>Big Data, Small Scale

>**Raphael Klebanov**, *Customer Experience at WhereScape USA*
>**Moshe A. Klebanov**, *Cornell University, College of Engineering*

>Big Data Definitions

First coined in the **1990s ***, *Big Data* refers broadly to the growing, diversifying, and evolving data sets in many businesses and research applications. The role and form of Big Data varies considerably depending on the business and application; it is not a particular format and rigorous rulebook that pilots the success of Big Data analytics, but the philosophy of flexibility and interoperability that promotes its greater prominence. Nevertheless, here are some useful definitions:

"… Data sets with sizes beyond ability of commonly used tools to capture, integrate, manage, process within reasonable amount of time … [a] constantly moving target. Eventually it will just be data!"
-- Claudia Imhoff, Ph.D. *Intelligent Solutions, Inc*.

"… High volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization."
-- Laney, Douglas. *The Importance of 'Big Data': A Definition*. Gartner. 21 June 2012.

* Steve Lohr. *The Origins of 'Big Data': An Etymological Detective Story.* Feb 1, 2013

>Big Data in Small to Medium Businesses

We hear a lot about eye-popping implementations of a Big Data in the "Big Boys'" environment:

- EBay supports 90PB Hadoop clusters for search, consumer recommendations, and merchandising…
- Facebook handles 50 billion photos from its user base…
- Google was handling roughly 100 billion searches per month…
- And my favorite: NSA Data Center in Bluffdale, UT (according to Der Spiegel), has enough capacity Big Data to store a Yottabyte (1 trillion TB!) of data… "large enough to store all the electronic communications of all of humanity for the next 100 years"

Impressive? Heck yeah!
But Big Data is not limited to those gargantuan projects; it can happily run in small and medium business environments.

Here are three characteristics small and medium businesses can take advantage with a Big Data solution:

1. Simplicity

Big Data solutions for small and medium businesses are relatively easy to implement and use; it takes only a few days—two weeks tops—for a company to deploy and start using it. Numerous solutions provided by Apache, Hortonworks, and few other vendors allow the Hadoop system et al. be integrated into a small environment seamlessly with the existing systems already in use. Such a setup is fairly simple and attainable without the expensive customization and fancy work skills.

The use case provided in this article illustrates how a small but functioning Big Data system can be built cheaply and with limited software/hardware skills.

2. Adaptability

Big Data solutions employed at large enterprises are usually a win-or-lose scheme, requiring customers to amend existing systems, thus imposing a hefty cost and time burden on IT departments. This is unacceptable for a small or medium business; Small- and medium- sized businesses simply operate differently than their larger counterparts.

If, let's say, a sales department of a small company needs an analytical system, it oftentimes acts self-reliantly of the IT department, developing its business needs, cost analysis, and system requirements by exploring and choosing the best solution to meet its needs. On the other hand, IT departments will follow the business' lead to employ the technologies that best suit their needs.

As a result, small businesses regularly deploy a plethora of solutions throughout the organization that leads to an environment with many different types of data. Big Data technologies allow the different business units of the small- or medium- sized company to choose only the capabilities they need and leverage the solutions and systems already in place in one pre-integrated package.

3. Cost

Lastly, but arguably the most importantly, a Big Data solution for small businesses is cheap. Customers should be able to pay for only the functionality they need, and the licensing strategy should allow them to start small and scale up as the need for analytics increases.

In our example, the cost of a small Hadoop system totals to less than $200, with no licensing fees (due to predominant use of open source software). This approach is particularly useful for a fast-growing small or medium sized business, where it is crucial for the cost and abilities of software investments to line up with the rate of growth of the operation and ability of the company to find the right IT solution. This solution should focus on the specific requirements of the small business and be able to recognize the benefits from the opportunity Big Data provides.

>WhereScape Support of the Big Data Solution

WhereScape provides a productivity solution that does not require a lot of training and includes self-service capabilities to cater to a larger audience of analysts and business users. Those practitioners can easily integrate the Big Data components into the other, more traditional sources and data warehouse subject areas.

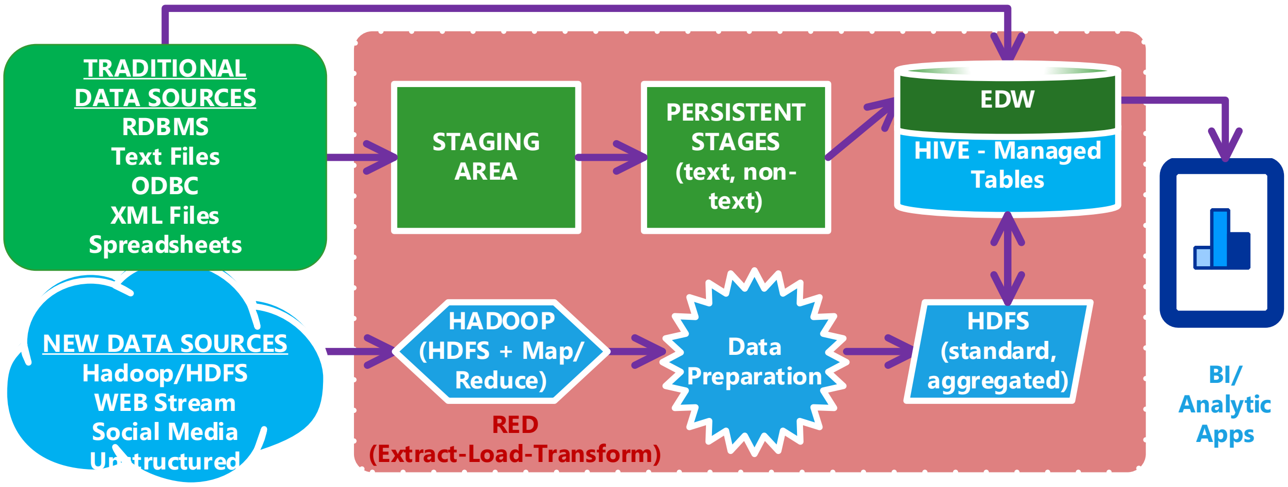


Figure 1 High-level Data Flow in RED-driven Solution

WhereScape supports the following functionality in relation to the Big Data subject:

1. Extending Hive as a target
2. DB-specific loaders for Hadoop to RDBMS (Oracle/Teradata/Greenplum)
3. Moving of data between RDBMS and HDFS or Hive.
4. Processing of ELT on Hive for standard object types – Load, Stages, Dims, Facts, Aggregates, etc.
5. New "File" object = 'HADOOP'. Load tables created from HADOOP file directly.
6. Creating Models from Hadoop and HDFS connections (in 3D)

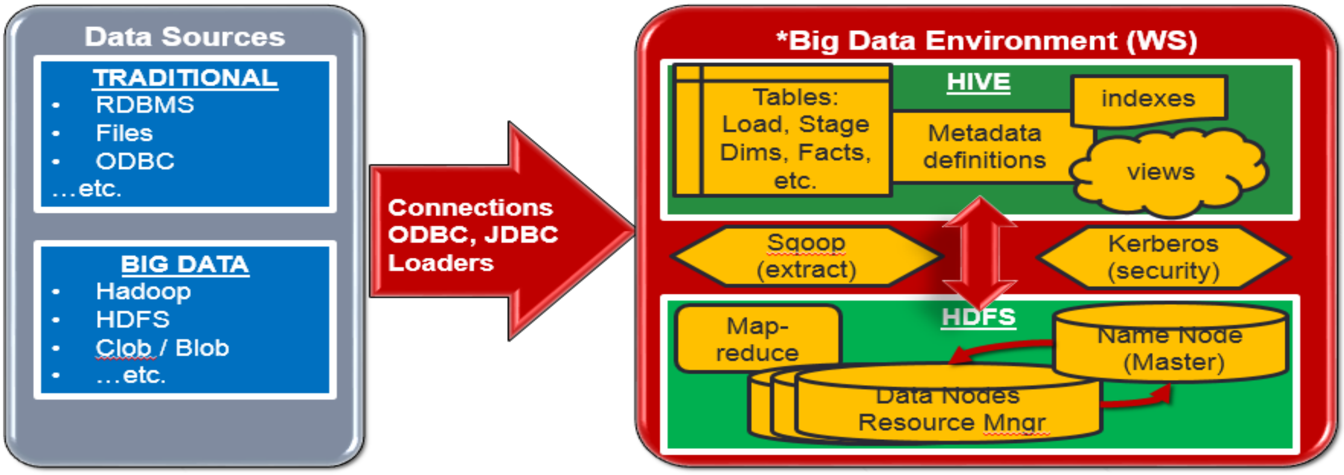


*Figure 2 New Big Data Warehouse from WhereScape. *WIP; subject to change*

>Use Case Setting-up Small Scale Hadoop Cluster

1. Data Processing Overview

a. Both relational and flat-file data stored in the HDFS/Hadoop distributed file system
b. Data in the HDFS is replicated and distributed over 3 slave nodes to prevent loss of data and ensure failover processing in case of system crash or network failure.
c. HDFS has one Name Node (master) node that store metadata; data is partitioned and stored redundancy across slave nodes.
d. All of the nodes reside on the Dell Precision T7500 Workstation virtualizer (see further)
e. Each node offers local storage and data computation

2. Data Querying Overview

a. Query is submitted from the Application interface (JDBC through Hive (SQL) or directly via the Java Map/Reduce) to the master node.
b. Master node uses a Map process to assign the I/O and processing tasks to the slave nodes
c. The tasks are executed in parallel on each node of the cluster against the node's local dataset
d. The slave nodes complete their tasks and return results to the master node
e. If necessary, the master node issues a Reduce task to the slave nodes, to aggregate the Map-processed data.
f. The tasks are executed in parallel on each node of the cluster against chunks of the resulting dataset. The master node pulls together the result and sends it back for analytics.
g. This can be an iterative process under some circumstances, when there are ≥2 (# relational joins + #group by). But that might be a little out-of-the-scope of the article.

3. The Hardware

The Hadoop cluster was configured across a set of four virtual machines on a dedicated Dell Precision T7500 Workstation Chassis:

- Intel Xeon CPU Quad Core E5506 2.13GHz 4.8 GT/s 4MB Cache - SLBF8 (x2).
- 2GB PC-8500R Dual Ranked DDR3 1066MHz Registered ECC RDIMM Memory(x4 for a total of 8GB)[1]
- 250GB SATA 7.2K RPM 1.5 GB/s 3.5" Hard Drive (x4) – configured to RAID 1[2].
- SAS/SATA 3.0 GB/s controller

This server cost me only $198 (including delivery) from Stellar Technology, Inc.

4. Initial Setup

- A minimal install of CentOS 7 (the latest version)
- QEMU-KVM used for virtualizing the Hadoop servers[3], with Virt-manager used as the GUI interface
- 4 virtual machines, a Name Node, a Resource Manager, and 2 Data Nodes were configured as part of the cluster, each with 2 GB of RAM, 2 virtual cores, 50GB of thickly-provisioned storage, and a bridge to the local network.
- Hadoop v2.6.0 running on Java 1.7.0 (on each machine)
- A static IP address for each virtual machine, for an easy network setup

5. Introducing Hadoop[4]

hadoop-1 (IP 192.168.1.101): The NameNode

- Acts as the “master” node of the HDFS cluster
- Stores the metadata for files across the cluster
- Keeps track of the DataNode servers, where the data lives
- Executes read+write operations for the distributed filesystem

hadoop-2 (IP 192.168.1.102): The ResourceManager and JobHistory node

- Issues MapReduce commands, the “software framework” that enables parallel processing of both flat-file and relational data (the latter backed by Apache Hive)
- Maintains configurations for memory and CPU resources allocated across the cluster
- Allows the application to track long-running operations, such as overnight batch jobs.

hadoop-3 (IP 192.168.1.103): DataNode and NodeManager
hadoop-4 (IP 192.168.1.104): DataNode and NodeManager

- Store (replicated) data in the Hadoop filesystem
- Accept requests from the NameNode for filesystem operations.
- Perform individual Map and Reduce operations, performing operations on local data whenever possible to improve throughput and decrease the amount of data moving over the network.
- Are specified in a cluster-wide conf/slaves file

```
# conf/slaves
192.168.1.103
192.168.1.104
                    (more nodes can be added dynamically, without needing to restart the entire cluster)
```

[1] Expandable to 32GB

[2] RAID 0 also supported

[3] When running a virtualized environment of any kind, there are many options to consider. Qemu-kvm, an open-source kernel-based VM hypervisor was chosen for this project due to its widespread open-source use, free license, and my personal experience with this tool. Other solutions such as VMWare® vSphere can be used, and the Hadoop cluster can of course run on bare-metal hardware

[4] Hadoop is designed to be very configurable, for a wide variety of setups and environments. Visit the Apache documentation for documentation regarding configuration entries.

6. Hive Setup

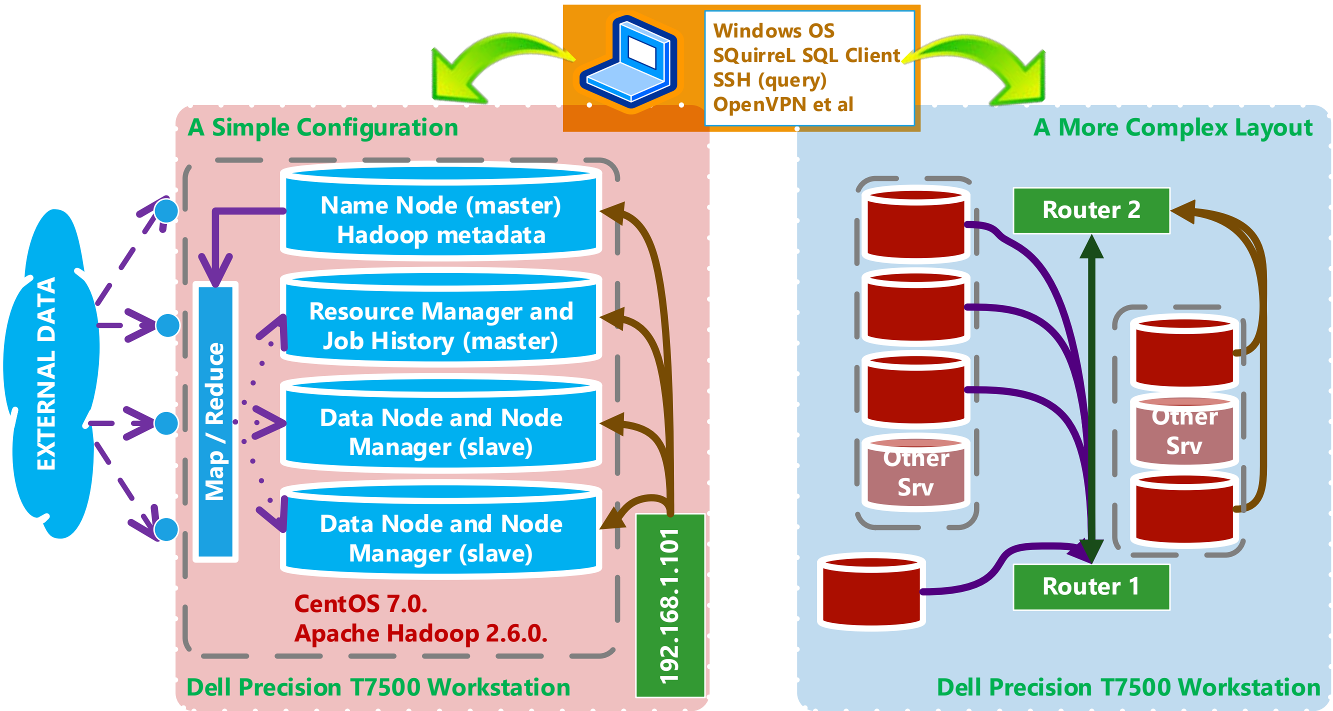


Figure 3 Simplified setting for Hadoop Cluster

According to Apache[5], "Hive provides a mechanism to project structure onto this data and query the data using a SQL-like language," utilizing the **Map-Reduce** framework to carry out CRUD and bulk data-processing operations. It is an abstraction layer that operates entirely above the Hadoop/HDFS infrastructure, softening the learning curve for SQL-literate, Hadoop-ready teams.
Since its job is to issue—rather than execute—SQL plans as **Map-Reduce** operations, Hive needs to be installed only on the *Name Node*.

1. Install Hive (use ver 0.14.0) from the Apache website
2. Create directories /tmp and /user/hive/warehouse from within HDFS
3. Start $HIVE_HOME/bin/hive
4. Using the command line tool beeline (or a GUI utility such as SQuirreL SQL Client), connect to the Hive daemon using the JDBC connection string

jdbc:hive2://192.168.1.101:10000/default – the IP address of the Name Node

7. Setting up the Database

Unlike the vast majority of relational databases —which operate on records stored in proprietary (typically binary) formats—Apache Hive is designed to be flexible, supporting a number of row-file formats. Instead of migrating your data from source to database, Hive understands data **as it is stored** in HDFS. Consider the following data set, representing stock trades in 2000-2001, stored in a CSV (comma-separated values) file on a non-HDFS machine. [6]

exchange	stock_symbol	date	stock_price_open	stock_price_high	stock_price_low	stock_price_close	stock_volume	stock_price_adj_close
NYSE	ASP	12/31/2001	12.55	12.8	12.42	12.8	11300	6.91
NYSE	ASP	12/28/2001	12.5	12.55	12.42	12.55	4800	6.78
NYSE	ASP	12/27/2001	12.59	12.59	12.5	12.57	5400	6.79
NYSE	ASP	12/26/2001	12.45	12.6	12.45	12.55	5400	6.78
				⋮				

8. Load Data into Hadoop

To load this data into Hive, first load the file into Hadoop using the 'hdfs' command line utility

```
bin/hdfs dfs –copyFromLocal NYSE-2000-2001.csv /tmp/NYSE-2000-2001.csv
```

[5] https://hive.apache.org/
[6] Source:

9. Create Table

Next, create the table (from a JDBC client such as beeline), using the usual CREATE TABLE command, specifying the data types and row structure (in this case, CSV—delimited by comma)

```
CREATE TABLE NYSE_2000_2001 (
     `exchange` VARCHAR(10),
     `stock_symbol` VARCHAR(10),
     `date` CHAR(10),
     `stock_price_open` DECIMAL(10, 2),
     `stock_price_high` DECIMAL(10, 2),
     `stock_price_low` DECIMAL(10, 2),
     `stock_price_close` DECIMAL(10, 2),
     `stock_volume` INT,
     `stock_price_adj_close DECIMAL(10, 2)
) ROW FORMAT DELIMITED FIELDS TERMINATED BY ",";
```

10. Load Data into Hive

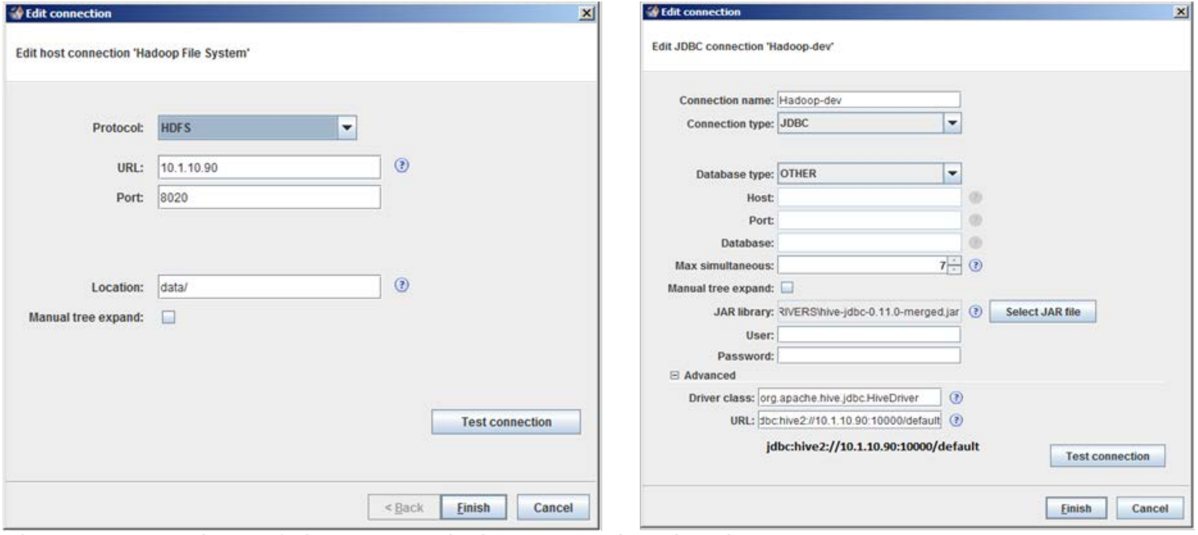
To load data into Hive, issue the following SQL command. This operation will move the source file into the location corresponding to the Hive table NYSE_2000_2001.

```
LOAD DATA INPATH '/tmp/NYSE-2000-2001.csv'
     OVERWRITE INTO TABLE NYSE_2000_2001
```

NYSE_2000_2001 is now populated with data and ready to index, query, and update using SELECT/INSERT/UPDATE/DELETE operations.

11. WhereScape 3D:

- Connect to Hadoop and Hive using JDBC (some configuration might require depending Java settings
- Discover structures of Hive tables and HDFS files
- Profile data in Hadoop and HDFS
- Convert to any of the DW models
- Integrate Big Data sources with others into 3D model



Edit connection

Edit host connection 'Hadoop File System'

Protocol: HDFS

URL: 10.1.10.90

Port: 8020

Location: data/

Manual tree expand: □

Edit connection

Edit JDBC connection 'Hadoop-dev'

Connection name: Hadoop-dev

Connection type: JDBC

Database type: OTHER

Host:

Port:

Database:

Max simultaneous: 7

Manual tree expand: □

JAR library: RIVERS\hive-jdbc-0.11.0-merged.jar

User:

Password:

⊟ Advanced

Driver class: org.apache.hive.jdbc.HiveDriver

URL: jdbc:hive2://10.1.10.90:10000/default

jdbc:hive2://10.1.10.90:10000/default

Figure 4 Screenshots of the HDFS and Hive connections in WhereScape 3D

> ***Note:*** *WhereScape 3D requires the JAR library to reside in one file; however, beeline (Hive's default JDBC command-line client) operates with a collection of Hadoop and Hive JAR files that together constitute the Hive's JDBC driver. Thus, it is necessary to combine these JAR files into a "merged one" that WhereScape 3D can use. Fortunately, Apache Ant makes this an easy task.*
>
> *1. Copy lib/hive-jdbc-*-standalone.jar from the Hive install and share/hadoop/common/hadoop-common-*.jar from the Hadoop install into a "lib" directory on the local machine.*
> *2. Create a build.xml file, parallel to the "lib" directory, with the following contents*

```
<!-- build.xml -->
<?xml version="1.0" standalone="no"?>
<project name="test">
  <target name="combine-jars">
```

```
    <jar destfile="hive-jdbc-0.11.0-merged.jar">
      <restrict>
        <not>
          <name name="META-INF/*.SF"/>
        </not>
        <archives>
          <zips>
            <fileset dir="lib" includes="*.jar"/>
          </zips>
        </archives>
      </restrict>
    </jar>
  </target>
</project>
```

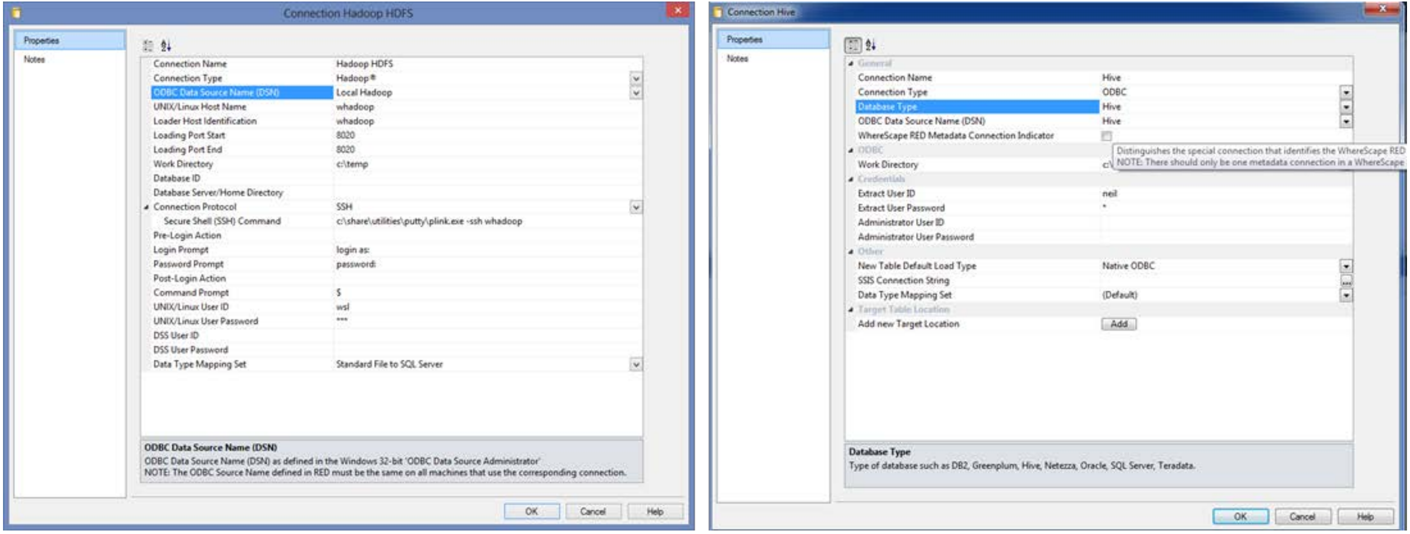
3. Execute the following command from the directory where build.xml resides:

ant combine-jars

hive-jdbc-0.11.0-merged.jar should now be in your current directory. This file can be used by WhereScape 3D—as well as any other JDBC-capable software—to access Hive.

12. WhereScape RED:

- Support of MPP, MSPDW and Ext columns(>512); “big tables”
- Connect to Hadoop HDFS and Hive using ODBC or Hadoop Connector®
- Build a DW iteration in any of the supported target platforms (SQL Server, Oracle, DB2, Teradata, Greenplum, Natezza)
- Extend RED to support Hadoop HDFS and Hive environments as “target”
- ELT processing on both EDW and Hive in single tool.
- Both-way transfer of data between EDW and HDFS or Hive.



Connection Name	Hadoop HDFS
Connection Type	Hadoop®
ODBC Data Source Name (DSN)	Local Hadoop
UNIX/Linux Host Name	whadoop
Loader Host Identification	whadoop
Loading Port Start	8020
Loading Port End	8020
Work Directory	c:\temp
Database ID	
Database Server/Home Directory	
Connection Protocol	SSH
Secure Shell (SSH) Command	c:\share\utilities\putty\plink.exe -ssh whadoop
Pre-Login Action	
Login Prompt	login as:
Password Prompt	password:
Post-Login Action	
Command Prompt	$
UNIX/Linux User ID	wsl
UNIX/Linux User Password	***
DSS User ID	
DSS User Password	
Data Type Mapping Set	Standard File to SQL Server

ODBC Data Source Name (DSN)
ODBC Data Source Name (DSN) as defined in the Windows 32-bit 'ODBC Data Source Administrator'
NOTE: The ODBC Source Name defined in RED must be the same on all machines that use the corresponding connection.

Connection Name	Hive
Connection Type	ODBC
Database Type	Hive
ODBC Data Source Name (DSN)	Hive
WhereScape RED Metadata Connection Indicator	□
Work Directory	c:\
Extract User ID	neil
Extract User Password	*
Administrator User ID	
Administrator User Password	
New Table Default Load Type	Native ODBC
SSIS Connection String	
Data Type Mapping Set	(Default)
Add new Target Location	Add

Distinguishes the special connection that identifies the WhereScape RED
NOTE: There should only be one metadata connection in a WhereScape

Database Type
Type of database such as DB2, Greenplum, Hive, Netezza, Oracle, SQL Server, Teradata.

Figure 5 Screenshots of the Hadoop and HDFS connections in RED

>Conclusion

A common sentiment of a Big Data has been that it is too big, too expensive, too NSA-ish for small and medium businesses because it requires colossal processing power to crunch all those peta-/exa-/zetta-bytes of “stuff”.

But it is becoming more and more clear that this is not always the case! Processing power is cheaper and more available than ever, letting small and medium-sized businesses harness Big Data and use it to work for small operations with limited IT resources and lack of time and money.

Although Big Data has long been—and continues— to be in the sphere of large-scale enterprises, small and medium businesses are increasingly able to gain benefits of Big Data technologies: its simplicity, high adaptability, and decreasing cost.

With time being, even tiny operations will be able to make use of Big Data’s big possibilities.