Data Warehouse Optimization
Embedding Hadoop in Data Warehouse Environments

A Whitepaper

Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

September 2013

Sponsored by talend
# Table of Contents

1. **Management Summary**
2. **Data Warehouses are Growing**
3. **Optimizing Data Warehouse Architectures**
4. **Hadoop in a Nutshell**
5. **Embedding Hadoop in Data Warehouse Architectures**
   5.1 **Approach 1: Using Hadoop to Store Cold Data**
   5.2 **Approach 2: Using Hadoop as Staging Area**
   5.3 **Approach 3: Using Hadoop as an Extra Data Warehouse Database**
   5.4 **Approach 4: Using Hadoop for ETL Processing**
   5.5 **Summary**
6. **ETL Requirements for Supporting Hadoop**
7. **Talend’s Support for ETL on Hadoop**
   - About the Author Rick F. van der Lans
   - About Talend Inc.
1 Management Summary

Data warehouses have grown to become the largest databases of an organization. This growth is not going to stop in the foreseeable future. Data warehouse environments are growing with respect to their number of users, the amount of stored data, the number of data sources, and the complexity of reporting and analytical demands. When data warehouse databases grow significantly, performance and concurrency problems can arise, storage and processing costs may become unacceptable, a reporting backlog may appear, stability can go down, and so on. In such situations, organizations have to optimize their data warehouse.

The last few years, many new data storage technologies have been introduced. All of them were designed for storing and managing massive amounts of data at relatively low costs. Of all these technologies Hadoop is, without any question, the most popular. Hadoop can be used in data warehouse environments to reduce storage and processing costs, and to improve the performance of reporting and analysis.

This whitepaper describes the following four approaches for using Hadoop to optimize a data warehouse environment:

- Using Hadoop to store cold data
- Using Hadoop as staging area
- Using Hadoop as an extra data warehouse database
- Using Hadoop for ETL processing

When Hadoop is used, ETL tools are needed to extract data from and load data into Hadoop. To do this efficiently, they have to support at least the following features:

- Pushdown of ETL logic into Hadoop to improve performance
- Data scalability for handling large quantities of data (big data)
- Handling of non-SQL concepts
- Extracting data from and loading data into Hadoop
- Schema-on-read for handling multi-structured data

In addition, ETL tools should make it easy for developers to work with Hadoop, because Hadoop’s native API’s are detailed and complex. ETL tools should hide these technical details to improve productivity and ease maintenance, and to make adoption of Hadoop as smooth as possible.

The whitepaper ends with a section on how Talend’s ETL solution supports Hadoop with their unified integration platform. It also explains how Talend meets the requirements for supporting Hadoop.
Data Warehouses are Growing

There must have been a time when all data warehouses were small. If there ever was such a time, it must be a long time ago. Most data warehouses have grown to become the largest databases of an organization. This growth won’t stop in the foreseeable future. Data warehouse environments are growing and will continue to grow and not only with respect to the amount of stored data, but in several dimensions:

- **The number of users who need reporting and analytical capabilities keeps increasing.** Most data warehouses were originally designed to support strategic and tactical management. Nowadays, operational management, the workforce, and even external parties need access to the data collected in the data warehouse. Usually, these new user groups consist of large numbers of users which leads to more reports, queries, and analytical sessions. In addition, they may need data not yet available in the data warehouse.

- **The amount of data needed for reporting and analytical purposes keeps growing.** One trend that is responsible for this growth is big data. Whether big data is structured, unstructured, multi-structured, or semi-structured, it’s always a massive amount of data. To illustrate how successful the big data trend is, Gartner\(^1\) predicts that big data will drive $232 billion in spending through 2016, and Wikibon\(^2\) claims that in 2017 big data revenue will have grown to $47.8 billion.

- **The number of data sources from which data has to be extracted keeps growing.** For new forms of analytics and reporting, classic operational data is not always sufficient anymore. Internal data has to be enriched with external data. For example, internal customer sales data has to be enriched with socio-demographic data to get a better understanding of customer buying patterns. Big data plays a major role here as well. Organizations want to analyze what customers are “saying” about their products on social media networks. They want to combine social media data with internal data.

- **The data latency is decreasing.** There was a time when data warehouse users were satisfied when the data used for their reports was one week old. Today, users don’t accept a data latency of one week anymore, they want one hour, or maybe even five seconds or less. Especially user groups such as operational management and external parties want to have insight in the most current situation—yesterday’s data is worthless. Lowering the data latency may increase the size of the data warehouse database grow, because more detailed data and more points in time of data values have to be stored.

---


3 Optimizing Data Warehouse Architectures

Data Warehouses are Growing – When data warehouse databases expand, data latency decreases, reporting and analytical demands increase, performance and concurrency problems can arise, storage and processing costs may become unacceptable, a reporting backlog may appear, stability can go down, and so on. In such situations, organizations have to optimize their data warehouse. They have to find a way to solve these problems.

In general, available optimization techniques range from minor techniques such as tuning of an index to optimize direct access to data, all the way up to techniques with which the entire environment is re-architected. Optimization techniques can be classified as follows:

- **Architectural Optimization**: A data warehouse environment can be optimized by making improvements on the architectural level. For example, a staging area is removed to lower the data latency level, or data is offloaded to an archival data store.

- **Tool Optimization**: A data warehouse environment can be improved by tuning and optimizing individual tools. For example, the performance of queries on database servers can be tuned by keeping more data in memory, or by adding more indexes. Also, server machines can be tuned to optimize processing. These forms of optimization are usually done by DBAs and developers.

- **Design Optimization**: Design optimization implies that changes are made to the initial design. For example, normalized data structures in a data mart are replaced by star schemas, or the style of data copying is changed from more batch-like to more real-time.

- **Technology Replacement**: An optimization technique may also be to deploy new technology. For example, to improve performance or scalability a more specialized database server, a more advanced ETL tool, or an in-memory analytical tool may be acquired.

The Coming of Hadoop – The last years, many powerful new database technologies have become available. Especially, for supporting big data numerous new systems have become available. All of them are designed for storing and managing massive amounts of data at relatively low costs. Of all these technologies Hadoop is the most popular.

Conclusion, Hadoop’s ability to handle large data stores at low costs makes it an optimization technique for reducing the storage and processing costs and for improving the performance of reporting and analysis. But how and where? This whitepaper addresses these questions. It describes four different approaches for embedding Hadoop in data warehouse environments. But first Hadoop and its modules are described.
4 Hadoop in a Nutshell

Introduction to the Hadoop Framework – Currently, a very popular technology for data storage is Hadoop. Hadoop is a software framework designed for supporting data-intensive applications. It’s more than suitable for storing massive amounts of data (even if it’s many petabytes of data), and designed for applications in which a continuous stream of new, incoming data has to be processed, stored, and managed, it excels at processing complex forms of analytics fast, and it’s relatively low-cost.

Typical examples of applications that can benefit from Hadoop are click stream and sensor-driven applications (for example, RFID-based) that continuously generate enormous amounts of records. Some of these applications literally generate thousands of records per second. All this data needs to be stored for future use, leading to a massive amount of data storage. Hadoop has been designed to support this type of application. In other words, it has been designed for the world of big data.

What’s special about Hadoop is its relatively low price. Its data storage costs and license costs are low compared to those of traditional database servers. It’s this combination of low-cost and massive data storage that makes it a technology fit for data warehouse environments.

The Modules of Hadoop – Hadoop consists of a set of modules. Here the core modules are described briefly. For a more extensive description, we refer to Tom White’s book3 on Hadoop.

- **HDFS**: The foundation of Hadoop is formed by HDFS (Hadoop Distributed File System). This file system is responsible for storing and retrieving data. It’s designed and optimized to deal with large amounts of incoming data per second and for storing and managing enormous amounts of data up to the petabytes. The key aspect of HDFS is that it can distribute data over a large number of disks and can exploit an MPP architecture. HDFS supports a well-designed programming interface that can be used by any application. HDFS is the only mandatory module of Hadoop, the others are all optional.

- **MapReduce**: The module called MapReduce allows that data inserts and queries can be distributed efficiently over hundreds of nodes. Important to note is that the programming interface of Hadoop’s MapReduce is very technical and requires a deep understanding of the internal workings. To read data from a file in HDFS, MapReduce always scans the entire file, because no indexes are available.

- **HBase**: The HBase module is designed for applications that need random, real-time, read/write access to data. It operates on top of HDFS.

- **Hive**: The module called Hive offers a SQL-like interface for querying data. It supports a dialect of SQL called HiveQL. HiveQL supports the more classic features of SQL, but some are missing. Internally, Hive translates SQL statements to MapReduce batch jobs to parallelize the processing.

• **Pig**: Next to Hive, developers can also use Pig for querying the data. The language supported by this module is called Pig Latin. Although Pig Latin supports a set of functions that slightly resembles the operators of SQL, such as group-by, join, and select. It’s a more technical language than HiveQL.

**The Batch-Oriented Nature of MapReduce** – MapReduce programs are executed as batch jobs that are scheduled and distributed over many nodes. In the background the process of starting and running all those jobs is monitored and managed. To distribute and manage all these jobs, requires additional processing. However, considering the amount of data commonly analyzed, this overhead of additional processing is probably acceptable for a non-interactive analytical environment. Note that some database servers would not even be able to query so much data, so this extra time for management is the penalty paid for being able to analyze big data with an adequate performance.

**The SQL-fication of Hadoop** – Hive was the only SQL interface on Hadoop, but lately, more and more modules have been released that offer SQL interfaces to Hadoop. For example, Cloudera has released Impala, HortonWorks has Stinger, and MapR Technologies has Drill. In addition, data virtualization vendors have also made their products available for Hadoop. Furthermore, some SQL database servers support access to Hadoop. Aster Database is one of them. There is clearly a growing demand for SQL-fication of Hadoop. Organizations want to have an easier to use and more productive interface to access Hadoop data than the HDFS or MapReduce interfaces.

**The Disadvantages of Hadoop** – As indicated, the Hadoop platform offers low-cost, high-volume data storage capabilities and analytical features. But no technology is free from disadvantages. Hadoop is no exception:

• To exploit the potential power of Hadoop, developers use interfaces such as the one supported by MapReduce. They write programs to load and query data using this interface. Unfortunately, this interface is very low-level and complex.

• Another disadvantage of Hadoop is that it can’t handle every type of query workload, and this is regardless of the modules used. Hadoop is very good at running complex analytical applications on all the available data. However, the environment is not suited for, for example, ad-hoc queries. A typical data mart workload won’t run fast on Hadoop. In addition, there is not a lot of technology on board to stabilize the query workload when a large number of users run reports simultaneously. Note that this disadvantage may disappear over time when Hadoop is extended with new modules.

## 5 Embedding Hadoop in Data Warehouse Architectures

At least four different approaches exist on how to use Hadoop in a data warehouse environment to make it cheaper, more scalable, or more efficient. This section describes these four approaches.

Note: In this section, the assumption is made that the architecture of a classic data warehouse resembles a chain or network of databases; see Figure 1. Data is pushed through this network
of databases using integration tools. In most situations all the databases are implemented using SQL databases.

![Figure 1](image)

**Figure 1** Most data warehouse environments resemble a chain or network of databases.

5.1 **Approach 1: Using Hadoop to Store Cold Data**

There may be many reasons why queries on data warehouses and data marts become slow. One reason could be that the tables being accessed have become huge. Imagine that a data warehouse currently holds 10 years of data. Maybe not all of that data is being used frequently. In many situations, the older the data becomes, the less it’s used. This allows data to be classified as cold, warm, or hot. Hot data is used almost every day, and cold data occasionally. Keeping cold data in the data warehouse slows down the majority of the queries and is expensive, because it’s stored within an expensive storage system.

In this situation, it might be useful to store cold data outside the SQL database and in Hadoop HDFS. This storage form is cheaper and data is still accessible. The architecture would look like Figure 2. Here, technically the data warehouse consists of two data databases, one that contains the warm and hot data and is implemented with a classic SQL database, and one that holds the cold data and uses Hadoop. The black arrow in Figure 2 indicates that periodically data that has turned cold is moved from the SQL database to the Hadoop data store. The data marts are filled with data coming from both data stores, and, if necessary, reports can access both data stores.

To summarize, this approach can save storage costs, but more importantly it can speed up queries on the SQL part of the data warehouse (less data), plus the architecture can handle larger data volumes by moving cold data to Hadoop.
5.2 Approach 2: Using Hadoop as Staging Area

The workload of a staging area is usually straightforward: new data comes in and is stored, next, the stored data is transformed and copied into a data warehouse database. Most staging areas are developed with SQL technology. Eventually, data that has been copied to the data warehouse can be deleted from the staging area. Usually, no reports are executed on a staging area.

The workload of a staging area fits Hadoop like a glove; see Figure 3. Hadoop is able to load large amounts of data fast, and it's great at extracting data using a batch-oriented approach. If data has to be heavily transformed before it's copied to the data warehouse, all that processing can be executed by MapReduce jobs.

Deploying Hadoop as staging area (instead of a SQL database) reduces data storage costs and allows for storing and processing more data.
5.3 Approach 3: Using Hadoop as an Extra Data Warehouse Database

With the first approach, the Hadoop data store does not contain tables that are not stored in the SQL database. It only contains tables that are also available in the SQL database, and then only if cold data can be identified in these tables. With this third approach, the two data stores contain different sets of tables (and thus different data); see Figure 4. In this solution, data is copied straight from production systems to Hadoop, some reports extract data from this Hadoop data store, and data is extracted for the data marts to support particular forms of reporting and analysis. In this approach, the data warehouse consists of two data stores: a more classic database and a Hadoop data store.

![Figure 4](image)

This architecture can be useful if some operational systems generate massive amounts of new data, and the SQL database used to create the staging area is not able to cope with this heavy workload or only at a very high price. This can happen for example in a factory with numerous sensors installed in the machines. Or, an application that monitors all the messages on social media networks that concerns the organization and its products and brands. By the way, it could even be that the new data is not even stored in a production system at all, but stored straight in Hadoop.

Hadoop is used in this architecture for an application area in which it excels: large database, heavy workload of incoming data, and periodic reports for extracting data.

5.4 Approach 4: Using Hadoop for ETL Processing

The sheer data volume in a particular data warehouse may be too much for a SQL database. It may become too costly and queries may be too slow. In this situation, the decision may be made to aggregate the data somewhat before it’s stored in the data warehouse database; see Figure 5. This shrinks the database size and speeds up the queries. In fact, besides doing aggregating data, other forms of processing may be applied as well.
However, if this solution is selected, a database must exist that contains all the detailed data (which is not the staging area) and some module must become responsible for aggregating and processing the data. Hadoop can be used as data store for all the detailed data and to let MapReduce do all the aggregations. In this architecture Hadoop contains the large data volumes—this is where data is pumped into first. Then, MapReduce jobs are run to efficiently aggregate and process that data and copy it to the data warehouse database. So, Hadoop is doing all the aggregations. In other words, it’s doing all the processing of data before it becomes available for reporting and analysis—Hadoop is used as an ETL engine.

This architecture may also be useful when so-called multi-structured data is stored and used for analysis and reporting. With multi-structured data, no schema is assigned to the data. It’s as if the data is stored in its raw format in which it was received. In this case, the schema must be derived when the data is read. This is called schema-on-read. (When stored data does have a schema, such as most data stored in SQL databases, it’s called schema-on-write.) Assigning a schema to data when it’s being read, requires heavy processing. Again, MapReduce jobs can do this in parallel and thus fast. This makes this architecture very relevant when multi-structured data must be processed by the data warehouse environment. The multi-structured data is stored in the Hadoop data store and MapReduce assigns a schema when the data must be copied to the data warehouse.

5.5 Summary

In this section four approaches are described in which Hadoop can be used to optimize a data warehouse environment. Optimization in this situation implies lower costs, improved performance, and improved data scalability. Evidently, there are more ways in which Hadoop can be used for optimizing data warehouse environments, but these are the more likely ones and the ones being used today. Note that multiple approaches can be used in one and the same environment.
6  ETL Requirements for Supporting Hadoop

Hadoop and the Need for ETL – For each of the approaches described in the previous section, ETL functionality is needed in combination with Hadoop. In the first approach ETL is used to copy data from a SQL-based data warehouse to Hadoop and onwards to data marts, in the second approach ETL is used for copying data from production databases to Hadoop and from Hadoop to the SQL-based data warehouse, in approach three ETL copies data from production systems to Hadoop and from Hadoop to the data marts, and in approach four, ETL extracts data from a SQL-based staging area and loads it into Hadoop and copies it to a SQL-based data warehouse.

But what does this mean for ETL tools? Can any ETL tool be used for loading data into and extracting data from Hadoop? Because JDBC/SQL interfaces are available for Hadoop and every ETL tool supports such interfaces, the answer is yes. However, the real question is, can ETL tools access Hadoop efficiently? In general, for reporting and analysis purposes JDBC/SQL interfaces are more than sufficient, but not for ETL-based loading and unloading of data, especially if large quantities of data are involved.

ETL Requirements for Supporting Hadoop – The requirements for ETL tools to support Hadoop efficiently are:

• **Pushdown**: Because Hadoop MapReduce can distribute the processing of logic over a vast number of processors it’s potentially highly scalable. MapReduce has no problems with parallel execution of data filters, string manipulations, mathematical functions, row aggregations, and so on. To be able to exploit this power, integration tools must be able to pushdown as much of the ETL logic into MapReduce jobs. In other words, they must be able to generate MapReduce jobs to extract data from HDFS and let Hadoop execute most of the transformation logic.

• **Data Scalability**: ETL tools must have been optimized internally to process massive amounts of data.

• **NoSQL Concepts**: ETL tools must understand how to handle multi-structured, unstructured (such as text), and structured data stored in HDFS. They must be able to transform such concepts to flat relational concepts, and vice versa.

• **Bi-directional**: ETL tools must be able to read from and write to Hadoop. Especially writing data to Hadoop is not a common feature of ETL tools yet.

• **Schema-on-read**: ETL tools must support schema-on-read. In other words, they must be able to assign a schema to data stored in HDFS the moment when the data is extracted.

• **Data Store Transparency**: MapReduce’s native API is very detailed and complex. Developers have to understand in detail how the technology works and what efficient processing strategies are. Such a low-level interface is bad for productivity and maintenance. In addition, in most BI departments and BICCs the skills required to work with MapReduce and to work with it efficiently are not available. ETL tools supporting Hadoop should hide Hadoop’s technical and low-level APIs. In the ideal situation, for a
developer it should all be transparent whether an ETL job is designed to extract data from a SQL database or Hadoop. Only then is high-productivity guaranteed.

7 Talend’s Support for ETL on Hadoop

When Talend was founded in 2005, they were the first open source vendor of ETL software. In November 2006 they released their first product, the ETL tool called Talend Open Studio. This section describes how well Talend’s ETL solution supports Hadoop.

Talend Open Studio Hides the Technical Aspects of Hadoop – Developers using Talend see one common runtime environment. Whether data has to be extracted from a SQL database, a flat file, a service, or Hadoop, the interface is the same. When ETL code has been developed, it’s deployed on that common runtime environment.

Internally, this common runtime environment consists of a number of runtime environments. Talend currently supports four runtime environments: Java, SQL, Camel, and Hadoop MapReduce. Based on the ETL code, Talend generates native, optimized code for these environments. So, for example, a developer writing code to extract data from a source system doesn’t have to deal with the technical aspects of that system. For each environment optimized code is generated. So, if ETL code has to be executed on Hadoop MapReduce, optimized MapReduce code is generated by the runtime environment developed specifically for Hadoop. Next, this code is pushed down into MapReduce for parallel execution.

In short, Talend developers don’t see MapReduce. They see their familiar, graphical, high-level interface for designing ETL jobs. These specifications are taken by Talend to generate MapReduce code which is pushed down to run in a massively parallel mode. It’s as if Talend Open Studio is the design and development environment and Hadoop is the runtime environment.

Advantages of Talend Open Studio for Hadoop – The advantages of “hiding” Hadoop are:

- Productivity increase and maintenance ease. Designers can design ETL jobs without having to consider Hadoop specifics. In fact, it’s irrelevant which data storage technology is used.

- No training of ETL developers in Hadoop is required.

- When new features are added to Hadoop or when new approaches to programming efficient MapReduce code are invented, Talend developers can deploy them without having to change their ETL specifications. It will purely be a matter of regenerating the code.

- Because the specifics of Hadoop are hidden, it’s relatively easy to embed Hadoop in an existing data warehouse environment. Imagine that an existing architecture uses a SQL database server as staging area. If a more optimized architecture can be designed by replacing the SQL database server by Hadoop, only the existing data must be copied to Hadoop, and the existing ETL code must be regenerated for Hadoop.
To summarize, by hiding the Hadoop technology for ETL designers, it’s easier to embed Hadoop in a current data warehouse environment. In other words, it becomes easier to embed Hadoop and to get the business advantages that Hadoop offers, such as higher data scalability and low costs. By removing the technical hassle of Hadoop, Talend ETL makes Hadoop an easy instrument for optimizing data warehouse environments.
About the Author Rick F. van der Lans

Rick F. van der Lans is an independent analyst, consultant, author, and lecturer specializing in data warehousing, business intelligence, service oriented architectures, data virtualization, and database technology. He works for R20/Consultancy (www.r20.nl), a consultancy company he founded in 1987.

Rick is chairman of the annual European Enterprise Data and Business Intelligence Conference (organized in London). He writes for the eminent B-eye-Network4 and other websites. He introduced the business intelligence architecture called the Data Delivery Platform in 2009 in a number of articles5 all published at BeyeNetwork.com.

He has written several books on SQL. Published in 1987, his popular Introduction to SQL6 was the first English book on the market devoted entirely to SQL. After more than twenty years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian. His latest book7 Data Virtualization for Business Intelligence Systems was published in 2012.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn and via Twitter @Rick_vanderlans.

About Talend Inc.

Talend provides integration solutions that truly scale for any type of integration challenge, any volume of data, and any scope of project, no matter how simple or complex. Only Talend’s highly scalable data, application and business process integration platform enables organizations to effectively leverage all of their information assets. Talend unites integration projects and technologies to dramatically accelerate the time-to-value for the business.

Ready for big data environments, Talend’s flexible architecture easily adapts to future IT platforms. Talend’s unified solutions portfolio includes data integration, data quality, master data management (MDM), enterprise service bus (ESB) and business process management (BPM). A common set of easy-to-use tools implemented across all Talend products maximizes the skills of integration teams.

Unlike traditional vendors offering closed and disjointed solutions, Talend offers an open and flexible platform, supported by a predictable and scalable value-based subscription model.

---

4 See http://www.b-eye-network.com/channels/5087/articles/
5 See http://www.b-eye-network.com/channels/5087/view/12495