Gaining Value From Big Data: Integrating Relational Systems with Hadoop

Colin White, BI Research
May 2013
Sponsored by ParAccel
INTEGRATING THE OLD WITH THE NEW

Big data has evolved from solutions that focus on managing and processing large volumes of multi-structured data to a broad and growing product marketplace that enables many new and innovative data management and analytic solutions. Despite the inevitable confusion caused by this rapidly changing marketplace, it is becoming quite clear that many customers are gaining considerable business value from big data and that this value is best gained from a hybrid of new and existing data systems. This in turns means that integration between these old and new systems is a key success factor and vendor differentiator. This paper discusses the various approaches to integrating new data systems into the existing data management and analytic environment. The paper focuses specifically on integrating the two key components of a big data ecosystem – relational database systems and the Hadoop distributed processing framework. As an example of this integration, the paper also reviews how ParAccel, the sponsor of this paper, integrates its relational analytic platform with Hadoop using the On Demand Integration (ODI) capability.

THE EVOLUTION OF HADOOP

Hadoop was designed to distribute large volumes of multi-structured data\(^1\) across a hardware cluster consisting of hundreds, potentially thousands, of low-cost hardware servers. High performance is achieved using a divide-and-conquer approach that distributes the application processing across these servers. This parallel processing technique is particularly beneficial for applications that sequentially process large data files. Hadoop also includes a programming model, known as MapReduce, which isolates application programmers from the need to know how to code distributed processing programs.

Hadoop is a free open source software framework available from the Apache Software Foundation. Several vendors take this framework, enhance it, and make it available as free or commercial software. Companies such as Cloudera, Hortonworks and MapR, for example, offer enhanced Apache Hadoop distributions. These companies generate their revenue through education and software services, but also, in some cases, by charging for more advanced Hadoop capabilities. In general, these distributions are more appropriate for enterprise purposes compared with the basic Hadoop framework available directly from Apache. Enterprise software vendors, such as IBM, also offer Hadoop, as do cloud-computing companies such as Amazon.

It is important when evaluating solutions for connecting existing enterprise systems to Hadoop to determine the software distributions, products, and product extensions that are supported, because it can affect both performance and compatibility. A good source that documents derivative Hadoop software is the Apache Wiki at: http://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support

---

\(^1\) This paper uses the term *multi-structured data* to describe this type of data. Many people also use the term *unstructured data* in this context, but this is an inappropriate term because most data has some form of structure.
The components of any given Hadoop distribution will vary based on the source of that distribution. From a data systems perspective, the core components of all Hadoop distributions include the Hadoop Distributed File System (HDFS) and the MapReduce (MR) programming model. HDFS is used to store and manage distributed data, and mapper and reducer programs are used to process that data.

A mapper program acts like a data integration tool in that it reads source data and loads it into a data store such as HDFS. A reducer program acts like an analytic tool in that it can be used to aggregate and analyze data. Each node of the cluster runs one or more copies of a mapper program and a reducer program. The performance advantage here is that the processing is taken to the data, rather than the data to the processing. MR programs are typically written in Java, but other programming languages can be used.

Almost all Hadoop distributions also include Apache Hive and Apache Pig. These components provide a higher-level interface to HDFS. Both store data in HDFS and generate batch MR programs to process this data. Hive presents data to developers in the form of tables and includes the HiveQL SQL-like language for accessing and manipulating the data in those tables. Pig provides Pig Latin, which is a procedural scripting language for processing and analyzing data in local and HDFS files. Most distributions also include the HBase database system.

One issue when accessing Hadoop data is that the descriptive information about the data (i.e., the metadata) may be located in HDFS, Hive or Pig. To overcome this issue, Apache has released the HCatalog facility to consolidate this metadata. Support for HCatalog is becoming increasingly important in tools that connect Hadoop to enterprise systems.

In summary, developers have a choice of several techniques for accessing and manipulating HDFS data: directly accessing HDFS files, using MR programs written in Java, using HiveQL, or writing Pig Latin scripts. An increasing number of vendors provide native Hadoop data integration and business intelligence tools that employ one of more of these techniques to process Hadoop data. The objective of these tools is to build out the Hadoop software ecosystem.

Vendors that offer interfaces from relational database systems to Hadoop also use these techniques to access and process data in Hadoop. The motivation here is to provide a hybrid data environment that enables companies to exploit the value of both existing and new data. The architecture of these interfaces has a significant affect on both performance and compatibility.

---

### THE EVOLUTION OF RELATIONAL TECHNOLOGY

Relational database products and SQL have existed for over three decades, and during that time they have become increasingly more sophisticated. Significant development has been done, especially in the last few years, to improve the analytic processing capabilities of relational products. This development work has been

---

2 Hadoop supports a number of different file systems, but HDFS is designed and optimized for use with MapReduce.
driven by increasing demand for enhanced analytic functionality and improved performance as organizations have begun to realize that analytics are a key factor in being able to compete, optimize business operations, and reduce costs.

Enhanced analytic capabilities, improved performance and lower costs have also enabled companies to deploy separate investigative computing platforms that allow data scientists and business analysts to experiment with different types of data and analytic algorithms and models. These experiments are used not only to improve existing business processes, but also to look for new business opportunities. These investigative platforms have been largely enabled by the capabilities provided by integrated analytic relational database hardware/software systems. It is now possible, for example, to use such a system to run risk models in a matter of minutes, whereas previously it took hours. This can provide significant business benefits.

Analytic relational database systems are also used to build new line-of-business (LOB) applications that often employ a mixture of both analytical and operational processing. The dividing line between these latter two types of processing is becoming less well defined as organizations become more information driven. These investigative platforms and LOB solutions do not, however, operate in isolation – they must operate in conjunction with the enterprise data warehouse and other related systems. Sound data integration is therefore vital to success.

Complicating this picture is the growth in the use of multi-structured data and systems such as Hadoop. Today, companies not only want to analyze structured data from traditional operational systems, but also data from the web, social computing, collaborative environments, external information providers, external data systems, and sensors. They want to blend this data with the existing structured data to improve the decision making process. This is especially true for both investigative computing and new LOB solutions.

To support this blending of structured and multi-structured data, vendors can either extend existing relational database products, or integrate relational database software with systems such as Hadoop. In many cases, they have chosen to do both. Several relational database vendors, for example, have extended their products to support the management and manipulation of certain types of multi-structured data. Some have also added the ability to use MapReduce-style analytic functions to process data. The SQL syntax in these products has been extended accordingly. Almost all relational database vendors now provide some form of integration with Hadoop.

**INTEGRATING RELATIONAL AND HADOOP SYSTEMS**

There are many different techniques for integrating disparate data systems. Three requirements, however, need to be addressed, regardless of the technique used:

1. **Copying data between systems,** e.g., copying Hadoop MapReduce (MR) processing results into a relational database for further analysis.

2. **Capturing source data,** **transforming it,** and **loading into a target system,** e.g., extracting useful business information from Hadoop web log data and loading it into a relational system for analysis.
3. **Querying and analyzing data managed by multiple data systems**, e.g., running an SQL query that accesses data managed by both a relational database system and Hadoop HDFS.

The first two of these tasks can be considered to be *data integration*, whereas the third task falls more into the category of *data manipulation*. Data integration can be done in batch or by continuously replicating data between systems. The integration may be done using a data transformation engine, or by generating batch code to do the required work. Data manipulation can be done in batch or interactively. The actual approach used for each of these tasks will depend both on the volume of data involved and performance requirements.

From a performance perspective, it is important that the software enabling data integration or manipulation exploit the capabilities provided by the underlying data systems. An example here would be for the software to take advantage of the parallel computing processing capabilities of both a relational database system and Hadoop. Another example would be to use optimized MR code to access Hadoop HDFS data, rather than generating HiveQL or Pig statements to do the job, which may lead to less efficient MR processing. For certain types of Hadoop data retrieval, however, direct access to the native HDFS file system will provide better performance.

There is not sufficient space in this paper to evaluate all the various techniques used to integrate relational and Hadoop data systems, or the products that support them. Instead, as a way of reviewing the considerations involved, the next section of the paper looks at the Hadoop integration provided by ParAccel.

**ParAccel on Demand Integration (ODI)**

The ParAccel ODI capability is a set of services that allow ParAccel applications and users to access data managed by external data systems. Access to these services is accomplished through the use of SQL user-defined functions (UDFs). For example, ParAccel provides two ODI UDFs for accessing Hadoop HDFS data using MapReduce – one for importing data and one for exporting data. These UDFs can be used in both SQL data definition (CREATE TABLE) and data manipulation statements (SELECT, INSERT, UPDATE and DELETE). This allows external data to be combined interactively with ParAccel data for reporting and analysis purposes. It also allows data to be extracted from external data systems and loaded into ParAccel tables for subsequent analysis. Multiple UDFs can be employed in any given SQL statement.

ParAccel provides ODIs for several external data systems including ODBC, Teradata, and Hadoop. Systems integrators and developers can also use the ParAccel ODI development kit to create their own ODI capabilities in a programming language of their choosing. This paper focuses specifically on the ODI for Hadoop, which supports Hadoop distributions such as Apache, Hortonworks and Cloudera.
ParAccel Analytic Database

Before reviewing ParAccel ODI for Hadoop in detail, it is necessary to first provide a brief overview of the ParAccel Analytic Database (PADB) architecture. A PADB system has four main components: the leader node (or leader), the compute nodes, the communication fabric, and an optional storage area network (SAN).

The leader (see Figure 1) interfaces with external applications and the rest of the IT infrastructure. It interacts with applications using SQL and standard ODBC, JDBC or .NET interfaces. PADB provides a UDF capability that enables developers to code their own analytic functions. ParAccel includes some 550 prebuilt functions with PADB.

The leader also controls the execution of the compute nodes, which are responsible for the storing and processing of data. Each compute node stores and manages a subset of the rows of each table. Data is distributed to a particular compute node based on a hashing algorithm applied to a user-defined distribution key, or by a round robin algorithm. PADB usually stores data on direct-attached storage to eliminate connection bottlenecks associated with external storage. Compute nodes, however, can also be configured to support SAN environments.

Compute nodes are logically subdivided into a set of parallel processes called slices, each with its own processor core, memory, and a portion of each disk. Slices work in parallel regardless of the work they are processing. When loading data, slices parse the data into columns, and then compress and write the data to disk.

Leader and compute node slices communicate using the communication fabric, which employs Gigabit Ethernet and the ParAccel interconnect protocol. This latter protocol implements multiple simultaneously running data streams to support the high-performance requirements of analytic processing. Leader and compute nodes are standard x86 servers running Linux.

ParAccel ODI for Hadoop

As discussed earlier, users and programmers access external data systems using ODI UDFs. The SQL statement below shows an example of using the ODI Hadoop import UDF (odi_hadoop_import) to populate a PADB table using data stored in Hadoop:

```
CREATE TABLE imported_parts AS
SELECT * FROM ODI_HADOOP_IMPORT
(WITH JOBNAME ('import_parts_job')
MASTERNODE ('hadoop-master-hostname')
INPUTDIR ('/hdfs-directory-name')
PADB_SCHEMA ('imported_parts_table');
```

This SQL statement creates and populates a table called imported_parts using textual data extracted from the Hadoop file in the directory /hdfs-directory-name. The structure of the new table is defined using the padb_schema option, which specifies a PADB table or view with a schema that matches the order and types of the fields in the Hadoop file. It is also possible to have ODI create the PADB schema automatically. ODI provides an equivalent export UDF for loading data into Hadoop from a PADB table.
Figure 1 will be used to explain how PADB interacts with Hadoop to process the SQL statement above using the ODI_HADOOP_IMPORT UDF.

1. ODI activity begins with a user or an application executing an SQL statement that includes the ODI_HADOOP_IMPORT function.

2. To start the Hadoop import, the PADB leader node connects with the ODI Manager installed on the Hadoop cluster master node, and passes it the name of the job that is to be used to extract the Hadoop data. The ODI Manager then launches this job, which in turn causes mapper instances on the nodes of the Hadoop cluster to be started. ParAccel provides mappers for common Hadoop data formats. The ODI development kit can be used to develop additional mappers. The PADB leader node also starts ODI UDF instances on PADB compute node slices.

3. Each UDF instance on a PADB compute node slice then polls the ODI Manager to be assigned a mapper instance from a Hadoop node. The ODI Manager responds by assigning an available mapper to each PADB UDF instance.

4. Each mapper task processes files in the specified HDFS directory and data transfer begins, in parallel, between each mapper instance on the Hadoop nodes and its assigned UDF instance on a PADB compute node slice. When all the data from each mapper instance has been sent, the corresponding mapper instances are terminated. The ODI Manager then informs all PADB UDF instances that the job has completed, and each instance then returns an end-of-data message to the leader node, signaling the import is complete.
Optional HCatalog Interface

Apache HCatalog is a metadata management service for data created by Hadoop components such as Hive, Pig and MapReduce. It provides a metadata abstraction layer that isolates Hadoop users and tools from needing to be concerned with how data is created, or where it is physically located.

ParAccel ODI provides an optional interface to the Hadoop HCatalog facility, which enables data to be imported from Hadoop using Hive table definitions. When using HCatalog, the SQL statement coded in the ODI Hadoop import function specifies the Hive tables, columns and partitions from which data is to be retrieved. The syntax for this SQL SELECT statement is shown below:

```
SELECT * FROM ODI_HADOOP_IMPORT
(WITH JOBNAME ('import_parts_job')
 MASTERNODE ('hadoop-master-hostname')
 HIVE_TABLE ('hive-table-name')
 COLUMNS ('hive-table-columns-list')
 PARTITION FILTER ('filter-condition')
```

The operation of ODI using HCatalog is similar to that shown in Figure 1, except that PADB now interacts additionally with HCatalog to retrieve the required Hive metadata for filtering the data to be retrieved from Hadoop. At the time of writing, ParAccel was extending ODI to support HiveQL, which will enable the Hadoop import function to be used to restrict data access to specific Hive records.

Comparing the facilities outlined above with the three requirements for integrating disparate data systems discussed earlier, it can be seen that ParAccel ODI supports the copying of data between Hadoop and PADB and the analyzing of data managed by both Hadoop and PADB. Although the SQL used in an ODI task could be used for transforming data, it would not be suitable for doing extensive data transformation.

The most likely use of ODI is to bring result data from Hadoop processing into PADB for further analysis, and if required, exporting the results back to Hadoop for low-cost storage. By exploiting the parallelism inherent in both PADB and Hadoop and by using prebuilt mapper code, the ODI facility can do this efficiently and with good performance.

CONCLUSION

The key to gaining value from the analytic innovations offered by big data technologies is to be able to process data managed by existing relational database systems and also new data in non-relational systems such as Hadoop. In some cases, this may involve the running of ad hoc analyses that process data from both environments, while in other situations the requirement may be to move data from one system to another to achieve better analytic performance, and then return the results from the analytical processing to the original system. Regardless, the requirements here are to provide good performance and compatibility.
Carefully evaluate the performance and compatibility of Hadoop integration options

As this paper has shown there are numerous options for interconnecting a relational database system with Hadoop. ParAccel ODI is one such option. In order to be successful organizations need to carefully evaluate these options to determine not only that it meets business needs, but also that it provides good performance and compatibility.

About BI Research

BI Research is a research and consulting company whose goal is to help organizations understand and exploit new developments in business intelligence, data integration, and data management.