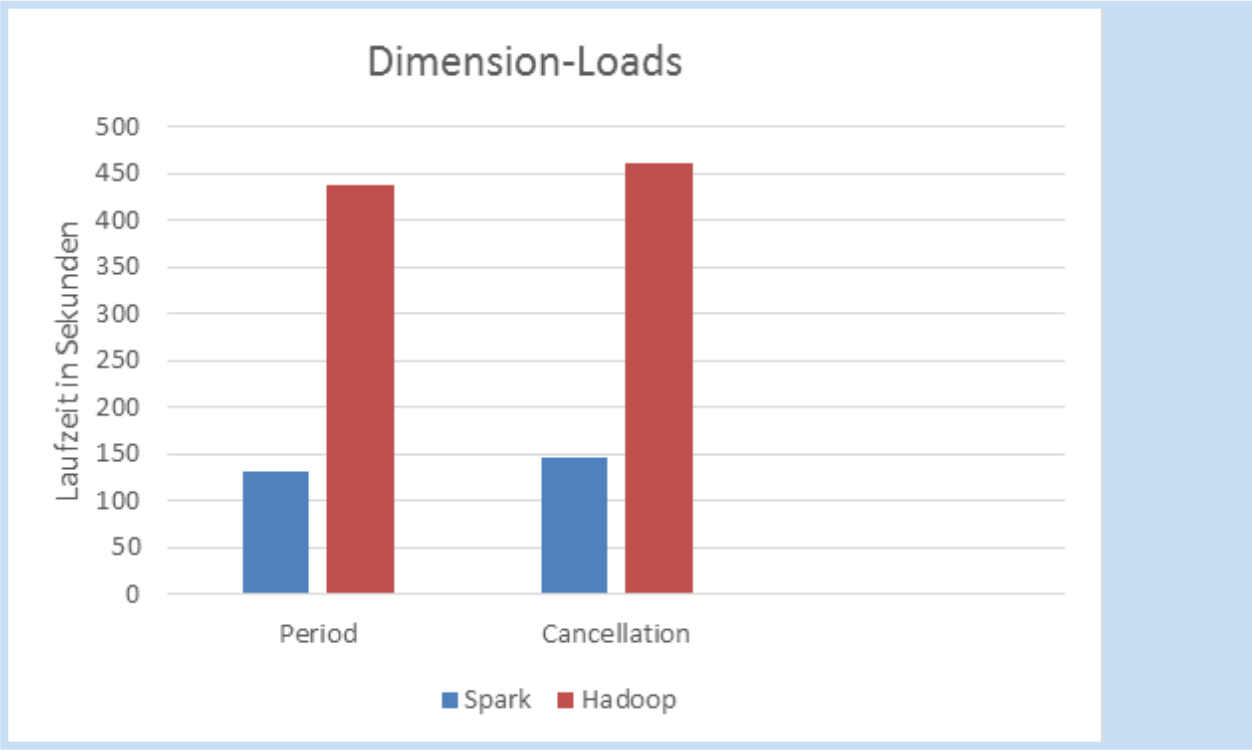
When measuring the dimension loads, we used the two dimensions Period and Cancellation as samples. The other cube dimensions displayed a similar run-time behaviour.

Queries used:

```
select distinct otp_flights.year, otp_flights.quarter, otp_flights.month,
otp_flights.flightdate
from otp_flights where year = 2014 and month = 3
```

```
select distinct otp_flights.cancellationcode
from otp_flights where year = 2014 and month = 3
```

Figures: Query 1 and 2



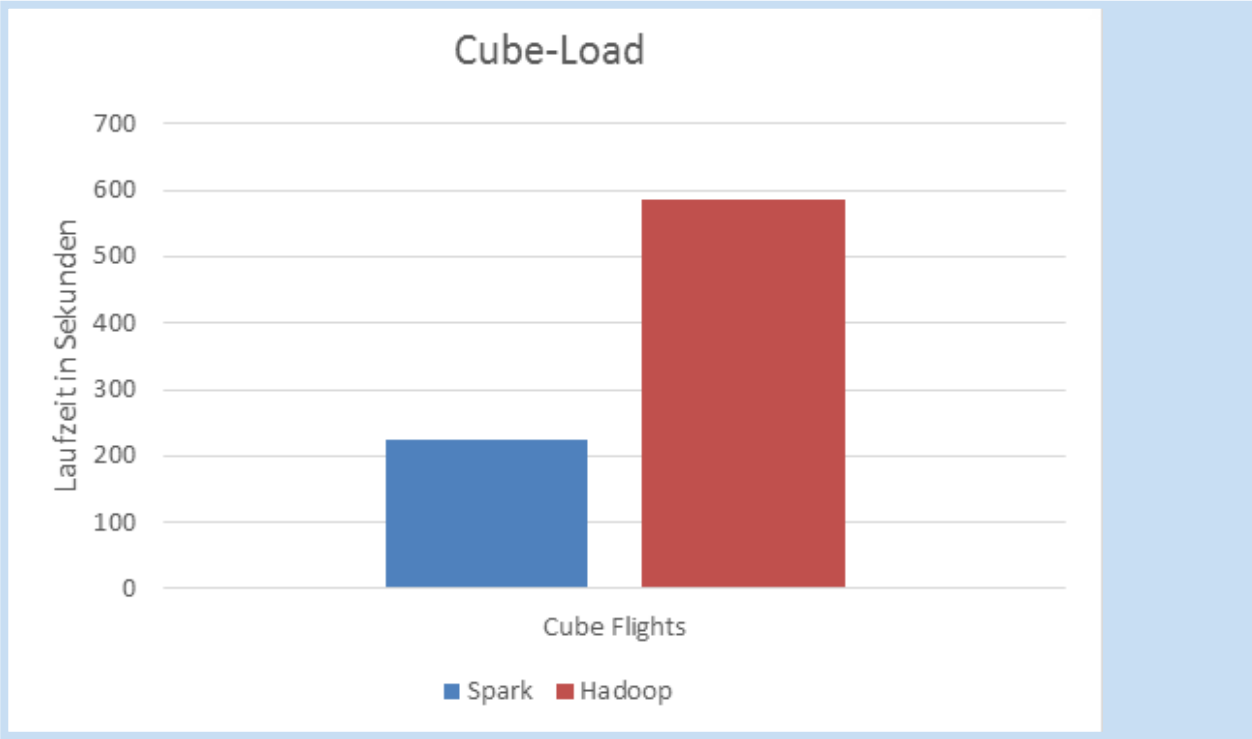
Result:

When loading the dimensions Period and Cancellation, an increase in performance of 334% and 316% was achieved with Spark.

The following query was used for viewing the cube loading run-time behaviour:

```
select year, quarter, month,
flightdate, cancellationcode, carrier,
destcityname, dest, origincityname,
origin, flights, distance,
airtime, arrdelayminutes ,
sum(distance) as distance1, sum(flights) as flights1,
sum(airtime) as airtime1, sum(arrdelayminutes) as delay
from otp_flights group by carrier, year, quarter, month, flightdate, cancellationcode,
destcityname, dest, origincityname, origin, flights, distance, airtime, arrdelayminutes
limit 100000
```

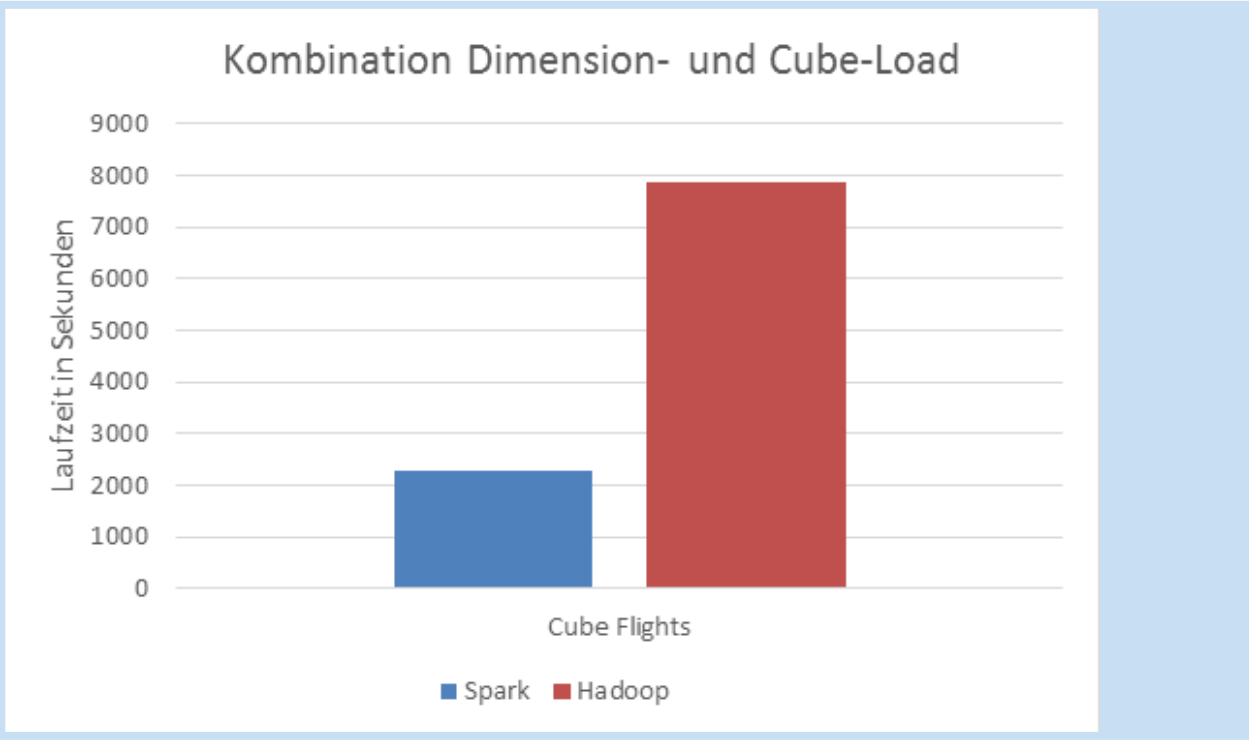
Figure: Query 3



Result:

It was possible to accelerate the loading of the cube by nearly 260% by using Spark.

Using the same query, we tested a combination of dimension and cube loadings as well as writing to a file. In doing this, all individual dimensions were initially loaded one after the other. As soon as this was completed, the cube was loaded.



Result:

We were able to shorten the most time-consuming job in our tests from over two hours running time to 38 minutes. In other words, a reduction of about 344%.

7. Conclusion

Using Hive on Spark with Jedox opens up completely new possibilities for preparing and evaluating Big Data databases. Although there are some associated limitations with the solution discussed here, our tests have shown that with a reasonable amount of effort, it is possible to integrate Hive on Spark with Jedox. As a result, performance gains of up to 400% for Big Data queries can be achieved. This applies especially to complex queries with joins and aggregations, which are often used in business intelligence and business analytics.

In particular, the loading processes for dimension tables and cubes via Spark can be significantly accelerated – by up to 344%. Because this kind of loading process is endemic to each Jedox roll-out, in our opinion a Hive on Spark installation should be considered – even if Spark support is currently in an experimental state. In future, with further development of Hive on Spark, even greater improvements in performance can be expected.

Despite the clear benefits of Hive on Spark, there is, unfortunately, a small down side: Configuration settings for Hive on Spark can only be applied per connection as an additional GET parameter in the JDBC URL. Overriding parameters per query, and thus doing any possible fine-tuning, is not possible with the version of Jedox we used.

Leading in Business Open Source solutions and consulting

it-novum is the leading IT consultancy for Business Open Source in the German-speaking market. Founded in 2001 it-novum today is a subsidiary of the publicly-held KAP Beteiligungs-AG.

We operate with 85 employees from our main office in Fulda and branch offices in Düsseldorf, Dortmund, Vienna and Zurich to serve large SME enterprises as well as big companies in the German-speaking markets.

it-novum is a certified SAP Business Partner and longtime accredited partner of a wide range of Open Source products. We mainly focus on the integration of Open Source with Closed Source and the development of combined Open Source solutions and platforms.

Due to the ISO 9001 certification it-novum belongs to one of the few Open Source specialists who can prove the business suitability of their solutions, proven by international quality standards.

More than 15 years of Open Source project experience

- Our portfolio contains a wide range of Open Source solutions within the applications and infrastructure area as well as own product developments which are well-established in the market.
- As an IT consulting company with a profound technical know-how within the Business Open Source area we differentiate ourselves from the big solution providers' standard offerings. Because our solutions are not only scalable and flexible but also integrate seamlessly in your existing IT infrastructure.
- We can assemle multidisciplinary project teams, consisting of engineers, consultants and business data processing specialists. Thus we combine business know-how with technological excellence to build sustainable business processes.
- Our target is to provide you with a high-quality level of consulting during all project phases – from the analysis and conception up to the implementation and support.
- As a decision-making basis prior to the project's start we offer you a Proof-of-Concept. Through a real-case simulation and a developed prototype you can decide on a new software without taking any risks. Moreover, you benefit from:
  - Security and predictability
  - Clear project methodology
  - Sensible calculation

Your contact person for Business Intelligence and Big Data:

Stefan Müller
Director Big Data Analytics
stefan.mueller@it-novum.com
+49 (0) 661 103 942

it-novum GmbH Germany
Headquarters Fulda: Edelzeller Straße 44 · 36043 Fulda
Phone: +49 (0) 661 103 333
Branches in Dusseldorf & Dortmund

it-novum branch Austria
Ausstellungsstraße 50 / Zugang C · 1020 Vienna
Phone: +43 1 205 774 1041

it-novum GmbH Switzerland
Hotelstrasse 1 · 8058 Zurich
Phone: +41 (0) 44 567 62 07