

# New Paradigms for High Performance Analytical Computing

---

Prepared for Vertica Systems by:

David Loshin  
Knowledge Integrity, Inc.  
October, 2009

## Introduction

Two high performance computing technologies are transitioning into the mainstream: high performance massively parallel analytical database management systems (ADBMS) such as Vertica (and at least a dozen others), and distributed parallel programming paradigms, such as MapReduce, and its affiliated implementations and support services such as Hadoop, Pig, and HDFS. Since the introduction of MapReduce as a scalable programming model in 2004, its popularity has grown; for example, a variety of over 50 commercial applications running on clusters using Hadoop, an open source MapReduce implementation, are described at <http://hadoop.apache.org>.

There are emerging schools of thought that contend that these paradigms are incompatible, are completely redundant, or are mutually exclusive. And although MapReduce and its companion technologies are not a replacement for an interactive analytical database system, there are specific business processes/applications that benefit from using both in a collaborative manner, particularly where the results of a MapReduce application feed the types of comprehensive analysis and reporting provided by analytical database systems. This paper is intended to consider both approaches, explore their similarities and differences, and demonstrate that a certain synergy can be achieved through a collaborative use of the two approaches together.

First, we look at how growing data volumes are driving the need for high performance analytical systems, and then reflect on the desired properties that high performance analytical systems should exhibit. The paper provides an introduction to the MapReduce programming model and to high performance, highly scalable analytical database management systems, especially those (such as Vertica's) that implemented using a columnar data orientation. The paper reviews the degrees to which each of these approaches exhibit the desirable characteristics, and how these approaches are similar and how they differ. This review will highlight where the two approaches can be combined to provide a best fit solution for different types of applications, and the paper provides some examples.

By providing an overview of both concepts and looking at how the two approaches can be used together, we conclude that combining a high performance batch programming and execution model with an high performance analytical database provides significant business benefits for a number of different types of applications.

## The Growing Volume of Data to be Analyzed

Recent reports show the monumental growth of structured data; what was described in 2008 as “a large commercial entity – that will soon have about 300 TB (*terabytes*) of data in its enterprise warehouse” does not seem to be so outrageous in 2009. However, the data continues to expand – “the size of the largest data warehouse ... *triples approximately every two years.*”<sup>1</sup> Structured data is not alone – its growth is “far outpaced by a 61.7% CAGR predicted for *unstructured data* in traditional data centers.”<sup>2</sup>

In conjunction with increasing data volumes is a thirst for computational power to absorb and analyze all of this data in an attempt to gain a competitive edge. In 2005, the analytical database market accounted for approximately 27% of the worldwide RDBMS market<sup>3</sup>, and if that trend has continued, it would suggest that more than \$4.5 billion of the \$18.8 billion worldwide market for RDBMS software in 2008<sup>4</sup> was spent on analytical database systems. Not only that, the computational scale is attracting significant investment as reported in an IDC study that expects “spending on IT cloud services to grow almost threefold, reaching \$42 billion by 2012 and accounting for 9% of revenues in five key market segments.”<sup>5</sup>

## Desired Properties for High Performance Analysis

As the volumes of data explode at a tremendous rate into the petabyte range, the need to analyze that data and synthesize the results into actionable knowledge within a reasonable time frame has become a driving force in the information-driven organization. By necessity, analysis platforms are becoming more reliant on high-performance, scalable environments in order to meet this increased demand for knowledge.

Operational and transaction systems generating billions of log entries, real-time web statistics, unstructured documents, millions of call detail records – all these are examples of continuously-streaming data sources that must be combined, analyzed, and then fed back results to improve the business. The need to absorb, analyze, and deliver results suggests that there are some desired characteristics that a high performance analysis framework should exhibit, such as:

---

<sup>1</sup>Richard Winter, “Why Are Data Warehouses Growing So Fast?” – B-eye-network.com, April 2008

<sup>2</sup>Beth Pariseau, “IDC: Unstructured data will become the primary task for storage,” reference to IDC survey at techtarget.com, October 2008

<sup>3</sup>Abouzied, Bajda-Pawiloski, Abadi, Silberschatz, Rasin, “HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads,” VLDB 2009

<sup>4</sup>Gartner report G00169004 “Market Share: Relational Database Management System Software by Operating System, Worldwide, 2008” Colleen Graham, Bhavish Sood, Hideaki Horiuchi, Dan Sommer, (see <http://www.gartner.com/DisplayDocument?id=1018712>)

<sup>5</sup>“IDC Finds Cloud Computing Entering Period of Accelerating Adoption and Poised to Capture IT Spending Growth Over the Next Five Years,” October 2008, IDC press release  
<http://www.idc.com/getdoc.jsp?containerId=prUS21480708>

- **Data volumes** – As more data is created and made available, the analysis platform must be able to absorb and handle larger volumes.
- **Performance** – There are two considerations that drive the need for increased performance: massive data volumes (which may include unstructured content, massive web transaction logs, daily terabyte feeds of call detail records), and the desire to distill critical bits of actionable knowledge from that data. Improving performance is driven by scalability so that the application's runtime improves in proportion to the computational, network bandwidth, and storage resources as they are added to the cluster.
- **Data integration** – Not only has the volume of data increased, so has its variety. Analytical applications increasingly depend on the combination of both structured and unstructured data, either managed internally or fed by external sources, from static or persisted sources or from continuous streams.
- **Fault tolerance** – Increased analytical complexity usually goes hand in hand with complex computations of greater duration, exposing the environment to the risks of system failures. It is desirable to enable recovery from a failure without having to restart the entire process. At a higher level, some systems provide a level of fault tolerance that allows queries or processes to be restarted at specific checkpoint location when some component fails.
- **Heterogeneity** – With emerging approaches to virtual cluster and cloud computing relying on a variety of computing resources, the potential to scale the analysis framework (to hundreds, or even thousands machines) using homogenous or heterogeneous systems provides potential for even more flexibility in resource allocation and usage.
- **Knowledge delivery** – The environment should be able to support the computational needs to perform the analysis as well as the interactive means to deliver and present the actionable results.
- **Latency** – The time from when data is recorded to when questions are answered is a critical component when reacting in a dynamic business environment. Low latency means fast ingest of data, dynamically scalable processing and fast query response times all operating concurrently.

Keeping these characteristics in mind, we can review two different approaches to scalable high performance data analysis that are growing in popularity: the MapReduce programming model, and the use of high performance analytical databases.

## MapReduce

MapReduce is a programming model introduced and described by researchers at Google for parallel computation involving large data sets that are distributed across clusters of many processors. In contrast to the explicitly parallel programming models typically used with imperative language such as Java and C++, the MapReduce programming model is reminiscent of functional languages such as Lisp and APL, in its reliance on two basic operational steps:

- *Map* which describes the computation or analysis to be applied to a set of input key/value pairs to produce a set of intermediate key/value pairs, and
- *Reduce*, in which the set of values associated with the intermediate key/value pairs output by the *Map* operation are combined to provide the results.

Conceptually, the computations applied during the *Map* phase to each input key/value pair are inherently independent, which means that both the data and the computations can be distributed across multiple storage and processing units and automatically parallelized.

### A Common Example

The ability to scale based on automatic parallelization can be demonstrated using a common MapReduce example that counts the number of occurrences of each word in a collection of many documents. Looking at the problem provides a hierarchical view:

- The total number of occurrences of each word in the entire collection is equal to the sum of the occurrences of each word in each document;
- The total number of occurrences of each word in each document can be computed as the sum of the occurrences of each word in each paragraph;
- The total; number of occurrences of each word in each paragraph can be computed as the sum of the occurrences of each word in each sentence;

This apparent recursion provides the context for both our *Map* function, which instructs each processing node to map each word to its count, and the *Reduce* function, which collects the word count pairs and sums together the counts for each particular word. The runtime system is responsible for distributing the input to the processing nodes, initiating the *Map* phase, coordinating the communication of the intermediate results, initiating the *Reduce* phase, and then collecting the final results.

While we can speculate on the level of granularity for computation (document vs. paragraph vs. sentence), ultimately we can leave it up to the runtime system to determine the best distribution of data and allocation of computation to reduce the execution time. In fact, the value of a programming model such as MapReduce is that its simplicity essentially allows the programmer to describe the expected results of each of the computational phases while relying on the compiler and runtime systems for optimal parallelization while providing fault-tolerance.

## MapReduce, Hadoop, and Characteristic Applications

Hadoop is a popular open source project that not only has incorporated an implementation of the MapReduce programming model, but also includes other subprojects supporting reliable and scalable distributed computing such as HDFS, (a distributed file system) and Pig (a high-level data flow language for parallel computing), along with others (see <http://hadoop.apache.org>). The range of applications that use Hadoop show the versatility of the MapReduce approach, and reviewing them provides some of the typical characteristics of problems suited to this approach:

- Massive data volumes,
- Little or no data dependence;
- Uses both structured and unstructured data;
- Amenable to massive parallelism;
- Requires limited communication.

Some good examples that display some or all of these characteristics include:

- Applications that boil lots of data down into ordered or aggregated results – sorting, word and phrase counts, building inverted indices mapping phrases to documents, phrase searching among large document corpuses.
- Batch analyses fast enough to satisfy the needs of operational and reporting applications, such as web traffic statistics or product recommendation analysis.
- Iterative analysis using data mining and machine learning algorithms, such as association rule analysis or k-means clustering, link analysis, classification, Naïve Bayes analysis.
- Statistical analysis and reduction, such as web log analysis, or data profiling
- Behavioral analyses such as click stream analysis, discovering content-distribution networks, viewing behavior of video audiences.
- Transformations and enhancements, such as auto-tagging social media, ETL processing, data standardization.

### High Performance Analytic Databases

The need to provide access to massive data sets for business intelligence analysis and reporting suggests that our intelligent information processing requirements are outgrowing the performance capabilities of traditional row-oriented relational database management systems. Instead, what has emerged is a mature, capable alternate database management approach, such as one that is provided by Vertica, which organizes data by columns instead of rows as a way to handle heavy business intelligence workloads while meeting the cost and performance requirements of the growing enterprise analytic and data warehousing infrastructures.

### Traditional Databases and the Performance Challenge

Traditional RDBMS systems store their records by rows. But business intelligence and analytic applications, generated reports, and ad hoc queries usually target selected attributes of a very large number of records, needing only the values in selected columns or aggregates of those columns to support the user’s analysis needs. Since row-oriented databases must read the entire record in order to access the needed attributes or column data, ad hoc queries will end up reading a lot more data than necessary to provide the requested result, leading to a heavy load for I/O and network bandwidth.

For predefined reports, system architects and DBAs can tune the system by building additional indexes and pre-aggregating data, although these add to both processing time and demand for persistent storage. These tweaks improve known queries, but do not contribute to performance optimization for ad hoc queries.

### Column-Oriented Data Management Systems

Clearly, there are limits to the performance which row-oriented systems can deliver when tasked with a large number of simultaneous, diverse queries. Therefore, if row-oriented layout degrades general performance for analysis, consider the alternative: organizing the data vertically along the columns.

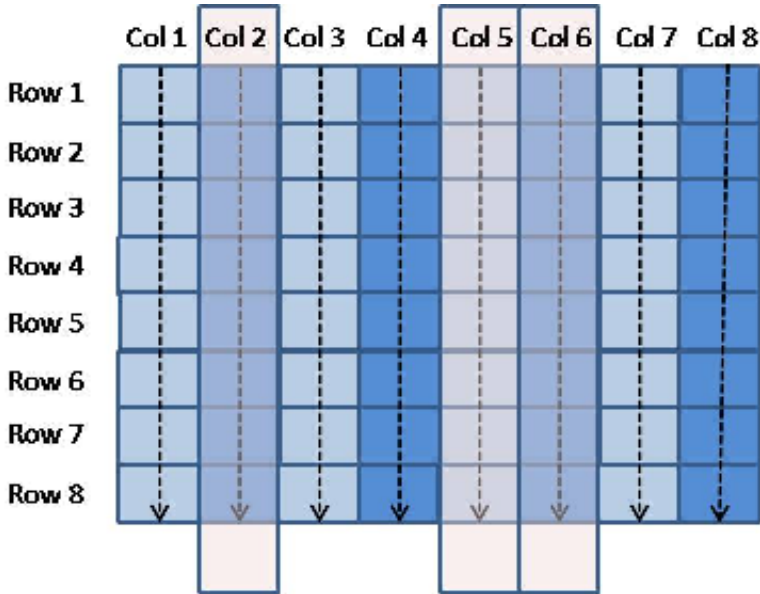


Figure 1: In a column-oriented database, only the columns in the query need to be retrieved.

The column-oriented approach can improve performance in a number of ways, such as these:

- Because each column can be stored separately, for any query, the system can evaluate which columns are being accessed and retrieve only the values requested from the specific columns.
- The columnar storage of data can eliminate the need for multiple indexes, views and aggregations, and is also amenable to compression, which reduces the storage footprint. Executing queries while maintaining the data in compressed format reduces the I/O burden and instead shifts that burden to the CPUs which are orders of magnitude faster than disks.
- Scanning data across columns allows for incremental computation of aggregate results, which are often the desired results within business intelligence applications. Columnar orientation coupled with aggressive compression frees up space so that data can be distributed in different ways. Columns that are accessed frequently together can be co-located to improve locality of reference.
- In addition, columns can be replicated multiple times, which not only speeds joins, also supports fault-tolerance.

### Architectural Choices

Analytical databases are deployed in different architectural models. Even on parallel platforms, many databases are built on a shared-everything approach in which both the disk and memory are shared by the different CPUs. A shared-disk approach may have isolated processors, each with its own memory, but the persistent storage on disk is still shared across the system. These types of architectures are layered on top of symmetric multiprocessor (SMP) machines. While there may be applications that are suited to this approach, there are bottlenecks that exist because of the sharing, because all I/O and memory requests are transferred (and satisfied) over the same bus. As more processors are added, the synchronization and communication needs increase exponentially, and therefore the bus is less able to handle the increased need for bandwidth. This means that there will be limits to the degree of scalability.

In contrast, in a shared-nothing approach, each processor has its own dedicated disk storage. This approach, which maps nicely to a massively parallel processing (MPP) architecture, is not only more suitable to discrete allocation and distribution of the data, it enables more effective parallelization, and consequently does not introduce the same kind of bus bottlenecks from which the SMP/shared-memory and shared-disk approaches suffer.

### Distinguishing Characteristics

Both of these models hold a strong appeal to anyone interested in high performance analysis of large volumes of data. However, there are contrary perceptions of the benefits of selecting one of these models over the other. Table 1 looks at how each supports the desired characteristics listed earlier in this paper.

Characteristic	MapReduce	High Performance Analytic Database
Data volumes	<ul style="list-style-type: none"> <li>• Can handle petabytes (or possibly scale up to greater orders of magnitude)</li> </ul>	<ul style="list-style-type: none"> <li>• Can handle terabytes and can scale to petabytes</li> </ul>
Performance and scalability	<ul style="list-style-type: none"> <li>• Automatic parallelization allows linear scaling, even with greater numbers of nodes</li> <li>• Communication (phase switch from Map to Reduce) is potential performance bottleneck</li> <li>• When application is not collocated with the data, the channel for loading data into the application becomes a potential bottleneck</li> <li>• Incrementally adding nodes is easy</li> </ul>	<ul style="list-style-type: none"> <li>• Designed for rapid access for analytic purposes (queries, reports, OLAP)</li> <li>• Shared-nothing approach provides eminent scalability</li> <li>• Direct operation on compressed columnar data improves performance</li> <li>• Compression decreases amount of data to be paged in and out of memory, and consequently, disk I/O</li> </ul>
Data integration	<ul style="list-style-type: none"> <li>• Supports structured, unstructured, and streaming data</li> <li>• Potentially high communication cost at transition between Map and Reduce phases</li> </ul>	<ul style="list-style-type: none"> <li>• Supports structured data</li> <li>• Supports real time analytics</li> <li>• Less amenable to integration with unstructured data</li> </ul>
Fault tolerance	<ul style="list-style-type: none"> <li>• MapReduce model is designed to withstand worker failure without restarting a process.</li> <li>• MapReduce often involves larger clusters of 50-4000 nodes</li> </ul>	<ul style="list-style-type: none"> <li>• Generally assume infrequent failures, and rely on underlying fault-tolerance techniques (e.g., RAID or replication).</li> <li>• Small and medium size (8-64 node) clusters are less likely to experience failures than clusters with hundreds of thousands of nodes.</li> </ul>
Heterogeneity	<ul style="list-style-type: none"> <li>• Can potentially (but not typically) be deployed across a heterogeneous cluster</li> <li>• Can be deployed in a cloud</li> </ul>	<ul style="list-style-type: none"> <li>• Can potentially (but not typically) be deployed across a heterogeneous cluster</li> <li>• Performance may suffer when deployed on heterogeneous cluster</li> <li>• Can be deployed in a cloud</li> </ul>
Knowledge delivery	<ul style="list-style-type: none"> <li>• Is a programming model</li> <li>• Requires external repository for comparisons with previous analyses</li> <li>• Executes in batch</li> <li>• Not intended for interactive access</li> </ul>	<ul style="list-style-type: none"> <li>• Provides interactive access (ad hoc queries)</li> <li>• Provides interface to business intelligence front ends</li> <li>• Feeds analytic tools</li> <li>• Supports comparisons and trends</li> </ul>

Table 1: Comparing support for desired characteristics

## Differentiation

The issue boils down to a simple differentiation: the two approaches are intended to support different business analytics needs. The MapReduce approach is a programming paradigm supporting the design and implementation of analysis algorithms and their subsequent deployment across a massive cluster. It is intended to batch process large volumes of many different types of data (including data stored in databases) to produce the multi-dimensional information used (and potentially fed back into the same databases) for analysis. MapReduce is not a database system, nor is it meant to be a replacement for a database system.

Alternatively, the analytical database paradigm, which is designed to manage structured information within a data warehouse, is intended to serve direct interaction with the business analysts and other consumers of business intelligence. It provides tools for ad hoc queries, generated reports, visualization, and may potentially embed data mining models and algorithms, data modeling, and other features. The analytical database allows the analyst to drill down into the data, and interact directly; it is not a programming model, nor is it meant to be a replacement for a programming model.

## Considerations for Deployment

From the perspective of the MapReduce-based approach, one might think that the inherent scalability, fault-tolerance, and simple, yet flexible computing framework are more suitable for analyzing large sets of data. On the other hand, parallel databases are engineered for efficiency and speed of storage and access to support reporting and analysis, and this approach might be perceived as the most appropriate choice.

Perusing the comparison in Table 1, though, you start to see that although the two paradigms differ, there are many ways in which the two approaches are similar. Instead of pitting one approach against the other as if their use were mutually exclusive, one can see that a combination of the two paradigms provides synergy for applications that rely on rapid analysis that needs to be updated and integrated back with a structured environment in order to meet real-time analytical needs. Importantly, there are some specific areas in which the two approaches complement each other:

- **Data Integration** – The analytical DBMS is very good at handling structured data, but is less so when it comes to integrating unstructured and streamed data. The MapReduce approach can fill this gap effectively, providing services to analyze and summarize structured, unstructured, and streamed (or transient) data, with the results forwarded back to the ADBMS for business analyst and end-user drill-down and review.
- **Scalability** – MapReduce provides linear scalability across large numbers of nodes, and in the right circumstances (i.e., in a shared-nothing columnar implementation), so does the ADBMS. By partitioning a cluster to allow some nodes to run the ADBMS and others to run MapReduce applications, the business applications can continue to scale simply by adding additional resources.

- **Communication** – Similarly, both paradigms incur a cost when it comes to loading data external to the cluster. Again, co-locating both approaches within the same cluster clears the data movement bottleneck and leads to a reduction in communication costs.
- **Analytic support** – The most compelling synergy is based on the differentiation noted earlier. Combining these approaches together provides batch computation when needed for the power analysts, especially for massive data sets, while enabling interactive analysis and drill-down for more general business activities.

## Complementary Execution

As mentioned, when using a shared-nothing approach for the analytical database, the synergy can be increased by a collaborative allocation of resources across an MPP architecture. Assigning one cluster of nodes for use by MapReduce applications and another cluster for the analytical database reduces the overhead associated with data movement and communication, since they can both piggy-back on top of the existing network. The batch application can be fed by external data coupled with data in the database, and the results can be streamed right back into the database.

This suggests how these two paradigms complement each other in two ways:

- 1) Environments that depend on collaboration between the power users performing complex analyses on large data sets and the business users exploiting the results. In this situation, a MapReduce application can execute the complex analysis using data sourced from the analytical database management system. Any valuable results can be communicated back (perhaps even using the high performance database) to the more typical business users who will drill-down into the data using standard SQL queries.
- 2) In environments in which the standard SQL operations are insufficient to satisfy the data consumer's needs (either because of complexity, dependence, or scale), MapReduce applications (or individual *Map* or *Reduce* functions) can be directly embedded within queries.

## Examples

Here are some examples of applications that can exploit the complementary nature of using both an ADBMS and MapReduce:

- **Time Series Analysis** – Organizations with a significant amount of transaction or interaction history and different types of data consumers may want to analyze the data to identify interesting patterns that are indicative of business opportunities. In turn, these patterns can be used proactively by data consumers to signal when the opportunity emerges in real-time. For example, in the financial services industry, quantitative analysts can develop MapReduce applications that use the time-series data in the analytical DBMS to look for profitable trading patterns. Any patterns that are found can be shared with the business analysts who can re view and validate the patterns through interactive queries with the database. Successful patterns can be turned over to traders, who can take action when those patterns emerge in real time.
- **Continuous Aggregation** – One immediate abstract example is using MapReduce for the periodic aggregation of continuously updated data, such as web log statistics or social media interactions. The actual streamed data can be maintained in its original form using a persistent storage service (such as HDFS); current aggregate counts and averages maintained within the database can be accessed as input as well, and the aggregations resulting from the MapReduce application managed within a high performance database for analysis or even operational purposes. This enables analysts to drill down at different levels of aggregation within the database, yet still allows branching out of the database to look at the original source data.
- **ETL** – It is often said that the bulk of the work of instituting a data warehouse involves data extraction, integration, and consolidation. A large part of that effort involves extraction,

transformation, and loading (ETL) of data into the warehouse. To some degree, the data transformations are largely independent, and when applied to large data sets, are amenable to massive parallelism, making it a good candidate for combining a MapReduce batch job with the use of an analytical database for analysis.

- **Real-time embedded analytics** – From enhancing operational activities to complex event processing, combining the results of analytics with continuous applications can add value to the bottom line. Some examples include adjusting call-center scripts based on customer profiles and immediate responses, modifying delivery routes based on real-time inventory data collected from thousands of stores, traffic, and weather, or adjusting the power grid in relation to individual residential usage. All of these examples rely on analyzing large amounts of continuous data and coupling the results with existing data to make operational decisions.
- **Large-scale graph and network analysis** – Social network environments demonstrate the utility of managing connectivity. However, a significant amount of knowledge can be extracted through social network analysis. Each individual entity represents a node in the graph, and each connection represents an edge. Applying graph analysis algorithms enables visibility into profile characteristics associated with key individuals in the network, or those that play certain types of roles (such as a “connector,” linking many different sub-groups together. As social networks grow, both in number of entities and (geometrically) in the number of edges, traditional machines become insufficient to manage the massive data structures. Implementing the analysis in a MapReduce application provides the scalability necessary as the network grows, and appending the resulting entity characterizations will enhance an individual’s profile, whether the purpose is direct marketing, behavior analysis, or analyzing terrorist activity.

## Integration Patterns

Since the two paradigms are in many ways complementary, it is valuable to consider the ways the two approaches can be integrated, such as:

- **ETL/Data Warehouse** – In this pattern, the MapReduce application acts as the transformation engine, taking the extracted data and preparing it for integration into the data warehouse implemented using an analytical DBMS.
- **Data Warehouse/Data Marts** – here, the MapReduce/Hadoop application manages the data as a back office warehouse feeding individual high performance data marts using the analytical database.
- **Data Warehouse/Archiving and Persistent backup** – Data in the analytical database can be archived to a Hadoop implementation and maintained in a useful state for longer periods of time.
- **Persistent data store/Batch Analyzer** – Large data sets managed within the analytical database are used as input to batch analysis algorithms, with the results fed back to the analytical DBMS.

## Summary

As businesses transition to become driven by data, information, and actionable knowledge, the volumes of data and complexity of computation will increase exponentially in order to increase the competitive edge. In order to meet these increased computational and data management requirements, scalable tools must be adopted and internalized as core competencies. Emerging tools such as analytic database management systems and MapReduce are becoming commonplace among those organizations that analyze large data volumes.

While these two approaches are not replacements for each other, together, the two types of MPP technologies provide a synergy for emerging classes of applications that can benefit from their collaborative use. Alternatively, organizations that do not recognize the value of blending analytical paradigms may decrease their ability to identify and respond to unexpected changes or blossoming business opportunities in a timely manner.

## About the Author

David Loshin, president of Knowledge Integrity, Inc, ([www.knowledge-integrity.com](http://www.knowledge-integrity.com)), is a recognized thought leader and expert consultant in the areas of data quality, master data management, and business intelligence. David is a prolific author regarding BI best practices, via the expert channel at [www.b-eye-network.com](http://www.b-eye-network.com) and numerous books and papers on BI and data quality. His book, "Business Intelligence: The Savvy Manager's Guide" (June 2003) has been hailed as a resource allowing readers to "gain an understanding of business intelligence, business management disciplines, data warehousing, and how all of the pieces work together." His book, "Master Data Management," has been endorsed by data management industry leaders, and his valuable MDM insights can be reviewed at [www.mdmbook.com](http://www.mdmbook.com).

David can be reached at [loshin@knowledge-integrity.com](mailto:loshin@knowledge-integrity.com).

## Bibliography

Abouzied, Bajda-Pawiloski, Abadi, Silberschatz, Rasin, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads," VLDB 2009

Chen, Shimin, Schlosser, Steven W., "Map-Reduce Meets Wider varieties of Applications," IRP-TR-08-05, Intel Corporation

Dean, Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI 2004

Gartner report G00169004 "Market Share: Relational Database Management System Software by Operating System, Worldwide, 2008" Colleen Graham, Bhavish Sood, Hideaki Horiuchi, Dan Sommer, (see <http://www.gartner.com/DisplayDocument?id=1018712>)

Hinchcliffe, Dion, "10 Ways to Complement the Enterprise RDBMS Using Hadoop," [http://www.ebizq.net/blogs/enterprise/2009/09/10\\_ways\\_to\\_complement\\_the\\_ente.php](http://www.ebizq.net/blogs/enterprise/2009/09/10_ways_to_complement_the_ente.php)

IDC, "IDC Finds Cloud Computing Entering Period of Accelerating Adoption and Poised to Capture IT Spending Growth Over the Next Five Years," October 2008, IDC press release <http://www.idc.com/getdoc.jsp?containerId=prUS21480708>

Pariseau, Beth "IDC: Unstructured data will become the primary task for storage," reference to IDC survey at [techtargt.com](http://techtargt.com), October 2008

Vertica, "The Vertica Analytic Database Technical Overview White Paper," Vertica white paper (see [www.vertica.com](http://www.vertica.com))

Vertica, "The Future of Data Management," Vertica white paper (see [www.vertica.com](http://www.vertica.com))

Winter, Richard, "Why Are Data Warehouses Growing So Fast?" – B-eye-network.com, April 2008