it-novum
Open Business Solutions. Delivered.

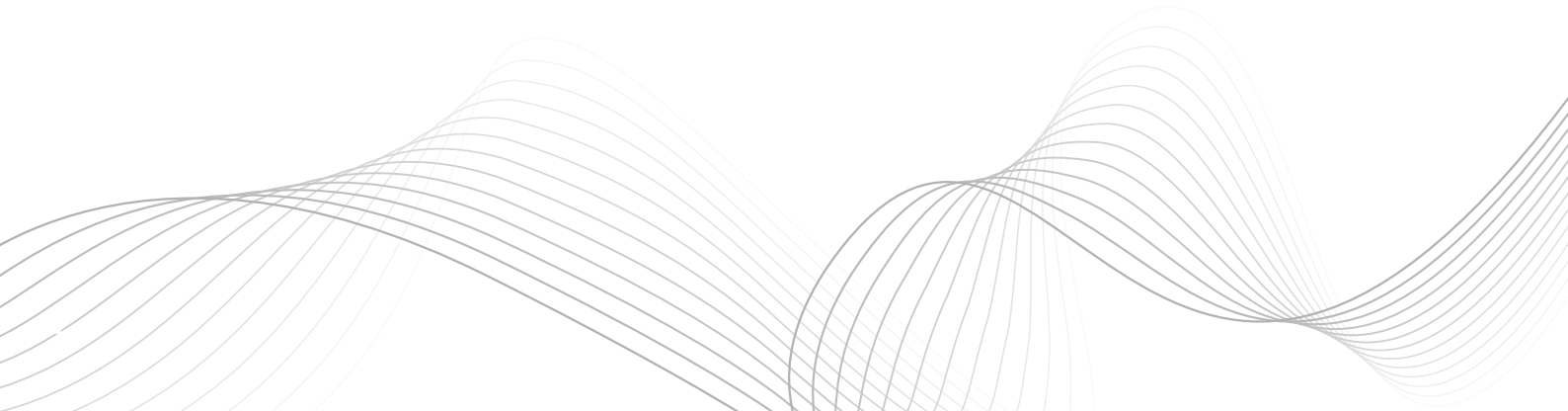
Research Paper

Big Data: Hive on Spark mit Jedox nutzen

Autoren:
Alexander Keidel, Michael Deuchert und Phillip Musch

Getestet mit:
Jedox: 6.0, 6.0 SR1
Cloudera CDH: 5.4.8

Inhalt



1. Big Data-Datenbestände mit Jedox und Hive on Spark auswerten

Mit dem Release von Hive 1.1 im Januar 2015 hat Apache Spark in Hive Einzug gehalten. Bei Apache Spark handelt es sich um ein Open Source-Projekt aus dem Bereich Cluster Computing zur Verarbeitung von großen Datenmengen.

Im Gegensatz zu Map-Reduce über Hadoop versucht Spark, viele Operationen bei der Datenverarbeitung im Arbeitsspeicher durchzuführen (In-Memory) und Zugriffe auf das HDFS gering zu halten. Für einige Anwendungsfälle ist es daher bis zu 100-mal schneller als Map-Reduce. Das betrifft insbesondere Anwendungen mit vielen Reduce-Schritten wie sie z.B. bei der Übersetzung von komplexen Queries oder im Business Intelligence-Umfeld an der Tagesordnung sind.

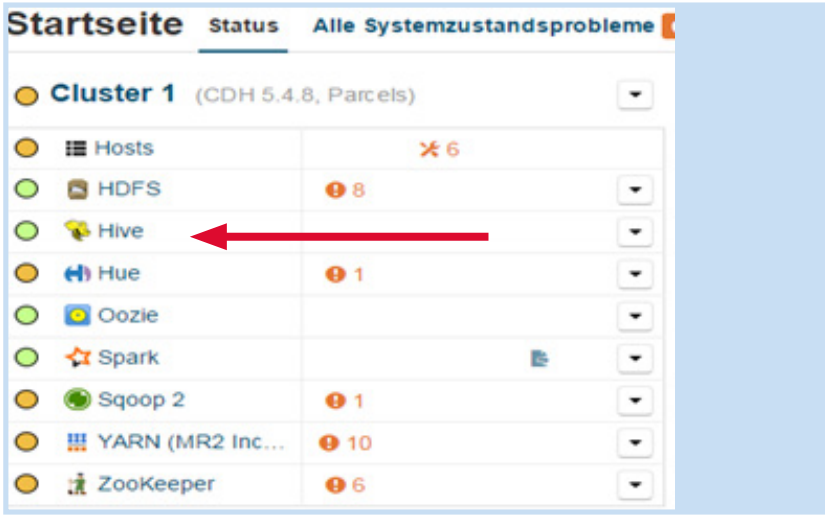
Bislang war es nicht möglich, Spark in der Business Intelligence-Plattform von Jedox zu nutzen und so von den Performancegewinnen zu profitieren. Aus diesem Grund haben wir diese Untersuchung gestartet: wir möchten zeigen, wie Spark in Verbindung mit Hive und mit einigen Einschränkungen auch für Jedox genutzt werden kann. Und dies, ohne dass Anpassungen an den Queries oder in Jedox vorgenommen werden müssen.

2. Konfiguration von Hive on Spark mit Hilfe des Cloudera Cluster Manager

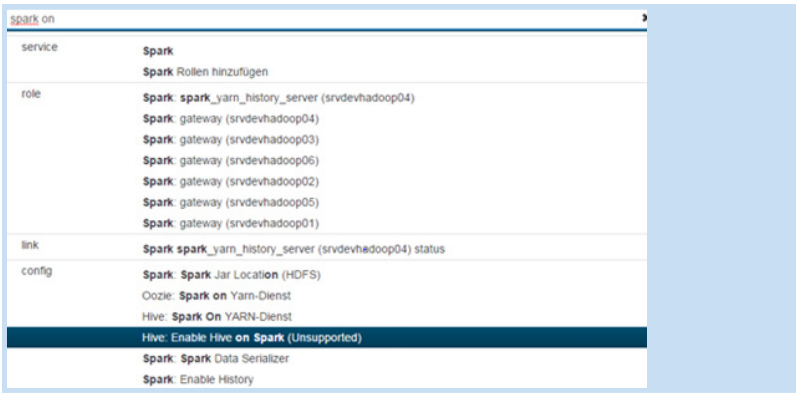
Die komfortabelste Möglichkeit, dem Nutzer *Hive on Spark* zu bieten, ist der Einsatz des Cloudera Software Stacks, wie er auch bei it-novum intensiv genutzt wird. Die Konfiguration kann bequem über die Oberfläche des Cloudera Cluster Managers erfolgen. Zu beachten ist allerdings, dass *Hive on Spark* erst ab Version 5.4.x enthalten ist und von Cloudera noch nicht für den Produktivbetrieb freigegeben wurde. Das heißt, dass es von Cloudera noch keinen Support für den Einsatz von *Hive on Spark* gibt.

Für unsere Tests haben wir die neueste Cloudera Version (CDH 5.4.8) eingesetzt. Um Cloudera entsprechend zu konfigurieren, müssen folgende Schritte durchgeführt werden:

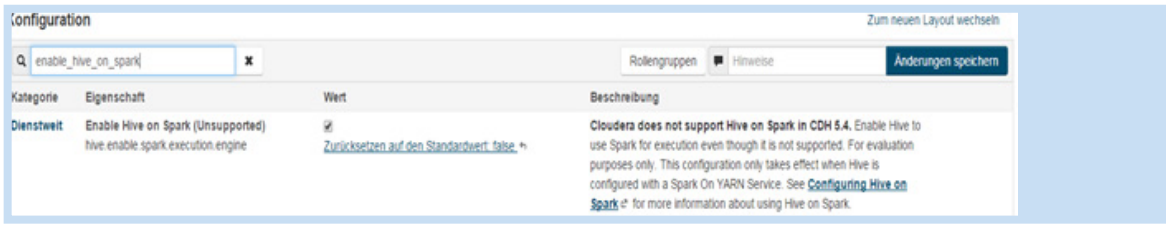
1) Click auf den Dienst Hive:



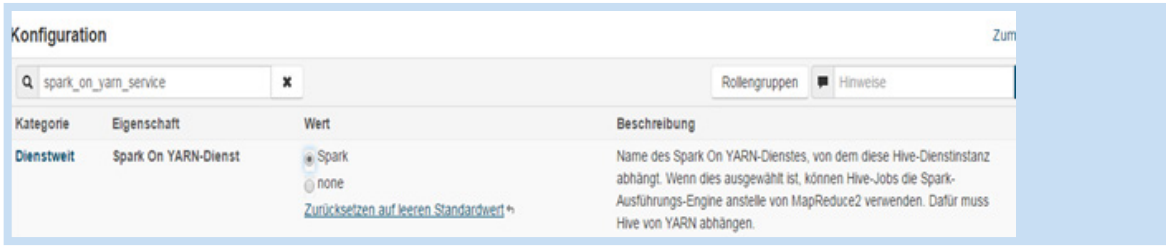
2) Im Suchfenster des Cloudera Manager muss die Konfigurationseinstellung **„Hive: Enable Hive on Spark“** mit einem Doppelklick ausgewählt werden:



3) Durch das Setzen der Checkbox und einem anschließendem Klick auf **„Änderungen speichern“** erzeugt Cloudera die entsprechenden Konfigurationseinstellungen:

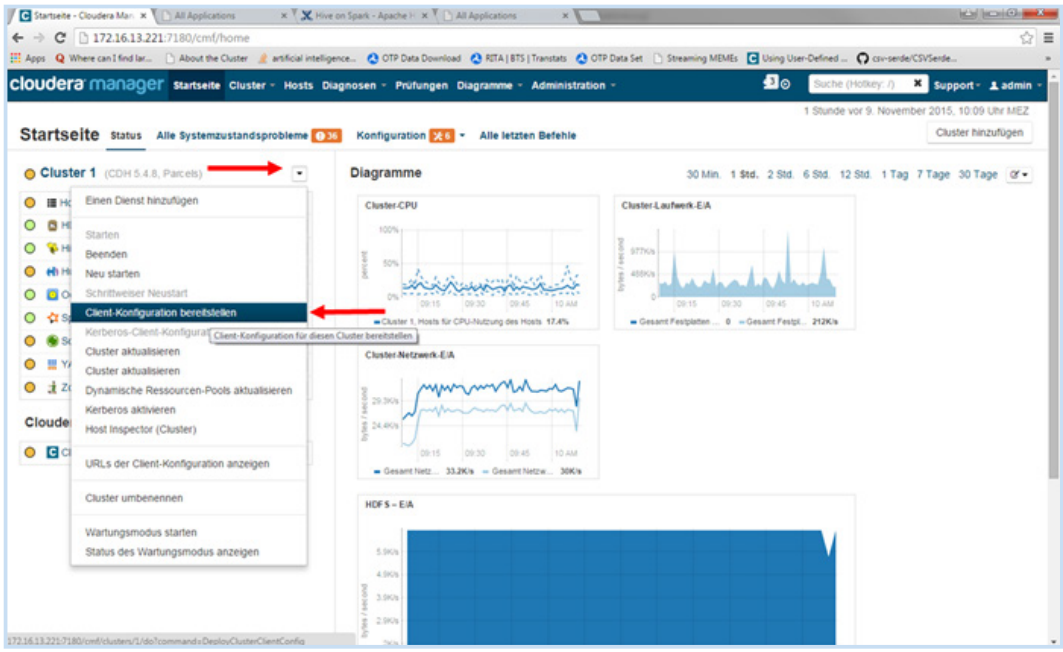


4) Eine Besonderheit bei Cloudera ist, dass *Hive on Spark* nur in Verbindung mit YARN genutzt werden kann. Das bedeutet zunächst eine Einschränkung, da dadurch eine zu sätzliche Abhängigkeit von Spark zu Hadoop entsteht. Aber es hat andererseits den Vorteil, dass sowohl Spark- als auch Hadoop MapReduce Jobs über eine einheitliche Schnittstelle verwalten können.

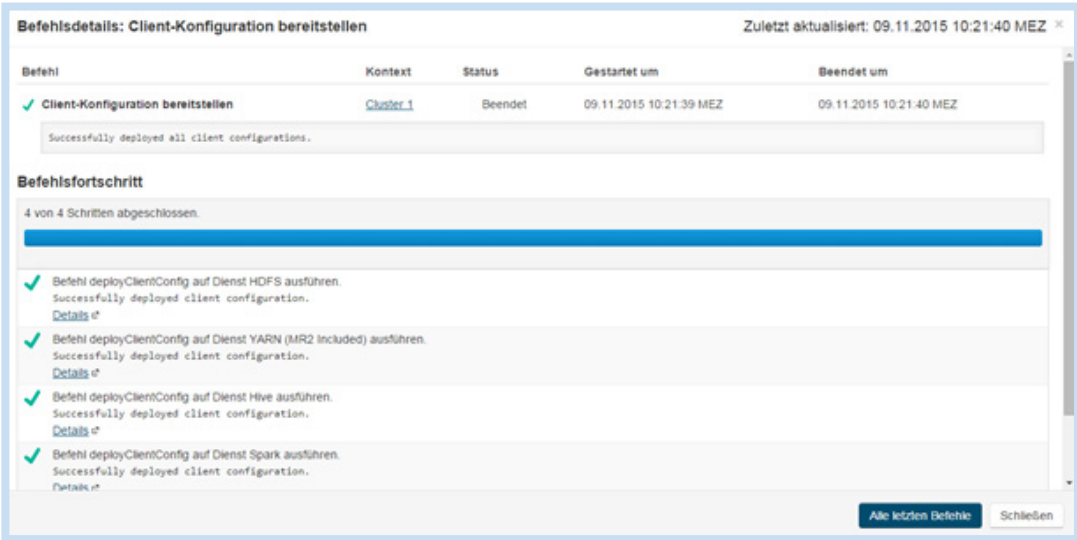


Um Spark on Yarn in Cloudera zu konfigurieren, muss zunächst im Suchfenster des Cloudera Cluster Manager „spark_on_yarn-service“ eingegeben werden. Danach klickt man auf Spark und anschließend auf „Änderungen speichern“, damit Cloudera die pas senden Konfigurationseinstellungen erzeugt.

5) Im letzten Schritt müssen die in den Schritten 1 bis 4 erzeugten Konfigurationseinstellungen an die einzelnen Cluster-Instanzen verteilt werden. Das erreicht man über einen Klick auf „Client Konfiguration bereitstellen“ im Kontextmenu des Clusters:



6) Nach der Bereitstellung der Clientkonfiguration kann *Hive on Spark* mit Cloudera genutzt werden:



3. Testen von Hive on Spark

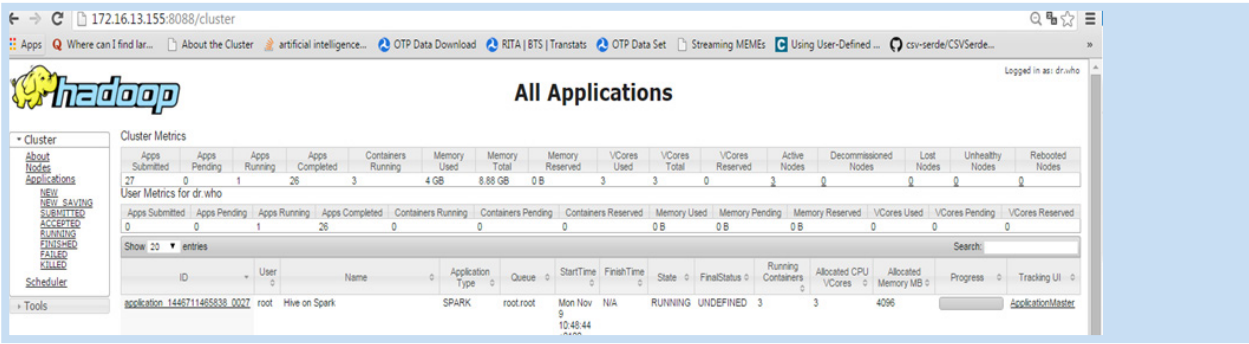
Das Testen der unter Abschnitt 1 erzeugten Konfiguration erfolgt am einfachsten über die Hive Shell. Dazu musst zunächst die Hive Execution Engine geändert werden; diese ist als Default auf MapReduce über Hadoop eingestellt. Der Befehl „set hive.execution.engine =spark“ in der Shell ändert die hive.execution engine für die jeweilige Session auf Spark:

hive>set hive.execution.engine=spark;

Eine Query, wie beispielsweise zum Anzeigen der Zeilenanzahl:

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

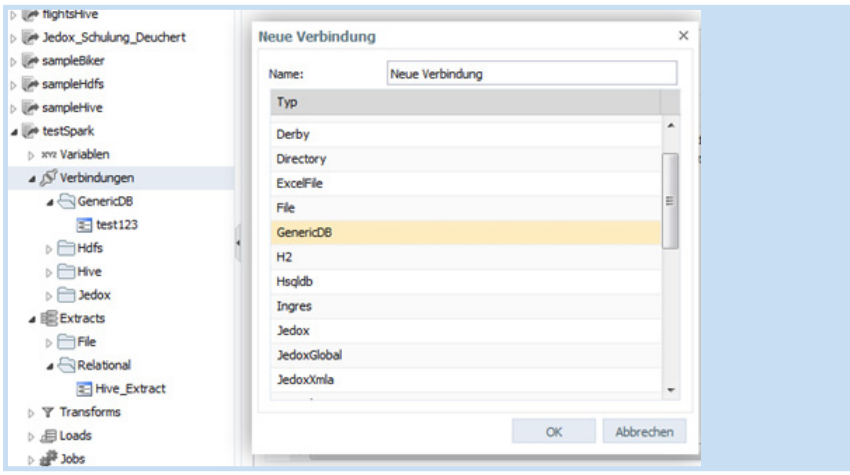
erzeugt nun einen Spark Job, der im Yarn Job Tracker angezeigt wird.



Hinweis: Queries wie z.B. Select * From my_table erzeugen in Hive keinen MapReduce oder Spark Job, da sie technisch gesehen lediglich ein Lesen der HDFS-Datei darstellen, die für die Tabelle hinterlegt ist.

4. Einrichten von Hive on Spark in Jedox

Um *Hive on Spark* mit Jedox nutzen zu können, benötigen wir zusätzlich den Jedox Hadoop Connector. In Version 6 von Jedox muss er als zusätzliches Package installiert werden. Es kann nun eine Standard Hive Verbindung (über MapReduce) mit dem neuen Verbindungstyp „Hive“ angelegt werden. In Jedox stellen wir zunächst im ersten Schritt eine Verbindung zu Hive her und setzen anschließend die Hive Execution Engine auf Spark. Das ist in Jedox momentan nur über Umwege möglich. Zuerst legt man dafür über Generic Database Connection eine neue Verbindung an:



…und trägt folgende Werte ein:

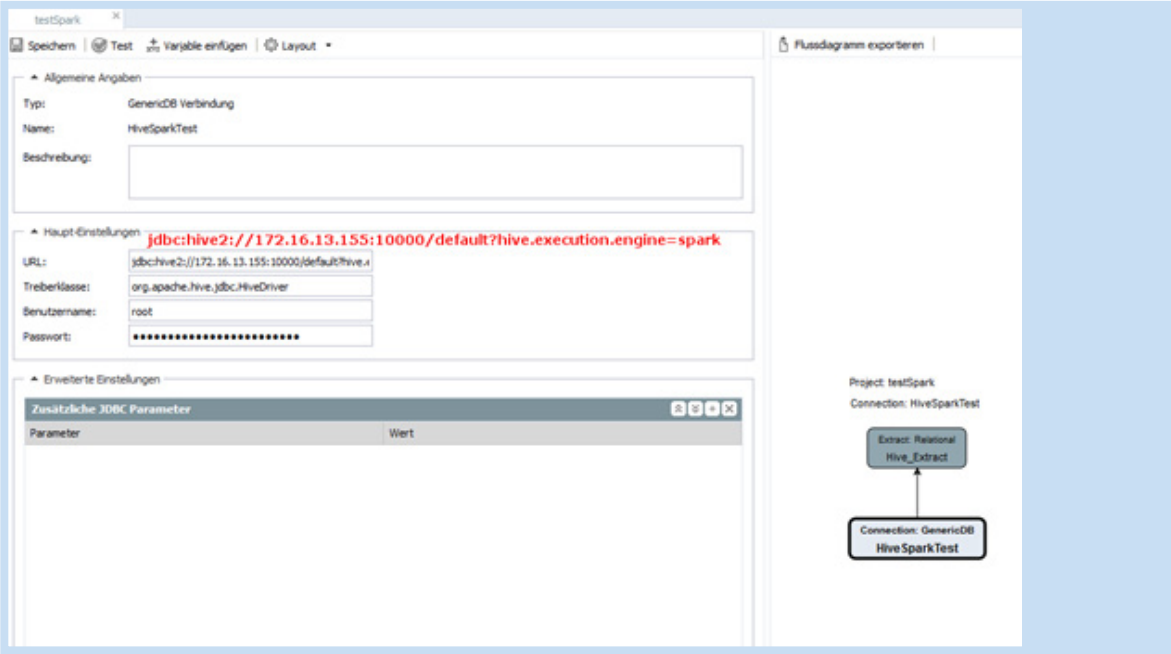
- URL
- Treiberklasse
- Benutzername
- Passwort

Die URL ist ein JDBC-String und setzt sich für die Verbindung mit Hive wie folgt zusammen:

Jdbc:hive2://IP_HIVE2_SERVER:HIVE_TCP_THRIFT_PORT/HIVE_DB

Um jetzt *Hive on Spark* nutzen zu können, muss man allerdings zusätzlich einen GET-Parameter anhängen, der beim Verbindungsaufbau mit dem Hive Server automatisch ausgeführt wird. Der generische JDBC-String für die Verwendung von *Hive on Spark* lautet:

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



Die von uns verwendete Treiber-Klasse ist der offizielle Hive-JDBC-Treiber:

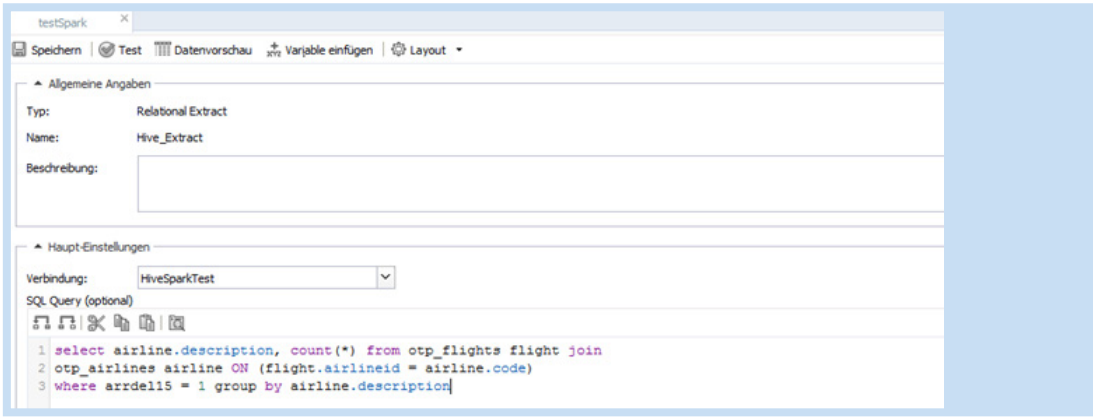
org.apache.hive.jdbc.HiveDriver.

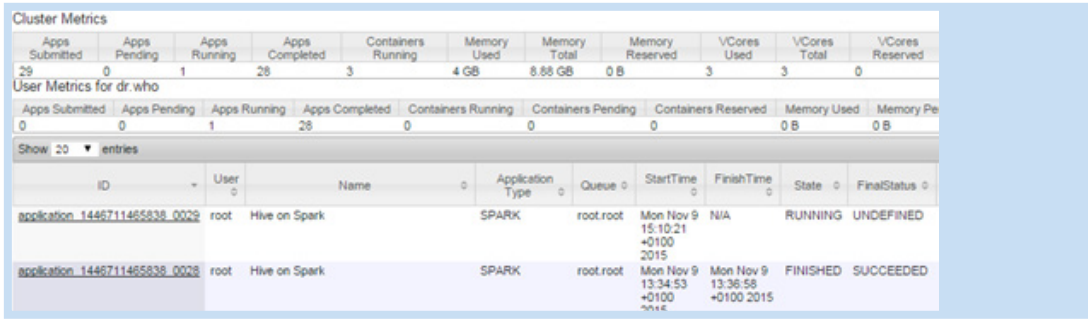
Die Verbindung kann nun mit einem Klick auf den Test-Button getestet werden.

Hinweis: In Jedox 6.0 SR2 wird diese Konfiguration direkt über die GUI über den Connection-Typ Hive möglich sein und damit stark vereinfacht. Eine ausführliche Dokumentation der Hive-Konfigurationsparameter und deren Übergabe in Form einer JDBC URL ist unter Hive Configuration Properties sowie Hive2 Connection URL Format erhältlich.

5. Hive on Spark mit Jedox

Mit der im letzten Abschnitt beschriebenen Verbindung werden die Queries an die Hive Table nicht länger als MapReduce-Job über Hadoop, sondern über Spark ausgeführt. So erzeugt z.B. die Query einen Spark Job, der wie bereits in Abschnitt 1 beschrieben im YARN Cluster Manager betrachtet werden kann.



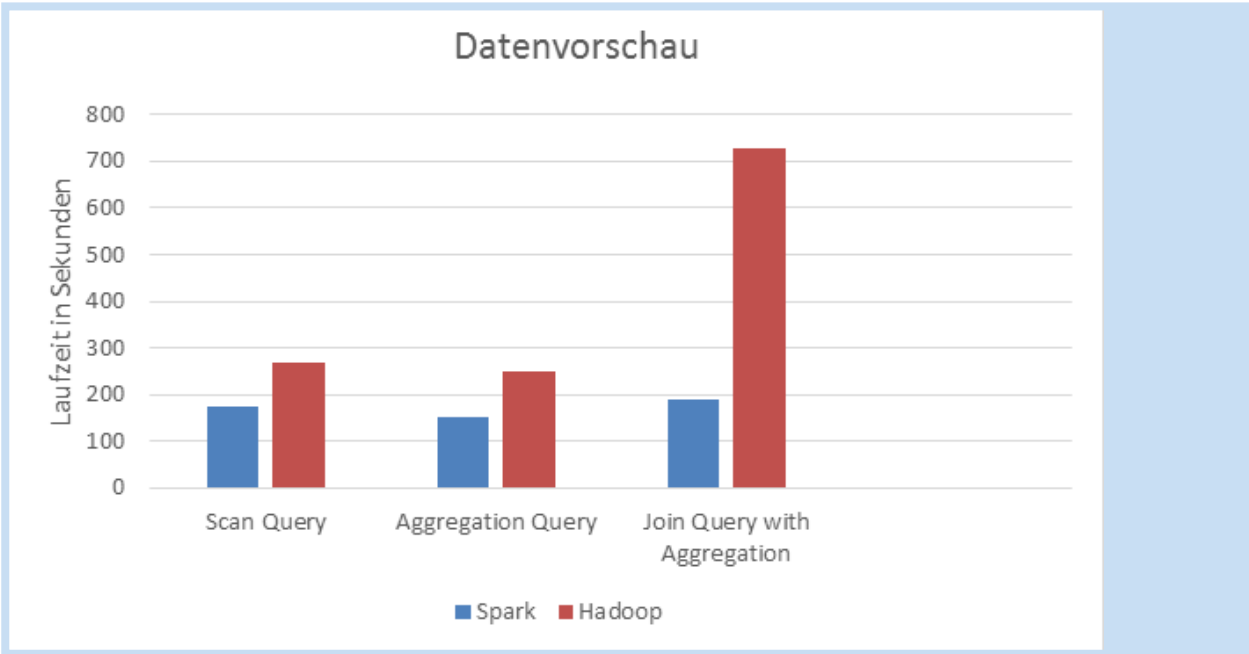


Nach dem Ende des Spark Jobs werden die Ergebnisse an Jedox übertragen. Hier können sie wie gewohnt weiter bearbeitet werden.

6. Messungen

Um messen zu können, wie die Leistung von *Hive on Spark* in Verbindung mit Jedox ist, haben wir mit drei verschiedenen Queries Messungen durchgeführt. Als Testdaten dienten uns Flugdaten aus den USA von Januar 2014 bis August 2015, ein unter http://www.transtats.bts.gov frei verfügbarer Datensatz mit ca. 8,2 Millionen Zeilen.

Die erste Messung, eine einfache Scan Query, diente dazu, den Datensatz nach den Daten vom März 2014 zu filtern. Die zweite Messung bestand aus einer Aggregation Query und hatte zum Ziel, die Anzahl aller Zeilen der Datenbank auszugeben. Als letztes haben wir eine für das BI-Umfeld realistische Query erzeugt, welche die Anzahl aller verspäteten Flüge ausgibt und nach Fluggesellschaft gruppiert. Dafür sind mehrere Join- und Aggregationen-Operationen notwendig. Die folgenden Queries wurden jeweils über die Datenvorschaufunktion ausgeführt. Durch das Setzen eines entsprechend hohen Limits wurde auch beachtet, dass das gesamte Ergebnis der Query in der Datenvorschau angezeigt wird.

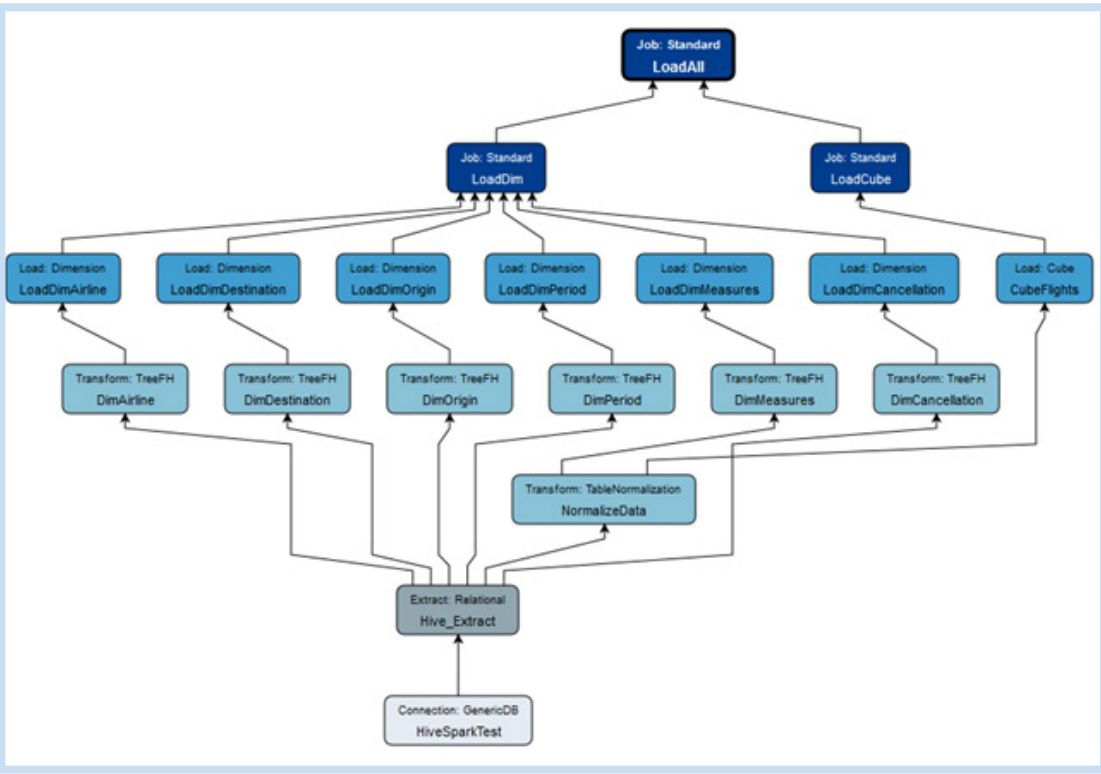


Bei dieser realistischeren Anfrage zeigt sich ein deutlicher Unterschied zwischen *Hive on Spark* und Hadoop MapReduce: die Performanz ist um fast 400% höher als bei der Default-Einstellung! Mit *Hive on Spark* lassen sich Datenanalysen also wesentlich schneller durchführen.

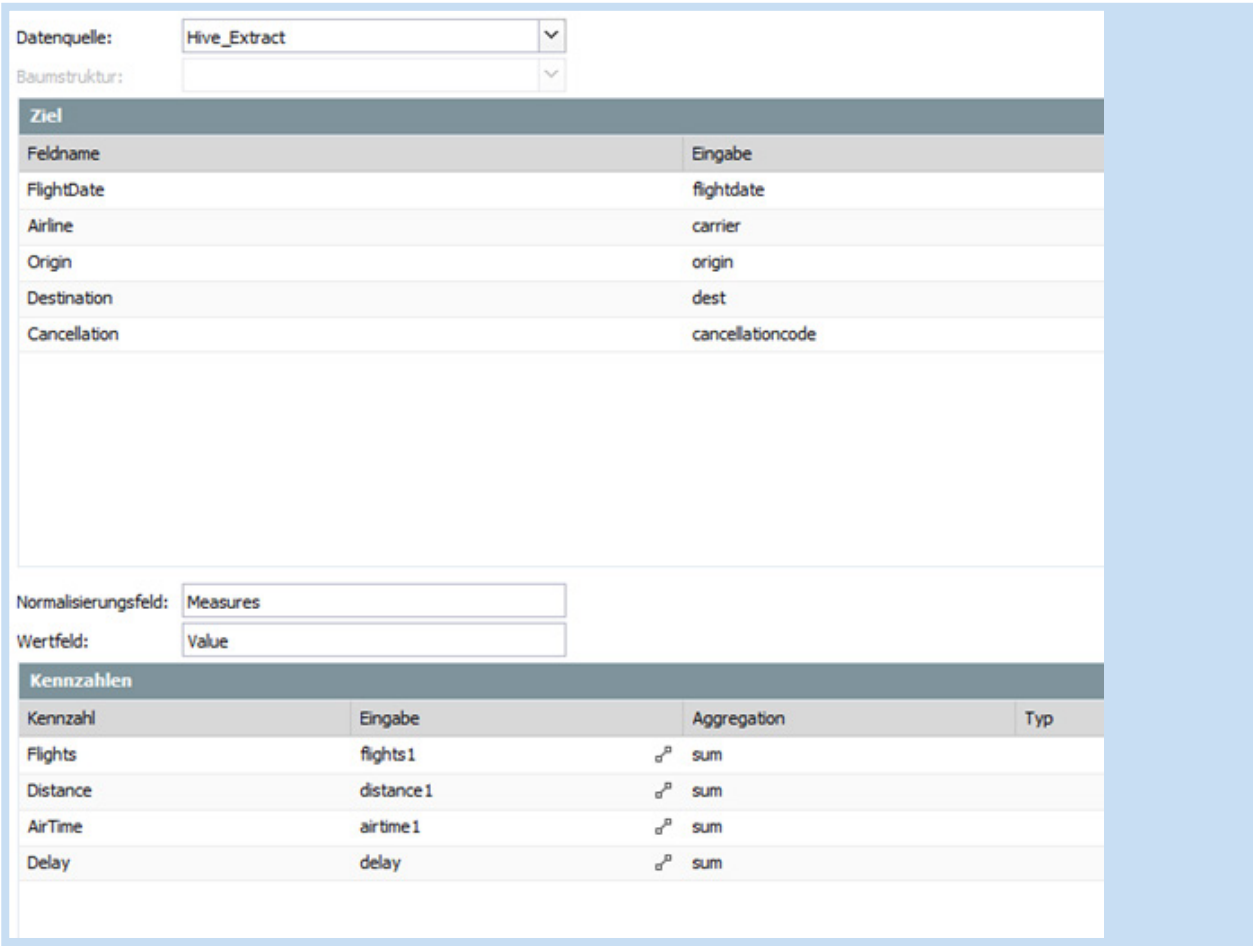
Die Queries mit den längsten Laufzeiten im BI-Umfeld kommen allerdings beim Beladen der Datencubes und Dimensionstabellen vor. Wir haben deshalb in einem nächsten Schritt die Eignung von Spark für diesen Einsatzzweck untersucht.

Dazu haben wir diverse Load-Typen wie Dimension-Load, Cube-Load und eine Kombination der beiden Varianten untersucht. Zusätzlich haben wir die Verarbeitungsgeschwindigkeiten beim Schreiben in eine Datei betrachtet. Dadurch werden eventuelle Bottlenecks z.B. durch den Ladeprozess in die In-memory DB ausgeschlossen.

Um die Messungen besser nachvollziehen bzw. reproduzieren zu können, soll zunächst auf den Aufbau des verwendeten Cubes eingegangen werden:



Das Flussdiagramm verdeutlicht die einzelnen Teilschritte des ETL-Prozesses. Die ersten beiden Bereiche Connection und Extract wurden bereits erwähnt. Als Abfrage haben wir Query 3 als Hive Extract Quelle benutzt. Ausnahmen bilden die Dimension Loads, bei dem wir für jede Dimension eigene Queries verwendet haben (siehe Queries 1 und 2).



Die TableNormalization Transformation zeigt den Aufbau der Faktentabelle. Sie setzt sich aus den Dimensionen FlightDate, Airline, Origin, Destination, Cancellation sowie Measures zusammen. Letztere beinhaltet die Kennzahlen Flights, Distances, AirTime und Delay.

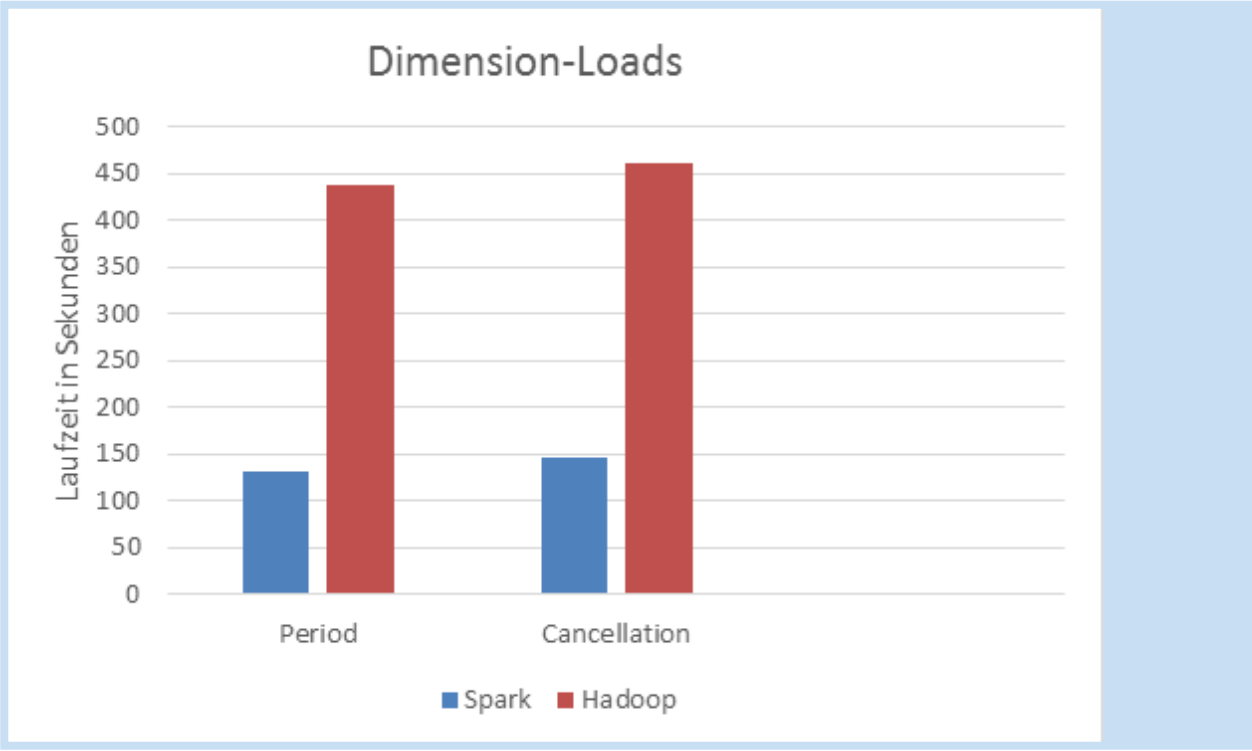
Bei der Messung der Dimension-Loads haben wir als Stichproben die beiden Dimensionen Period und Cancellation untersucht. Die weiteren Dimensionen des Würfels weisen ein ähnliches Laufzeitverhalten auf.

Verwendete Queries:

```
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
```

Abbildung: Query 1 und 2



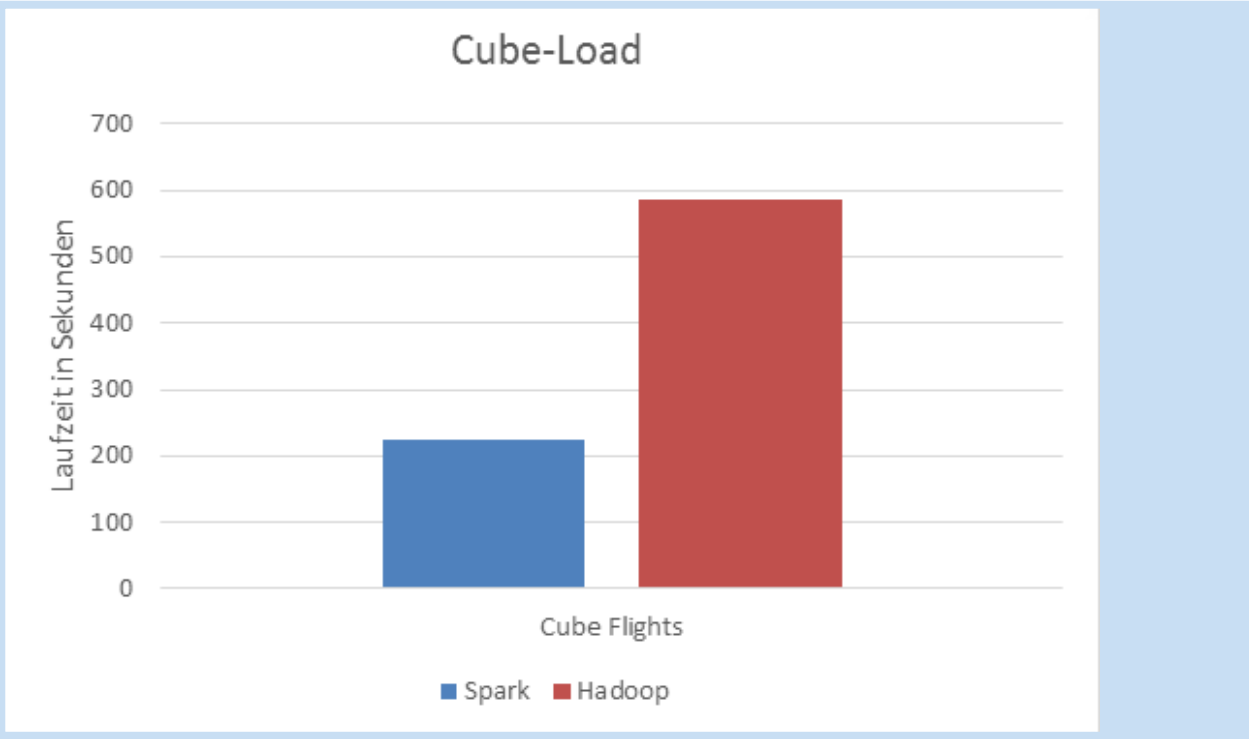
Ergebnis:

Beim Laden der Dimensionen Period und Cancellation konnten mit Spark Leistungssteigerungen von 334 % und 316 % erreicht werden.

Für die Betrachtung des Laufzeitverhaltens des Cube-Load kam die folgende Query zum Einsatz:

```
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
```

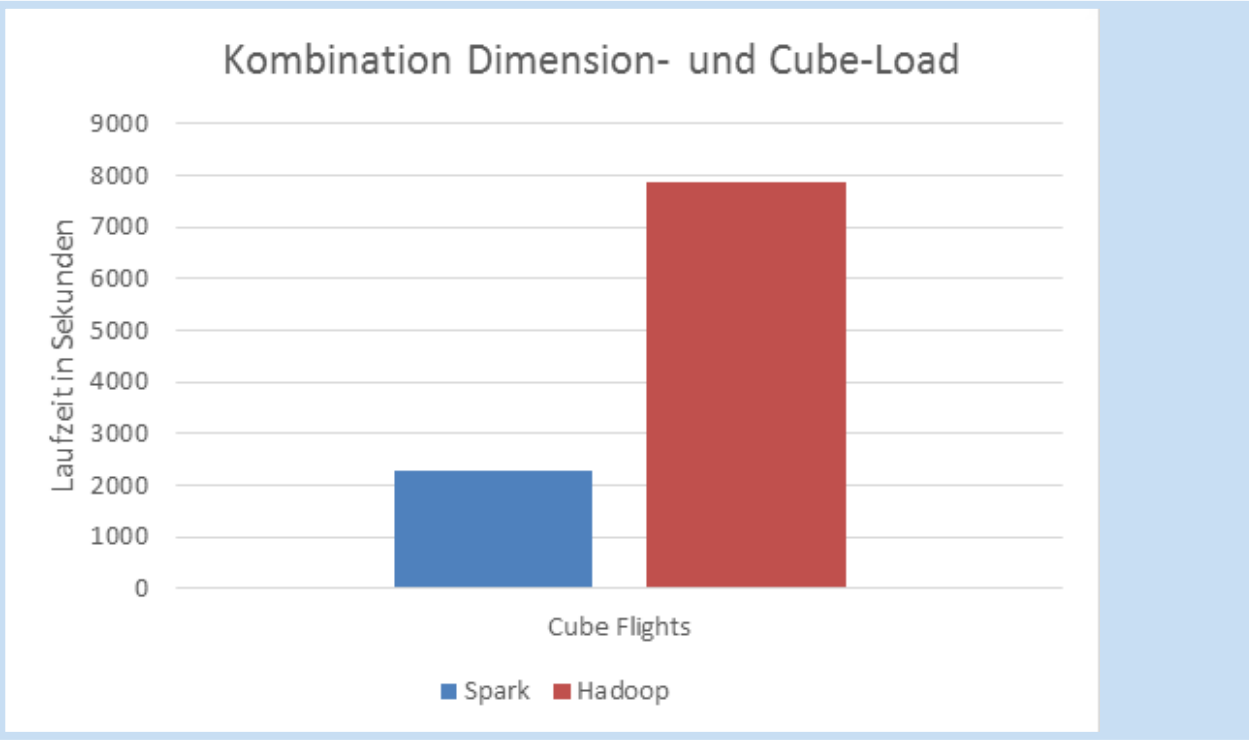
Abbildung: Query 3



Ergebnis:

Das Laden des Würfels konnte mit Spark um knapp 260 % beschleunigt werden.

Mit der gleichen Query haben wir auch eine Kombination von Dimension- und Cube-Load sowie das Schreiben in eine Datei getestet. Dabei werden zunächst alle einzelnen Dimensionen nacheinander geladen. Sobald dies abgeschlossen ist, folgt der Load des Würfels.



Ergebnis:

Der zeitaufwendigste Job unserer Betrachtung konnte von über zwei Stunden Laufzeit auf 38 Minuten verkürzt werden. Das bedeutet eine Verkürzung der Laufzeit um etwa 344 %.

7. Fazit

Die Nutzung von *Hive on Spark* in Verbindung mit Jedox eröffnet ganz neue Möglichkeiten für die Aufbereitung und Auswertung von Big Data-Datenbeständen. Obwohl die vorgestellte Lösung mit einigen Einschränkungen verbunden ist, zeigen unsere Tests, dass mit geringem Aufwand die Integration von *Hive on Spark* in Jedox möglich ist. Dadurch lassen sich Performancegewinne von bis zu 400% für Big Data Queries erreichen. Das betrifft insbesondere komplexe Queries mit vielen Joins und Aggregationen wie sie oft im Umfeld von Business Intelligence und Business Analytics vorkommen.

Insbesondere die Ladeprozesse der Dimensionstabellen und Cubes durch Spark werden um bis zu 344 % erheblich beschleunigt. Da diese Art von Ladeprozessen in jedem Jedox Roll-out vorkommt, sollte unserer Meinung nach ein *Hive on Spark* Einsatz grundsätzlich in Betracht gezogen werden - auch wenn sich die Spark-Unterstützung zum aktuellen Zeitpunkt noch in einem experimentellen Status befindet. Zukünftig ist bei einer Weiterentwicklung von *Hive on Spark* mit noch größeren Leistungssteigerungen zu rechnen.

Trotz der deutlichen Vorteile von *Hive on Spark* gibt es leider auch einen kleinen Wermutstropfen: Konfigurationseinstellungen für *Hive on Spark* können immer nur pro Verbindung als zusätzlicher GET-Parameter in der JDBC URL angelegt werden. Das Überschreiben von Parametern pro Query und damit ein mögliches Tuning ist in der von uns benutzen Jedox-Version nicht möglich.

Führend in Business Open Source-Lösungen und -Beratung

it-novum ist das führende IT-Beratungsunternehmen für Business Open Source im deutschsprachigen Markt. Gegründet 2001 ist it-novum heute eine Konzerntochter der börsennotierten KAP Beteiligungs-AG.

Mit unseren 85 Mitarbeitern betreuen wir vom Hauptsitz in Fulda und den Niederlassungen in Düsseldorf, Dortmund, Wien und Zürich aus vorwiegend große Mittelstandskunden sowie Großunternehmen im deutschsprachigen Raum.

Wir sind zertifizierter SAP Business Partner und langjähriger akkreditierter Partner zahlreicher Open Source-Produkte. Unsere Schwerpunkte sind die Integration von Open Source mit Closed Source und die Entwicklung kombinierter Open Source-Lösungen und -Plattformen.

Mit seiner ISO 9001 Zertifizierung gehört it-novum zu den wenigen Open Source-Spezialisten, die die Businesstauglichkeit ihrer Lösungen auch durch ein Qualitätssicherungssystem belegen.

TÜV SAAR CERT ISO 9001

Über 15 Jahre Open Source-Projekterfahrung

- Unser Portfolio umfasst die vielfältige Bandbreite von Open Source-Lösungen im Applications- und Infrastruktur-Bereich sowie eigene, im Markt etablierte Produktentwicklungen.
- Als IT-Beratungshaus mit profunder technischer Expertise im Business Open Source-Bereich grenzen wir uns von den Standardangeboten der großen Lösungsanbieter ab. Denn unsere Lösungen sind nicht nur skalierbar und flexibel anpassbar, sondern fügen sich auch nahtlos in Ihre bestehende IT-Infrastruktur ein.
- Wir stellen fachübergreifende Projektteams zur Verfügung, bestehend aus Entwicklern, Consultants und Wirtschaftsinformatikern. So verbinden wir Business Know-how mit Technologieexzellenz und schaffen nachhaltige Geschäftsprozesse.
- Unser Ziel ist es, Ihnen eine qualitativ hochwertige Beratung in allen Projektphasen zu bieten – von der Analyse, über die Konzeption bis hin zu Umsetzung und Support.
- Als Entscheidungshilfe vor Projektbeginn bieten wir Ihnen einen Proof-of-Concept an. Durch die Praxissimulation und den erstellten Prototypen können Sie sich risikofrei für eine neue Software entscheiden und profitieren von Sicherheit und Planbarkeit, klare Projektmethodik und vernünftige Kalkulation.

Ihr Ansprechpartner für Business Intelligence und Big Data:

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

it-novum GmbH Deutschland
Hauptsitz Fulda: Edelzeller Straße 44 · 36043 Fulda
Telefon: +49 (0) 661 103 333
Niederlassungen in Düsseldorf & Dortmund

it-novum Zweigniederlassung Österreich
Ausstellunsstraße 50 / Zugang C · 1020 Wien
Telefon: +43 1 205 774 1041

it-novum Schweiz GmbH
Hotelstrasse 1 · 8058 Zürich
Telefon: +41 (0) 44 567 62 07