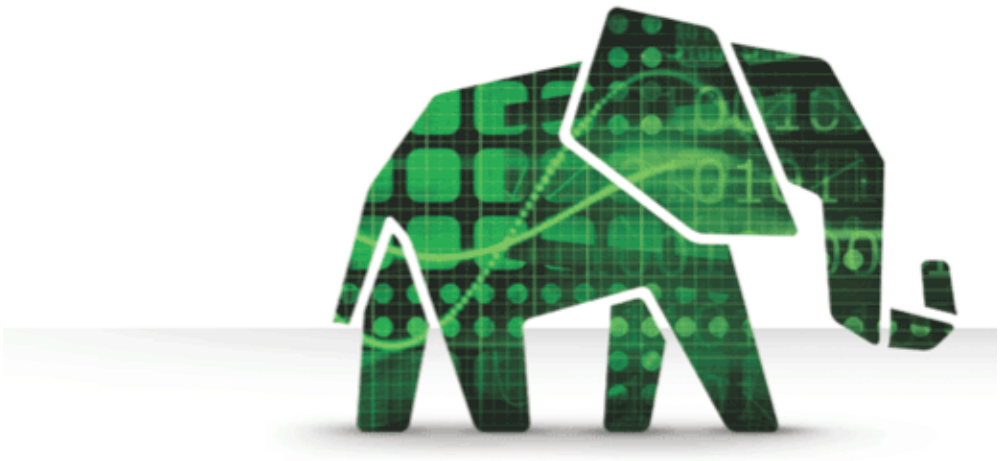
Hortonworks We do Hadoop.

Best Practices for Hadoop Data Analysis with Tableau

September 2013



http://www.hortonworks.com

Tableau 6.1.4 introduced the ability to visualize large, complex data stored in Apache Hadoop with Hortonworks Data Platform (HDP) via Hive and the Hortonworks Hive ODBC driver. This whitepaper discusses best practices, known limitations and advanced techniques that will help Tableau users discover and share insight from their Hadoop data.

About This Article

Intended Audience

This article is for analysts and data scientists, but will also help developers and IT admins. Developers who build complex algorithms, create UDFs and design data cleaning pipelines can use Tableau to see and understand the raw data, to test their UDFs and to validate their workflows using effective visualization techniques. IT can use Tableau to visualize and share operational metrics, design KPI or monitoring dashboards, and accelerate the company's access to fresh, valuable Hadoop data through Tableau data extracts.

Prerequisites

You must have a Hortonworks Distribution including Apache Hadoop with Hive v0.5 or newer -- consult the following administration guide or work with your IT team to ensure your cluster is ready: http://www.tableausoftware.com/support/knowledge-base/administering-hado...

Additionally, you must have the Hortonworks Hive ODBC driver installed on each machine running Tableau Desktop or Tableau Server.

External References

We will describe some advanced features in this KB article, which may refer to the following external sources of information.

- Hortonworks has a wealth of information in their documentation: https://ccp.Hortonworks.com/display/DOC/Documentation
- The Apache Hive wiki provides a language manual and covers many important aspects of linking Hive to the data in your Hadoop cluster which we will rely on in this article: https://cwiki.apache.org/confluence/display/Hive/LanguageManual

Hortonworks We do Hadoop.

Getting Started with Hive

Hive is a technology for working with data in your Hadoop cluster by using a mixture of traditional SQL expressions and advanced, Hadoop-specific data analysis and transformation operations. Tableau works with Hadoop via Hive to provide a great user experience that requires no programming.

The sections below describe how to get started with analyzing data in your Hadoop cluster using the Tableau connector for Hortonworks Data Platform.

Installing the Driver

You must install the Hortonworks Hive ODBC driver. If you have a prior version of the driver installed you will need to first uninstall it, since the driver installer does not support in-place upgrades.

Connecting to Hortonworks Data Platform

Next, open the dialog for the Hortonworks Data Platform connector. Fill in the name of the server and port that is running your Hive service on the Hadoop cluster. Administering the Hive components on the Hadoop cluster is covered in a separate Knowledge Base article: Administering Hadoop and Hive for Tableau Connectivity.

Select the schema that contains your data set, choose one or more tables, or create a connection based on a SQL statement.

Performing Basic Analysis

Once connected, building a visualization is no different than when working with traditional databases. Drag-and-drop fields on to the visual canvas, create calculations, filter your data, and publish your work to Tableau Server.

Hortonworks

3460 West Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com

Twitter: twitter.com/hortonworks
Facebook: facebook.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks

Working with Date/Time Data

Hive does not have native support for date/time as a data type, but it does have very rich support for operating on date/time data stored within strings. Simply change the data type of a string field to Date or Date/Time to work with pure date or date/time data stored in strings. Tableau will provide the standard user interfaces for visualizing and filtering date/time data, and Hive will construct the Map/Reduce jobs necessary to parse the string data as date/time to satisfy the queries Tableau generates.

Features Unique to the Tableau Connector for Hortonworks Data Platform

Tableau offers several capabilities for the Hive connector that are not present in most of the other connectors.

XML Processing

While many traditional databases provide XML support, the XML content must first be loaded into the database. Since Hive tables can be linked to a collection of XML files or document fragments stored in HDFS, Hadoop provides a much more flexible experience when performing analysis over XML content.

Tableau provides a number of functions for processing XML data which allow users to extract content, perform analysis or computation, and filter the XML data. These functions leverage XPath, a web standard utilized by Hive and described in more detail in the Hive XPath documentation.

Web and Text Processing

In addition to XPath operators, the Hive query language offers several ways to work with common web and text data. Tableau exposes these functions as formulas which you can use in calculated fields.

- JSON objects: GET_JSON_OBJECT retrieves data elements from strings containing JSON objects.
- URLs: Tableau offers PARSE_URL to extract the components of a URL such as the protocol type or the host name. Additionally, PARSE_URL_QUERY can retrieve the value associated with a given query key in a key/value parameter list.

- Text data: The regular expression find and replace functions in Hive are available to Tableau users for complex text processing.

The Hive documentation for these functions offers more detail:
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+UDF

On-the-Fly ETL

Custom SQL allows users the flexibility of using arbitrary queries for their connection, which allows complex join conditions, pre-filtering, pre-aggregation and more. Traditional databases rely heavily on optimizers, but they can struggle with complex Custom SQL and lead to unexpected performance degradation as a user builds visualizations. The batch-oriented nature of Hadoop allows it to handle layers of analytical queries on top of complex Custom SQL with only incremental increases to query time.

Because Custom SQL is a natural fit for the complex layers of data transformations seen in ETL, a Tableau connection to Hive based on Custom SQL is essentially on-the-fly ETL.

Initial SQL

The Tableau connector for Hortonworks Data Platform supports Initial SQL, which allows users to define a collection of SQL statements to perform immediately on connecting. A common use case is to set Hive and Hadoop configuration variables for a given connection from Tableau to tune performance characteristics, which is covered in more detail below. Another important use case is to register the existence of custom UDFs as scripts, JAR files, etc. which reside on the Hadoop cluster. This allows developers and analysts to collaborate on developing custom data processing logic and quickly incorporating that into visualizations in Tableau.

Since initial SQL supports arbitrary Hive query statements, you can use Hive to accomplish a variety of interesting tasks upon connecting.

Custom Analysis with UDFs or Map/Reduce

Hive offers many additional UDFs which Tableau does not yet expose as formulas for use in calculated fields. However, Tableau does offer "Pass Through" functions for using UDFs, UDAFs (for aggregation) and arbitrary SQL expressions in the `SELECT` list. For example, to determine the co-variance between two fields 'f1' and 'f2', the following Tableau calculated field takes advantage of a UDAF in Hive:

```
RAWSQLAGG_REAL("covar_pop(%1, %2)", [f1], [f2])
```

Similarly, Tableau allows you to take advantage of custom UDFs and UDAFs built by the Hadoop community or by your own development team. Often these are built as JAR files which Hadoop can easily copy across the cluster to support distributed computation. To take advantage of JARs or scripts, simply inform Hive of the location of these files on disk and Hive will take care of the rest. You can do this with Initial SQL – which is described in the section above – with one or more SQL statements separated by semicolons as seen in the following example:

```
add JAR /usr/lib/hive/lib/hive-contrib-0.7.1-cdh3u1.jar;
add FILE /mnt/hive_backlink_mapper.py;
```

For more information consult the Hive language manual section on CLI.

For advanced users, Hive supports explicit control over how to perform the Map and Reduce operations. While Tableau allows users to perform sophisticated analysis without having to learn the Hive query language, experts in Hive and Hadoop can take full advantage of their knowledge within Tableau. Using Custom SQL a Tableau user can define arbitrary Hive query expressions, including the `MAP`, `REDUCE` and `TRANSFORM` operators described in the Hive language manual for Transform. As with custom UDFs, using custom transform scripts may require you to register the location of those scripts using Initial SQL.

Here is an interesting example of using custom scripts and explicit Map/Reduce transforms in the following blog post:
http://www.Hortonworks.com/blog/2009/09/grouping-related-trends-with-hadoop...

Hortonworks
3460 West Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com

Twitter: twitter.com/hortonworks
Facebook: facebook.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks

Designing for Performance

There are a number of techniques available for improving the performance of visualizations and dashboards built from data stored in your Hadoop cluster. While Hadoop is a batch-oriented system, the suggestions below can reduce latency through workload tuning, optimization hints or the Tableau Data Engine.

Custom SQL for Limiting the Data Set Size

As described earlier, Custom SQL allows complex SQL expressions as the basis for a connection in Tableau. By using a `LIMIT` clause in the Custom SQL, a user can reduce the data set size to speed up the task of exploring a new data set and building initial visualizations. Later, this `LIMIT` clause can be removed to support live queries on the entire data set.

It is easy to get started with Custom SQL for this purpose. If your connection is a single- or multi-table connection, you can switch it to a Custom SQL connection and have the connection dialog automatically populate the Custom SQL expression. As the last line in the Custom SQL, add `LIMIT 10000` to work with only the first 10,000 records.

Creating Extracts

The Tableau Data Engine is a powerful accelerator for working with large amounts of data, and supports ad-hoc analysis with low latency. While it is not built for the same scale that Hadoop is, the Tableau Data Engine can handle wide data sets with many fields and hundreds of millions of rows.

Creating an extract in Tableau provides opportunities to accelerate analysis of your data by condensing massive data to a much smaller data set containing the most relevant characteristics. Below are the notable features of the Tableau dialog for creating extracts:

- Hide unused fields. Ignore fields which have been hidden from the Tableau "Data Window", so that the extract is compact and concise.
- Aggregate visible dimensions. Create the extract having pre-aggregated the data to a coarse-grained view. While Hadoop is great for storing each fine-grained data point of interest, a broader view of the data can yield much of the same insight with far less computational cost.

Hortonworks

3460 West Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com

Twitter: twitter.com/hortonworks
Facebook: facebook.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks

- Roll up dates. Hadoop date/time data is a specific example of fine-grained data which may be better served if rolled up to coarser-grained timelines – for example, tracking events per hour instead of per millisecond.
- Define filters. Create a filter to keep only the data of interest – for example, if you are working with archival data but are only interested in recent records.

Advanced Performance Techniques

Below are some performance techniques which require a deeper understanding of Hive.

While Hive has some degree of support for traditional database technologies such as a query optimizer and indexes, Hive also offers some unique approaches to improving query performance.

Partitioned Fields as Filters

A table in Hive can define a partitioning field which will separate the records on disk into partitions based on the value of the field. When a query contains a filter on that partition, Hive is able to quickly isolate the subset of data blocks required to satisfy the query. Creating filters in Tableau on one or more partitioning fields can greatly reduce the query execution time.

One known limitation of this Hive query optimization is that the partition filter must exactly match the data in the partition field. For example, if a string field contains dates, you cannot filter on `YEAR([date_field])=2011`. Instead, consider expressing the filter in terms of raw data values, e.g.: `[date_field] >= '2011-01-01'`. More broadly, you cannot leverage partition filtering based on calculations or expressions derived from the field in question – you must filter the field directly with literal values.

Clustered Fields as Grouping Fields and Join Keys

Fields are clustered – sometimes referred to as bucketing – can dictate how the data in the table is separated on disk. One or more fields are defined as the clustering fields for a table, and their combined fingerprint ensures that all rows with the same content for the clustered fields are kept in close proximity within the data blocks.

Hortonworks
3460 West Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com

Twitter: twitter.com/hortonworks
Facebook: facebook.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks

This fingerprint is known as a hash, and improves query performance in two ways. First, computing aggregates across the clustered fields can take advantage of an early stage in the Map/Reduce pipeline known as the Combiner, which can reduce network traffic by sending less data to each Reduce. This is most effective when you are using the clustered fields in your `GROUP BY` clause, which you will find are associated with the discrete (blue) fields in Tableau (typically dimensions).

The other way that the hashing of clustered fields can improve performance is when working with joins. The join optimization known as a hash join allows Hadoop to quickly fuse together two data sets based on the precomputed hash value, provided that the join keys are also the clustered fields. Since the clustered fields ensure that the data blocks are organized bashed on the hash values, a hash join becomes far more efficient for disk I/O and network bandwidth because it can operate on large, co-located blocks of data.

Initial SQL

As discussed previously, Initial SQL provides open-ended possibilities for setting configuration parameters and performing work immediately upon establishing a connection. This section will discuss in more detail how Initial SQL can be used for advanced performance tuning. It is by no means comprehensive, since there are numerous performance tuning options which may vary in utility by the size of your cluster and the type and size of your data.

This first tuning example uses Initial SQL to force more parallelism for the jobs generated for Tableau analysis with that data source. By default, the parallelism is dictated by the size of the data set and the default block size of 64 MB. A data set with only 128 MB of data will only engage two map tasks at the start of any query over that data. For data sets which require computationally intensive analysis tasks, you can force a higher degree of parallelism by lowering the threshold of data set size required for a single unit of work. The following setting uses a split size of 1 MB, which could potentially increase the parallel throughput by 64x:

```
set mapred.max.split.size=1000000;
```

The next example extends our discussion of using clustered fields (aka bucketed fields) to improve join performance. The optimization is turned off by default for many versions of Hive, so we simply need to enable the optimization with the following settings. Note that the second setting takes advantage of clustered fields which are also sorted.

```
set hive.optimize.bucketmapjoin=true;
set hive.optimize.bucketmapjoin.sortedmerge=true;
```

This last example shows how configuration settings are sometimes sensitive to the shape of your data. When working with data, which has a highly uneven distribution – for example, web traffic by referrer – the nature of Map/Reduce can lead to tremendous data skew where a small number of compute nodes must handle the bulk of the computation. The following setting informs Hive that the data may be skewed and Hive should take a different approach formulating Map/Reduce jobs. Note that this setting may reduce performance for data which is not heavily skewed.

```
set hive.groupby.skewindata=true;
```

Known Limitations

Below are some of the notable areas where Hive and Hadoop differ from traditional databases.

High Latency

Hadoop is a batch-oriented system and is not yet capable of answering simple queries with very quick turnaround. This can make it difficult to explore a new data set or experiment with calculated fields, however some of the earlier performance suggestions can help a great deal.

Query Progress and Cancellation

Cancelling an operation in Hadoop is not straightforward, especially when working from a machine which is not a part of the cluster. Hive is unable to offer a mechanism for cancelling queries, so the queries, which Tableau issues, can only be "abandoned". You can continue your work in Tableau after abandoning a query, but the query will continue to run on the cluster and consume resources.

Additionally the progress estimator for Map/Reduce jobs in Hadoop is simplistic, and often produces inaccurate estimates. Once more, Hive is unable to present this information to Tableau to help users determine how to budget their time as they perform analysis.

Hortonworks

3460 West Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com

Twitter: twitter.com/hortonworks
Facebook: facebook.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks

Date/time processing

While Hive does offer substantial functionality for operating on string data which represents a date/time, it does not yet express date/time as a native data type. This requires user intervention by informing Tableau which fields contain date/time data. Additionally, the string data operations for date/time data do not support the complete SQL standard, especially those involving calendar operations like adding a month to an existing date.

Authentication

The Hortonworks Hive ODBC driver does not yet expose authentication operations, and the Hive authentication model and data security model are incomplete. Tableau offers functionality for exactly this case – User Filters in Tableau allow an analyst to express how the data in each visualization should be restricted before publishing them to Tableau Server.

Hortonworks
3460 West Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com

Twitter: twitter.com/hortonworks
Facebook: facebook.com/hortonworks
LinkedIn: linkedin.com/company/hortonworks