Integrating SQL and Hadoop

Jean-Pierre Dijcks and Martin Gubar

Abstract

There is little doubt that big data is here to stay; over the last couple of years, it has grown in importance as a critical element to any data warehousing strategy. However, there are still many questions about the best way to implement a big data strategy and what tools are best for the job. One approach that has generated significant interest is integrating SQL and Hadoop.

This article highlights the benefits of integration, the pros and cons of various integration methods, and how adoption of this approach will change the data warehouse and the big data landscape.

Jean-Pierre Dijcks is senior principal product manager for big data with Oracle.
jean-pierre.dijcks@oracle.com

Martin Gubar is director of product management for big data with Oracle.
martin.gubar@oracle.com

Introduction

Do we need both a data warehouse and Hadoop? In the short term, the answer is clearly "Yes." A more meaningful question is: What's the best way to use them together to advance the goals of our business?

Clearly, a successful enterprise needs a well-defined business strategy to advance its goals. For an automobile manufacturer, the corporate mission may be to produce the highest-quality cars that deliver an exhilarating driving experience and to consistently exceed the expectations of its customers. How does the company deliver on that mission? How does it capture the metrics required to constantly improve quality, reach out to customers, and enhance their driving experiences? Companies must look for data both inside and outside the organization.

The data warehouse is the foundation that allows an enterprise to achieve its mission by integrating information from operational systems, ERP, and master data. Effective use of the data warehouse and business

intelligence applications allows an enterprise to forecast demand, manage suppliers and inventory levels, and monitor quality and customer satisfaction. However, these traditional data sources are no longer sufficient. Relying solely on "structured" or "strongly typed" data means missing opportunities that are now within reach.

Consider the potential impact of the connected car. In the past, the key measurements generated from an automobile consisted of little more than miles driven, the amount of fuel in the tank, and engine oil pressure. Even these few measurements—with the exception perhaps of miles clocked on the odometer—were not tracked consistently. It was an analog world; capturing automobile performance outside of controlled environments was extremely difficult.

Today, modern vehicles have more than 20 different sensors constantly capturing critical aspects of performance, including oil life, mileage, speed, coolant temperature, engine RPMs, and tire pressure. Combining these on-board diagnostics with global positioning and weather information adds two more dimensions to the types of analyses that can be performed. Furthermore, customers leave a constant record of their interactions with products and companies via social media, websites, online forums, phone calls, surveys, and purchases.

What does this mean for the automobile manufacturer that can harness this "connected car" data stream? First, quality control testing is no longer limited to the confines of in-house environments and a fixed set of use cases. Real-world experience in uncontrolled settings can now be captured—with hundreds of variables correlated to identify problematic patterns. This allows engineering teams to improve product designs and focus their quality control efforts.

In addition, these patterns can also feed predictive models that are subsequently deployed to the on-board computer diagnostics system in the automobile. Instead of the analog world where a driver is stuck on the side of a road waiting for a tow, the early detection system can alert the driver to a problem before it becomes serious. The early detection system can also notify the local auto dealer's customer service representative, enabling him or her to proactively contact the customer, schedule an appointment, and order any required parts.

The challenge of integrating fast-moving, high-volume, "weakly typed" data sources generated by devices such as the connected car have largely been overcome with the introduction of new big data technologies. Hadoop and NoSQL data stores provide an economically viable approach to collecting data that is often of low value.

But wait—after we have just explained the virtues of leveraging on-board diagnostic information, why are we now considering it "low value"? Diagnostic data is fed into the system for a given car thousands of times per day. Because we are now collecting all the readings—not just a small sample—some "dirty data" is acceptable. A missing diagnostic reading is unlikely to materially impact the result of an analysis that utilizes millions or even billions of data points as input. The same claim cannot be made for traditional high-value enterprise data. For example, incorrect inventory for a given part may lead to a stock outage, resulting in service delays and decreased customer satisfaction.

The information management and analysis process of structured data is familiar:

- A business objective is identified, such as "improve product quality"
- Relevant data is captured, prepared, and analyzed
- Analytic models are generated from the data and evaluated for accuracy
- Successful models are deployed to operational systems where appropriate
- Results from the model are monitored with business intelligence tools using time-series analyses, comparing actual results to targets, etc.

This is an iterative process. Over time, models must be refined based on changes to underlying data and conditions.

For example, an engine part in a new car model may begin to fail at a certain age based on particular driving conditions and climates. Clearly, these new tendencies will emerge as important considerations in the predictive model—whereas initially they may have been ignored. You want to optimize the performance and productivity of this analytic life cycle, maximizing the number of iterations to continuously improve product quality.

The addition of weakly typed data doesn't significantly change the overall process, although it does add new data types and coding practices may differ. However, the end goal is still to bring the results of the new analysis into the models that drive business processes. The question of where processing takes place—in Hadoop or the relational database—will depend on numerous factors and considerations, including:

- Tooling maturity
- Performance
- Security
- Ingestion rate
- Cost effectiveness of storing low-value data
- ETL simplicity
- Data sparsity
- Variety of data formats
- Data management requirements

There are certain areas where each platform will shine, which is not to say that it is impossible for each platform to potentially support given criteria. Hadoop is "schema-less"; a schema is not imposed until the data is accessed (known as "schema on read"). This flexibility has many benefits. It allows extremely fast ingestion rates of weakly typed data. If the format of the car diagnostic data changes over time, it will have less impact on ETL procedures (although you must be aware of these changes on read!). The MapReduce framework is massively parallel—allowing effective processing of this highly sparse and weakly typed data.

From a storage and computing perspective, Hadoop-based systems are very cost effective. On the other hand, relational databases are structured, providing "schema on write" support. A wide range of highly mature tools allows users to productively explore and analyze relational data with outstanding performance. Business users can perform ad hoc, multidimensional analyses using shared definitions of business metrics and hierarchies. Data is secured, trusted, and audited using proven methodologies.

The challenge today is not "can we do everything in Hadoop or everything in the data warehouse?" Today's challenge is to integrate these two environments to maximize the business value to the organization. Indeed, it's not just in the short term that these two platforms will coexist and complement each other. The dynamics that push certain kinds of data or use cases to one platform or the other are likely to be around for a while, and the data warehouse/Hadoop combo is here to stay.

Addressing the Challenge

To create a single data repository and the language for big data, we should now ask new questions. We need to know how to integrate data across Hadoop and the data warehouse and what language should be used to query all that data.

As with any new technology, adoption happens in phases. As the new technology matures, its integration in the IT landscape starts to evolve. We can identify three stages in integration and desired integration, as illustrated in Figure 1.

As the systems mature (both Hadoop itself and the organization's adoption of Hadoop), the value provided by the combination and integration of systems grows. The main reasons for the increase in value derive from the reduction of complexity, the faster availability of data (no more data movement), and the ability to join and correlate data as if it were in the same system.

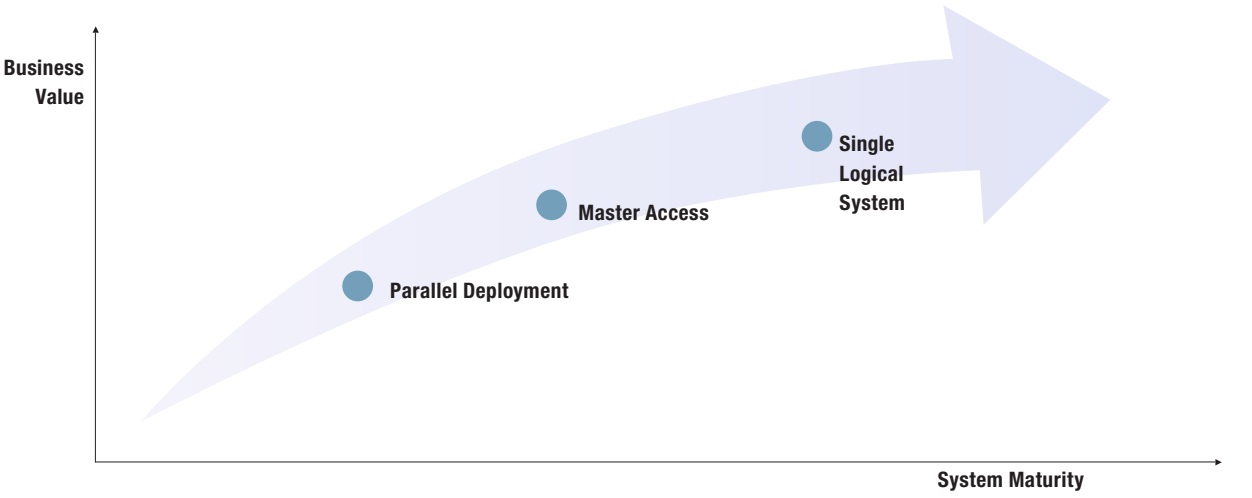

Figure 1: Maturity phases for Hadoop and RDBMS implementations.

Of course, we also need to answer three big questions: where in this maturity curve is the industry, what should a user aim for, and why? It is best to describe the phases and their benefits and drawbacks in more detail.

Phase 1: Parallel Deployment

In the first phase of Hadoop adoption, the Hadoop system is implemented in parallel with the current RDBMS-based systems. This isolates the Hadoop system from the rest of the organization, allowing the organization to learn and understand Hadoop's strengths and weaknesses.

Currently, this is the state of Hadoop deployments in a large number of organizations. These organizations often are testing the systems in proof-of-technology projects (should my organization look at implementing Hadoop?) and are primarily focused on understanding the technology or solving a very specific use case.

Hadoop comes with a powerful MapReduce programming framework, but it is unfamiliar to most people. Accordingly, when trying out the Hadoop system, one of the first steps is to turn to Hive and its HiveQL language. Hive provides database-like metadata describing Hadoop (HDFS) data in tables, columns, and even partitions.

On top of that metadata, HiveQL provides a declarative "SQL-like" language that is translated into MapReduce programs. The key benefit of Hive is that it allows SQL skills to be used to access data on Hadoop using a language most people are comfortable with. The main downside (when compared to a database) is the response time of a typical query. Because HiveQL is executed as MapReduce, it has long response times.

> What most organizations have is a hybrid approach to accessing data. At different times, Hadoop, the data warehouse, and BI tools are all used.

Other than in specific use-case solutions where the Hadoop system is isolated from the rest of the data ecosystem, organizations generally leave this state once they decide that the Hadoop solution either drives value or reduces cost. At that point, the organization will

quickly look at moving into Phase 2 to reap the benefits of analyzing all the data together via a common set of tools and languages. SQL becomes a critical part of this equation.

Phase 2: Hybrid Master Access

Once an organization has demonstrated value from using Hadoop, it will try to expose more end users to the data stored and analyzed in that platform. We could qualify this as accessing data from a "master data point"—the central consolidation point that provides an integrated view of the data. Generally there are three places that may act as the master:

- The data warehouse is fed using ETL jobs. The Hadoop system may be considered another data source.
- The BI tool federates across the data warehouse and the Hadoop system.
- The Hadoop system receives data from other repositories, such as the enterprise data warehouse, where dimension data is moved into the Hadoop system.

Early adopters of the Hadoop technology are typically in Phase 2 when providing data access for all user communities. Pragmatism demands they employ a hybrid of the three masters above. These hybrid solutions provide dedicated access to a subsystem while requiring both data movement and some level of aggregation. For example, a small subset of users might access Hadoop via Hive, whereas most users access data via the BI tool that solely queries the DW or federates.

Hybrid master access is a practical way to acess all data, but this approach also causes a number of problems. The most notable issues are the complexity introduced across the system and the further replication of data.

Federation could be a good answer in that this type of access allows the data to stay in one place. However, federating data across a data warehouse and Hadoop introduces latency issues. A typical HiveQL query will run much slower than a similar query in a data warehouse. Joining data—federated in the BI tier—will incur a performance cost as well because the data from the Hadoop system will be streaming more slowly. In summary, federation will usually degrade query run times to most BI users' dissatisfaction.

The data warehouse is the primary access point because it already contains a large quantity of data that has been scrubbed and is known to be valuable. It is also a more secure environment and more likely to meet regulatory and data governance requirements. Unlike Hadoop, the drawback of the data warehouse is not query performance. Because data is aggregated (the raw data remains in Hadoop), the data warehouse may not be suitable for answering all questions. When that situation occurs, you will need to revisit the raw data, using Hadoop to run a different filter or analysis, before loading a new aggregate.

Technology solutions to address some of these problems are slowly entering the market. On the Hadoop side, SQL engines are starting to emerge. These engines replace the MapReduce execution component with a SQL execution engine and provide a new optimizer to generate these SQL execution plans. Almost all of these engines are in their very early versions, often written from scratch. Due to their newness, they lack a large number of common RDBMS features and fail to bridge the gap between Hadoop and the data warehouse. They are best thought of as a way to improve access to the subset of enterprise data that is stored on Hadoop—not as a unified access point.

BI tools, which now often have direct Hive access, are starting to leverage in-memory engines (such as Oracle Exalytics) that allow the cache for the Hadoop system to reside in the BI server tier, automatically caching results that are accessed often. The caching is, in many cases, more mature than that performed by the Hadoop SQL engines. Caching is frequently a more well-rounded solution when using federation.

What most organizations have, then, is a hybrid approach to accessing data. At different times, Hadoop, the data warehouse, and BI tools are all used. This necessitates

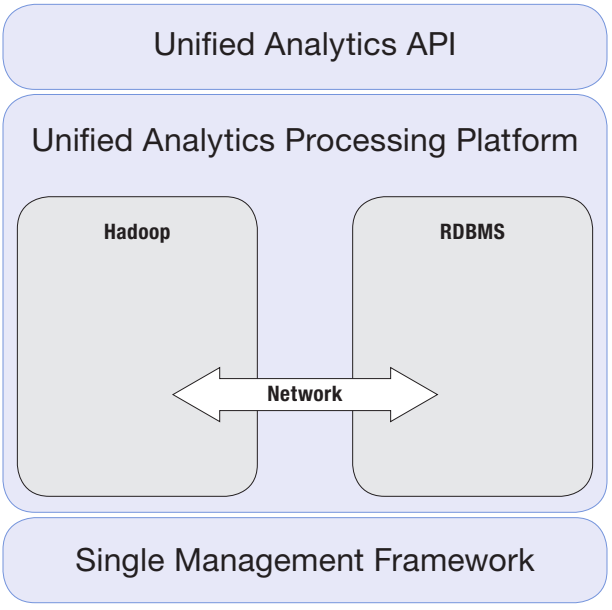

Figure 2: A unified processing platform for big data.

moving key subsets of the entire data set between the different environments.

Phase 3: A Single, Logical System

The real solution to the access and data movement problems—and the real answer to a single data store for analysis—is to create a more tightly integrated system that unifies a relational database with Hadoop without losing sight of the unique value proposition each subsystem adds to an organization.

The industry as a whole has not made it to this stage, and almost no actual implementations are currently there. The industry's energy should be focused on finding a language and implementation to build a single, logical system.

SQL (Data Warehouse + Hadoop) = More Business Value

The single, logical system discussed in Phase 3 will need to ensure that a large number of constituents have access to the system. One approach is to standardize on a declarative language used across data systems. As seen earlier, SQL in various forms is already becoming the access language of choice for Hadoop, just as it is for the data warehouse. Although there will always be other languages employed for specific problems (see sidebar), SQL will be the preferred language for an integrated Hadoop and data warehouse environment.

By embracing SQL, all existing tools (BI, applications, etc.) will have full access to data without changes in the vast ecosystem of end-user tools and applications. Embracing SQL also means that the system should optimally execute queries across the data warehouse and Hadoop without some of the drawbacks we see in Hive. We will want to ensure that Hadoop is still Hadoop—retaining its high data ingestion rates, flexibility, and agility in working with data, and, of course, it must retain its scalability.

The blueprint for such a system is shown in Figure 2. The unified analytics processing platform encompasses both the Hadoop system and the RDBMS. These systems are first-class citizens and allow for specific workloads to be directed at them. The network connecting the RDBMS and Hadoop should be a high-speed connection to optimize data flows between both systems. Examples include InfiniBand and the latest Ethernet technologies (those greater than 10GigE).

To query the system via SQL, the API must access a unified metadata store that captures definitions across the logical system components. On Hadoop, technologies such as Hive and HCatalog will most likely be the basis for this metadata, and it will be connected and integrated with the database catalog to allow a unified SQL syntax and unified access to data across the processing platforms.

The big advantage of the approach in Figure 2 is that information consumers now have access to all data without changes to their language or their applications for data access. The performance of this system is much better because the SQL is pushed directly into the relevant processing layer of the system. Data movement is reduced as queries are executed locally in either Hadoop—at which point only results flow back to the user—or in the RDBMS.

All other approaches still work, of course. Federation is now much simpler because the BI servers can focus on caching hot data rather than caching all data, allowing more queries to run faster. All Hadoop workloads and tools work seamlessly on the Hadoop side, enabling the required flexibility and deep analytics on fine-grain data.

As we discussed, the industry is currently just reaching Phase 2 of our model. Hadoop—as an extension to the data warehouse—certainly drives considerable value into the organization, but it is crucial for companies to refrain from architecting systems with Phase 2 as the end goal. The end goal for each and every installation should be to allow all users to gain access to all relevant data using SQL.

Conclusion

Hadoop will be a key ingredient for most data warehouses in the next few years, and SQL on or with Hadoop is an increasingly important topic. Today's solutions are either federating data in the BI layer or moving data into the data warehouse for mass access. New SQL engines on Hadoop are currently limited to performance boosts in federated scenarios. The real end game is to bridge both data stores with SQL. Over the next few years, this approach will become the mainstay of enterprises in deriving value from all their data. ■

Just SQL?

This article focuses on SQL as the main language to query data across the Hadoop and RDBMS platforms. SQL is probably the most widely "spoken" data access language, but your organization should be careful about focusing on SQL alone. To round out a big data platform, we expect other languages to be important for exploration and analytics. Here are the three key areas to consider.

MapReduce

Often declared hard to write and complex, MapReduce is a powerful framework that allows exploration and procedural code across an endless stream of data types at incredible scale. Sometimes just being able to solve a problem is good enough, and we expect MapReduce to have a place in most large-scale Hadoop deployments. We would, therefore, expect MapReduce to also be introduced as a programming paradigm that works in the RDBMS. As big data platforms evolve, having MapReduce capabilities inside the RDBMS will become a table stake and will allow database programmers an additional avenue to program in the RDBMS.

R

One of the most interesting and promising languages for big data is R. However, R is typically used on small data volumes because it cannot scale. As R is adopted by large vendors such as Oracle, scaling limitations are starting to disappear and R is becoming interesting as a statistical language and discovery framework in big data.

When working with languages such as R, a user wants scalability and direct access to all necessary source data (which could span both Hadoop and the RDBMS). R, therefore, should also start to follow the unified API paradigm or at least be executable across these two platforms because it is much easier to move code than to move all that data.

Graph Analytics

Along with R, graph analytics is increasingly popular as we venture into social graphs and causal effects modeled in graphs. Languages such as SPARQL and APIs such as Blueprints are used across data sets and are becoming increasingly interesting for very large data sets, often stored in Hadoop or NoSQL databases. Having a portable API that works across platforms and data will help users in driving business actions.