Data Virtualization in the Time of Big Data

A Technical Whitepaper

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

September 2017

Sponsored by

Table of Contents

1 Introduction

Introduction – Big data has raised the bar for data virtualization products! Until now data virtualization servers have focused on making big data processing *easy*. They can hide the complex and technical interfaces of big data storage technologies, such as Hadoop and NoSQL, and they can present big data as if it's stored in traditional SQL systems. This allows developers to use their existing skills and to deploy their traditional ETL, reporting, and analytical tools that all support SQL. Additionally, the products can extend the data security mechanisms for accessing and processing big data across multiple big data systems. But with scale and performance rising, making big data processing easy is not enough anymore. As such, the next challenge for data virtualization is parallel big data processing.

The next challenge for data virtualization is parallel big data processing.

Parallel Processing + Query Pushdown = Parallel Pushdown – Most big data technologies process big data fast, because they make use of their massively parallel architectures. They are capable of distributing data across hundreds of nodes and of processing all that data in parallel. This whitepaper describes two important techniques for processing requests on big data: *parallel processing* and *query pushdown*. Both parallel processing and query pushdown are the key instruments to boost the performance of big data systems. Data virtualization servers have always supported advanced forms of query pushdown for all kinds of data sources including big data storage technologies. To really support fast big data processing they have to support parallel processing as well. In fact, parallel processing must be combined with query pushdown to form *parallel pushdown*.

Parallel processing + query pushdown = parallel pushdown.

Cisco Information Server Goes Parallel – *Cisco Information Server* (CIS) has extended its architecture to follow in the footsteps of these big data technologies and has introduced a massively parallel architecture as well. Now, CIS can also distribute data processing across hundreds of nodes in a similar way as the familiar big data technologies. It can make big data processing *easy* as well as *fast*.

The new architecture of *Cisco Information Server* version 8 encapsulates significant big data processing advancements. Its new MPP query engine allows for requests to run in parallel and, combined with the query pushdown capabilities, exploits the full power of big data technologies. This whitepaper describes how Cisco Information Server version 8 has implemented these two key performance boosting instruments: parallel processing and parallel pushdown. Further, it lists multiple big data use cases that these advancements can now enable.

2 Big Data Storage Technologies

Following a brief introduction, this section describes the key techniques that big data storage technologies apply to improve performance and scalability. Topics include data partitioning, parallel processing of requests, special programming interfaces, specialization, and "polyglot persistence."

Big Data Has Arrived – The big data concept was introduced at the end of the 1990s and has, by now, been embraced by all kinds of organizations. Data can be "big" because of its enormous volume, because it's

not structured in the traditional way, or because it's streaming in with enormous quantities. The business advantages of big data systems are clear, they can improve, deepen, and strengthen an organization's analytical capabilities with relatively attractive infrastructure economics.

The technology is available to store, process, and analyze big data. Systems such as the *Hadoop* platform and numerous *NoSQL* products have been designed and optimized to work with big data. With specialized analytical tools all that data can be analyzed fast and efficiently, self-service data preparation tools help business users and data scientists understand big data by applying artificial intelligence techniques. In addition, architectures exist, such as data lakes, to help organize and process big data.

Technique 1: Parallel Processing of Big Data Requests – Crucial for big data technologies is that they can process requests on large data sets fast. Therefore, they apply several techniques, one is *parallel processing*. When all the data of a file is stored on one disk and only one process can access the file, access is *serial*; see Figure 1. Two records of that same file are never processed simultaneously, only serially. Processing real *big* data sets serially takes too long. The first technique to improve the performance of requests on big data was introduced years ago in SQL systems and is called *file* or *data partitioning* and is the basis for *parallel processing*. This technique has also been implemented in big data technologies.

Processing big data sets demands parallel processing.

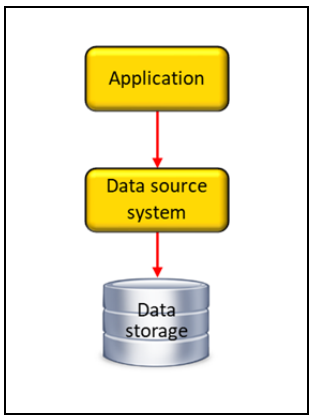

Figure 1 *When all the data of a file is stored on one disk and only one process can access the file, data access is by definition serial and not parallel.*

To support parallel processing, big data technologies support data partitioning and their internal architectures consist of one *master* (or a few) and several *worker* processes; see Figure 2. Application requests are received by the master process. The master knows that the data is partitioned and divides the request into a set of *subrequests*. These subrequests are send to the workers on the nodes that contain the data. Next, each worker processes its own partition of the data. In other words, the master starts up multiple processes to process the partitions in *parallel*. When the workers are finished, they return the intermediate results to the master. Then the master combines all the results from the workers and returns it to the application. Note that applications are not aware of this parallel processing, except that they will experience a much faster performance than with serial processing of the same data.

Technique 2: Massive Parallel Processing – For a long time, data partitioning and parallel processing features offered sufficient performance improvements, until big data came along. Purely the sheer volume of big data demands a level of parallelization higher than was common. Unfortunately, most SQL systems can only parallelize processing efficiently across a limited number of nodes. The Hadoop platform and the NoSQL products, on the other hand, are designed specifically to support *massive parallel processing*. They

can distribute data and processing across hundreds of disks and nodes and have a similar number of workers running in parallel. Their internal architectures make these systems *big data-ready*.

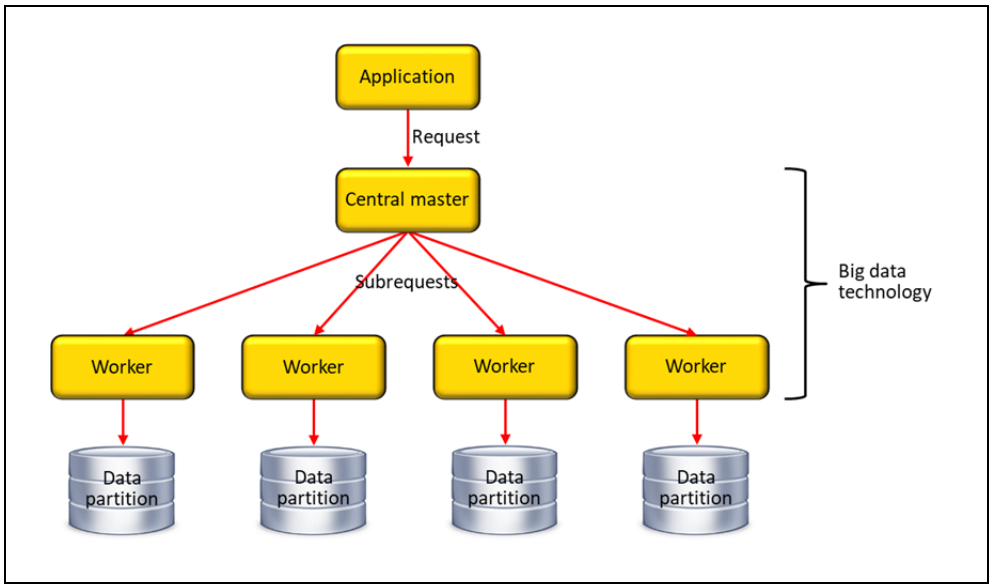

Figure 2 *Big data technology is able to process requests in parallel by dividing them in subrequests and sending them to the workers for parallel processing.*

Technique 3: Non-Standard Programming Interfaces – Big data technologies support application programming interfaces that are suitable for developing big data systems. The consequence is that they don't support well-known interface standards, such as SQL and SOAP. They all support proprietary and quite technical interfaces that can only be deployed from within programs developed in languages, such as Java, C#, or Python.

In addition, these systems all support different programming interfaces. Products such as Apache HBase, Cassandra, and MongoDB, all support their own native API. No standards exist for the programming interfaces of current big data technologies. There is no equivalent for SQL in the NoSQL world.

Technique 4: Specialization – Another technique used by big data technologies to improve performance is *specialization*. Traditional SQL database servers, such as Microsoft SQL Server and Oracle, are so-called *generic* database servers. They are good for almost any type of application, whether it's a transaction system, a portal, or a reporting environment. But they don't really excel at anything.

By contrast, to make them as fast as possible, big data technologies are designed specifically for a limited set of use cases. They are *specialized* database servers. They can do a few things extremely well, but not everything. For example, Apache Spark was built for advanced analytics on big data, whilst Apache HBase was designed to support a massive transactional workload. Again, this specialization is needed to make them suitable for big data systems. Compare this to cars. Traditional SQL database servers are like traditional sedans; good for everything. However, to drive very fast, a special type of car is preferred, for example a Ferrari 458, and for transporting a large family with piles of bags another specialized car is more appropriate, such as the Dodge Grand Caravan. The traditional car is a generic car, while the Ferrari and the Dodge are specialized cars. Big data technologies are like Ferrari's and Dodge's.

Big data technologies are specialized technologies.

The effect of specialization is that big data technologies don't only differ with respect to their programming interfaces, but also with respect to their potential use cases. This can lead to a situation in which organizations use different big data storage technologies for different use cases; for example, MongoDB for running transactions and Apache HDFS for storing sensor data. This use of different data storage technologies is referred to as *polyglot persistence*[1]. Working with specialized systems works well until their data must be integrated, for example, for reporting or analytics. The developers will have to deal with this multitude of programming interfaces. They must understand and be experienced with all of them.

3 Current Data Virtualization Features Make Big Data Processing Easy

Until now, *data virtualization* has brought several features to the big data table to help organizations adopt and process big data more easily.

Simplifying Transforming Native and Technical Programming Interfaces – Data virtualization servers have always allowed developers to use well-known and standardized interfaces, such as SQL and SOAP, to access a wide range of data sources using complex, technical, and mostly proprietary interfaces, such as Excel spreadsheets, flat files, and mainframe systems. Nowadays, data virtualization servers also allow SQL or SOAP developers to access the proprietary interfaces of big data technologies. The benefits are that they don't have to learn new programming interfaces, most reporting and analytical tools can be used to access big data, and it unlocks big data to a wide range of developers, from BI developers to IT specialists.

Simplifying Polyglot Persistence – Dealing with polyglot persistence is simplified with data virtualization. All the different programming interfaces can be hidden to the developers. Developers don't have to work with a multitude of different interfaces and different big data technologies. Instead, all the big data sources are presented as one integrated database accessible through one interface. While maintaining the parallel processing power of those big data technologies, data virtualization simplifies the development of applications dealing with all these diverse data storage technologies easier. Similarly, it also reduces the risks for an organization to deploy an additional specialized big data storage technology for a new big data system with a special use case.

Seamlessly Integrating Big Data with Traditional Data – Data virtualization simplifies integration of big data with data stored in data warehouses or transactional databases developed with traditional database technology. In fact, the entire set of data sources, including the big data stores, can be presented to the developers as one integrated database. Because of the *data federation* capabilities of data virtualization servers, developers won't even know that some of the data is stored in big data systems and some isn't. The benefit is that developers can, for example, integrate historic data (stored in a SQL-based data warehouse) with streaming data stored in Hadoop, and can combine descriptive data stored in data marts with sensor data kept in MongoDB.

Enhancing Data Security Capabilities for Big Data – Data virtualization servers offer extensive data security mechanisms for accessing data sources in general, including the big data systems. With a data virtualization server one integrated data security layer can be defined on a heterogeneous set of data sources. The data security features offered by data virtualization servers are much more extensive and

[1] Wikipedia, August 2017, Polyglot Persistence; see https://en.wikipedia.org/wiki/Polyglot_persistence

more detailed than those of most big data technologies, allowing big data to be secured in a way that's not possible with that technology itself. In addition, security specialists only have to deal with one tool for specifying data access rules for a wide range of data sources.

To Summarize – Data virtualization has enriched the big data world with features that makes working with it much easier, and in addition, it fills some important functionality gaps. But until now, data virtualization has not focused on speeding up big data processing.

4 Parallel Processing and Query Pushdown: Two Sides of the Same Coin

Is Parallel Processing Sufficient? – As indicated in Section 2, the basis of parallel processing in a big data environment is the partitioning of data across many nodes. Records of one file or table are distributed across many nodes and disks. Now, spreading data of one file across *x* partitions, doesn't make the request go *x* times faster. There is always some *overhead*. For example, combining the intermediate results by the master takes some time and sending data from the workers to the masters takes time as well. But the most important aspect that determines how much or how little overhead there is, is how much of the request is handled by the workers and how much by the master. Processing done by the workers is done in parallel and processing done by the master is done serially. The more work that is done by the workers in parallel and not by the master serially, the faster it is.

Parallel processing and query pushdown are both needed to guarantee high performance.

Conclusion: Parallel processing by itself is not enough to guarantee high performance. A second aspect is as important: *query pushdown* or simply *pushdown.*

The Importance of Query Pushdown – The concept of *query pushdown* is directly related to the master process breaking up incoming requests into subrequests which are processed by the workers. The big challenge in this architecture is to let the workers handle most of the request processing and to let the master do as little as possible. The reason is simple, work executed by the workers is done in parallel and the work executed by the master is done serially by one single process. A system offers a high level of parallel processing if it's able to *push* most of the processing to the workers. In other words, if it is able to delegate as much of the processing to the subrequests. Platforms such as Hadoop offer these features to push processing to the workers.

Pushing Down Operations to the Workers – Each request is made up of a set of *operations*, such as filters, joins, aggregations, and projections. With query pushdown, the goal is to push the execution of as many request operations as possible to the workers, because the more operations are processed by the workers, the more is processed in parallel. This makes a system more suitable for big data processing. For example, if the workers of a big data system can only process some of the request's operations, performance can still be bad, because too much of the processing is executed serially by the master and not in parallel by the workers; this is illustrated in Figure 3. The different colored dots represent different operations that make up the request. The entire request consisting of all the operations is send to the master. Here, only two of the eight operations (the black and purple dots) are processed in parallel, six of them are not pushed down and are executed serially, which is far from efficient.

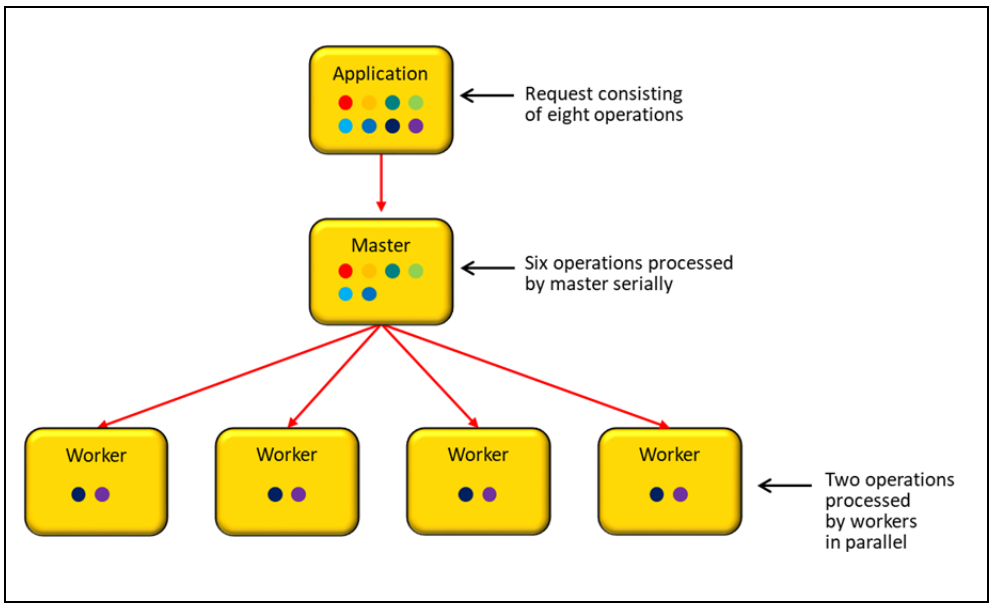

Figure 3 *In this architecture only two of the operations making up the application's request are pushed down to the parallel worker processes and six are executed by the master process resulting in bad performance.*

Figure 4 visualizes a system with a higher pushdown level. Here, six of the eight operations are pushed down and only two (the red and orange dots) are processed serially.

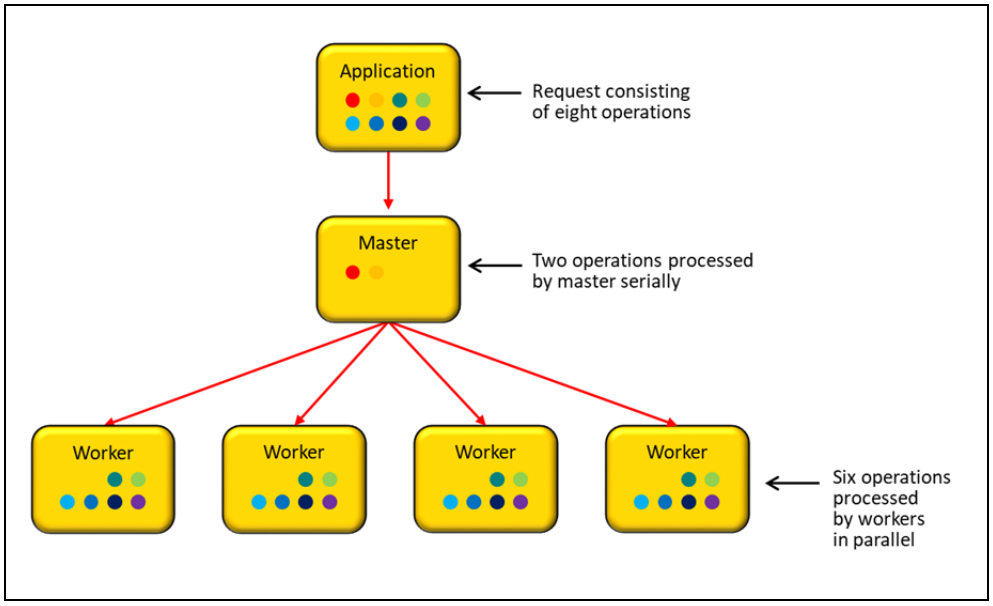

Figure 4 *In this architecture six of the operations making up the application's request are pushed down to the parallel worker processes and only two are executed by the master itself.*

5 Data Virtualization and Query Pushdown

Data Virtualization Supports Query Pushdown – Data virtualization servers have always supported extensive query pushdown capabilities. Their goal is to push the processing as close to the stored data as possible. They take the incoming request and determine how many of the request's operations can be pushed down to the data source. If some operations of the request processing can't be pushed down, the data virtualization server executes them itself. Note that the capabilities of the data source system determine whether the request pushed down by the data virtualization server is executed in parallel.

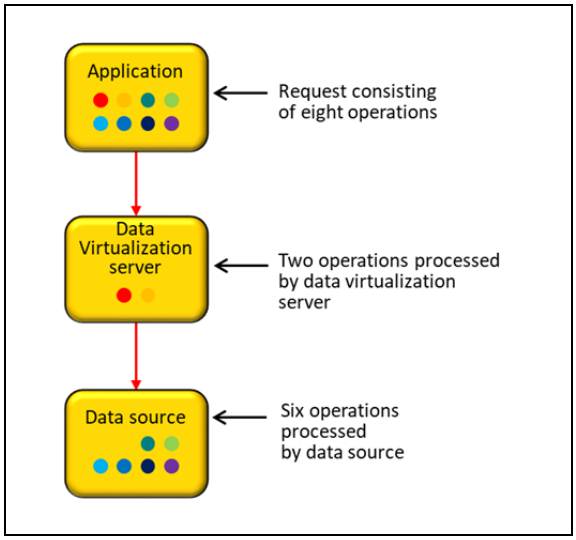

Figure 5 *In this example, six of the operations are pushed down to be executed by the data source, and two are executed by the data virtualization server.*

Why Data Virtualization Requires Parallel Processing – Data virtualization servers support advanced pushdown, but offers no parallel processing capabilities. They normally operate the same way as the master process of a data source system does: execution is primarily serial, not parallel.

The fact that a data virtualization server does not support parallel processing is not always a big problem. For example, if a data virtualization server can push down most of the request's operations to a data source system that supports parallel processing and query pushdown, the requests are executed efficiently.

A problem arises when the data source system *cannot* pushdown many operations of the requests to its worker processes: there will be parallel processing, but very limited query pushdown. In this case, the data virtualization server can't improve the performance, not even when it supports parallel processing. It's the data source system that is the bottleneck. But there are situations in which parallel processing by the data virtualization server can speed up processing dramatically. In fact, to scale in big data environments and to become a *big data virtualization server*, the products must combine query pushdown *with* parallel processing.

6 Data Virtualization: Parallel Processing + Query Pushdown = Parallel Pushdown

A Parallel Architecture for Data Virtualization Servers – To step up from data virtualization to *big data virtualization*, next to query pushdown, support for parallel processing is a requirement. An internal architecture comparable to the one presented in Figure 6 is required. Incoming requests are studied by the master process of the data virtualization server and a decision is made whether they can be executed in parallel. If possible, the requests are broken into subrequests which are executed by several data virtualization worker processes. This way, the processing of operations is pushed down to the data virtualization workers, who are then responsible for pushing as much of the processing down to the data source.

This combination of parallel processing and query pushdown can be called *parallel query pushdown* or simply *parallel pushdown*. Parallel pushdown makes data virtualization servers more suitable for big data.

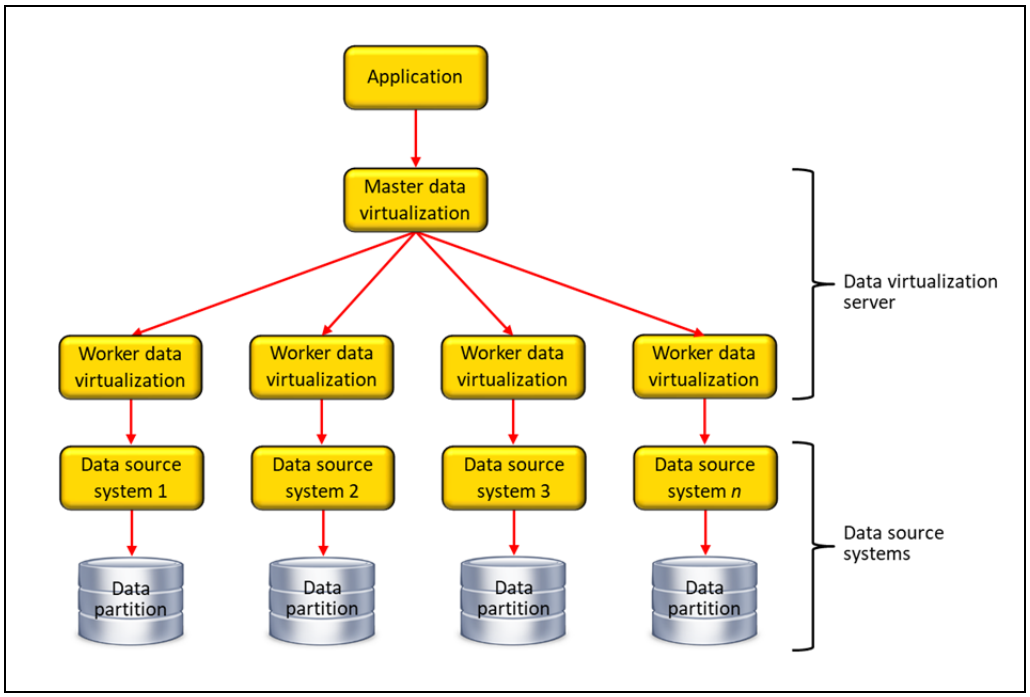

Figure 6 *With parallel pushdown the concepts of parallel processing and query pushdown are combined by data virtualization servers.*

Different forms exist where parallel pushdown can be used by data virtualization servers:

Form 1: Parallel Execution on Data Source without Query Pushdown – As indicated, some new big data storage platforms support data partitioning but only very limited query pushdown capabilities. In this case, a data virtualization server is able to use parallel pushdown to avoid a performance bottleneck. The full power of the data storage platform is then utilized; see Figure 7.

Note that if a data source system has not partitioned its data, and does not support parallel execution of requests, there is no need for the data virtualization server to use its parallel pushdown capability. A powerful pushdown feature is more important.

Form 2: Parallel Big Data Federation – Not all big data is stored in one system. It can be that big data is distributed across multiple Hadoop clusters, or that it's kept in different big data technologies (polyglot persistence). A data virtualization server needs to support parallel pushdown across multiple platforms, as depicted in Figure 6.

For example, an incoming request is a join of five tables. Each table is stored in a separate big data source. All these data sources can process certain operations in parallel. There are two data virtualization worker processes. When the master data virtualization process receives the five-table join, it divides the request into two requests and sends them to the two data virtualization worker processes. The workers decide which operations can be send to the data sources. The effect is that all the data source systems work in parallel and the two data virtualization workers work in parallel. This speeds up this federation of big data systems.

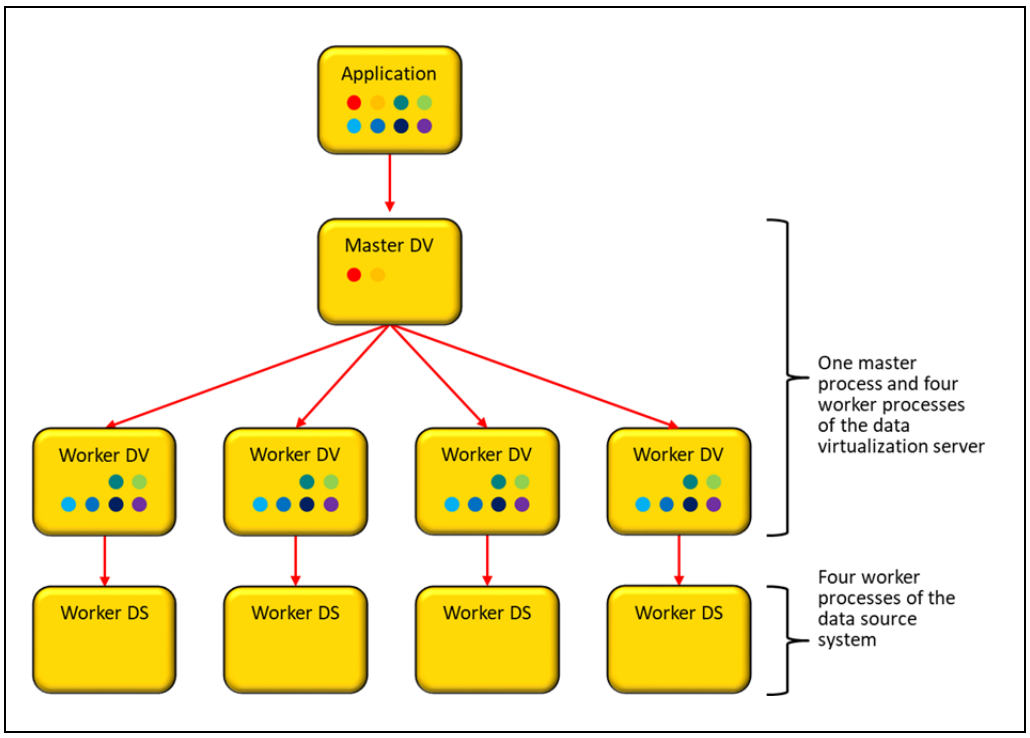

Figure 7 *A data virtualization that supports parallel pushdown can speed up the performance of a big data system that supports paralle processing but only a limited form of query pushdown.*

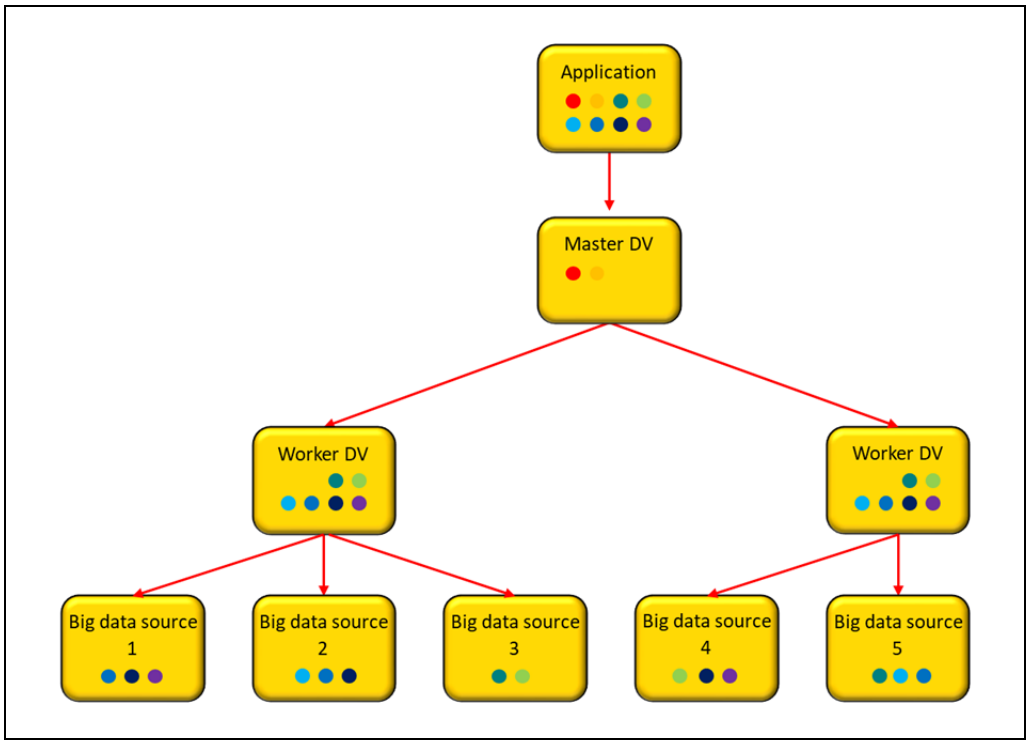

Figure 8 *The master process of the data virtualization server uses parallel pushdown to distribute the processing across its workers. The workers do the same by pushing the processing to the big data sources.*

Form 3: Parallel Heterogeneous Data Federation – There is still a massive amount of data stored in data warehouses, data marts, and transactional systems that is not implemented with new big data technology. For example, a large part is still stored in SQL systems. In such situations, the word "big" doesn't refer to the huge amount of data in one data source, but to the total amount of all the data distributed across all the data sources.

As with the previous form, a data virtualization server supporting parallel pushdown can let all the data sources, big data technology or SQL-based, work in parallel to speed up the overall processing time of the requests. In other words, the federation of a heterogeneous set of data sources can be processed efficiently with parallel pushdown.

Form 4: Parallel Data Federation of Cloud-Based and On-Premise Data Sources – Most organizations are in the process of migrating their data stores to the cloud. But only a few have succeeded fully or have not done anything at all. The effect is that when data needs to be integrated, on-premise and cloud-based data sources must be integrated. Data virtualization servers support this capability, they can make all the data stores (cloud-based or not) look like one logical database. With query pushdown, the data sources are accessed one by one. This can be slow, especially with the potential network delay. With parallel query pushdown, however, all the data sources, both on-premise and in the cloud, are accessed and operate in parallel.

7 Big Data Virtualization Use Cases

The combination of query pushdown and parallel processing enables virtualization servers to deliver the scale and performance required for big data processing. First, data virtualization made big data processing easy to use, and now it's making it fast. A data virtualization server doesn't have to be the bottleneck anymore in a big data system, because it uses the same architecture deployed by big data storage technologies. All in all, it makes data virtualization suitable for a wider range of use cases. This section describes some of the new use cases.

Instant and Fast Analysis of Big Data – Due to their proprietary interfaces, many analytical tools can't access the big data storage technologies. To analyze this data, it must be copied first to another data source that supports an interface accessible by the analytical tool. Unfortunately, if the analytical tool needs instant access, a time-consuming copying process is not an option. A data virtualization server can offer a supported interface on the big data source, but such a product with no support for parallel processing will not speed up performance. It will exploit its pushdown capabilities.

Parallel pushdown allows data virtualization servers to give instant and fast access to big data stores without the need to copy the data. Parallel pushdown really unlocks big data sources on demand for instant requests whilst maintaining high performance.

Parallel pushdown allows data virtualization servers to give instant and fast access to big data stores.

Easy Staging of Big Data – Data virtualization servers support *caching* of virtual tables. With caching a copy of the virtual content of a virtual table is physically stored. Afterwards, every time when the virtual table is queried, the cached table is accessed. If big data is stored in a specialized database server for transaction processing, analyzing it is not trivial. In fact, analyzing can lead to too much interference on the transaction system. With a data virtualization server, big data can be cached in a data source optimized for analysis. In other words, big data is staged in another data storage system. The parallel processing and parallel pushdown capabilities make this an attractive option.

Speeding Up Slow Data Sources – If a data source is too slow for analysis or an analysis exercise creates too much interference on a data source, caching can be used to create a "fast" copy of the data. As with the previous use case, a cache can be defined on this data source. By selecting a system for the cached data

that supports data partitioning and parallel processing, the analytical processing will be much faster than on the original data source.

Simplifying the data lake with data virtualization servers.

Simplifying the Data Lake – Most data lakes are developed with Hadoop technology. According to the definition of data lake, all the data is stored in its original form and keeps its original format. Data virtualization servers have always made it easier for business users to access data lakes, because they can offer one interface to all the data stored in the data lake. Now, with parallel processing, if a data scientist joins data from multiple data sets in the data lake, a large portion of the processing can be done in parallel by the data virtualization server, improving the performance.

Accessing Remote Big Data – Some big data is produced and stored across various sites, such as sensor data generated by offshore oil rigs and sales data produced in stores across the country. To analyze all the data, in most situations, first it has be copied to one centralized data store. But this involves copying big data across WANs and can be slow. As some say: "big data can be too big to move." With parallel pushdown, the request can be send to all the remote sites in parallel. All the sites execute their subrequest and only the results are send back to the data virtualization server. The benefits are that it minimizes the amount of network traffic and thus speeds up query performance, there is no need to design, test, run, and manage a centralized environment that holds all the data, and the data being accessed has no latency.

Accessing Offloaded Cold Data – Some organization offload their cold data from the data warehouse to a separate data source. This keeps the data warehouse smaller (and faster) and allows for the storage of cold data on a cheaper storage platform. Data virtualization servers can hide the fact that a table has been divided in two physical tables, one with the warm and one with the cold data. If users still access both data, the two data sources with different temperatures are accessed in parallel.

Processing Traditional Data Sources in Parallel – If data is distributed across more traditional data sources that don't support parallel processing, the combination of parallel processing and query pushdown can still be applied by the data virtualization server. Instead of accessing all the data sources one by one, or by moving data from one data source to another, parallel processing can be applied. Requests are divided into a number of subrequests and these subrequests are pushed down to the traditional data sources who then process these subrequests in parallel.

8 The MPP Architecture of Cisco Information Server Version 8

Cisco Information Server version 8 offers a massively parallel architecture.

Cisco Information Server version 8 (CIS) is the first data virtualization server offering a massively parallel architecture, or in other words, it supports parallel pushdown and parallel processing on a heterogeneous set of data sources. This is a major re-architecting of CIS.

The New Architecture of CIS – Figure 9 presents a high-level view of the new CIS version 8 architecture. The system is setup as a number of data virtualization worker processes. Each worker contains all the standard CIS features, including data security, caching, and data federation. Each one also contains the proven data access layer for so many different data sources. What's new is

that each worker houses two optimizers: the existing one, the *Classic Query Engine* (CQE) and the new one called the *MPP Query Engine* (MQE). To the MQE in V8 several new query optimization techniques have been added. The internal architecture of a CIS worker process is presented in Figure 10.

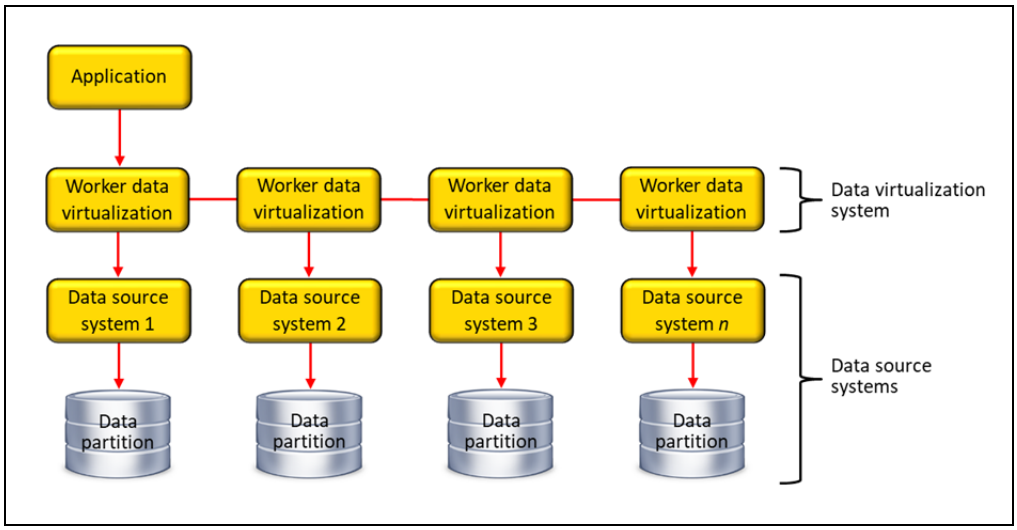

Figure 9 *In the new massively parallel architecture of CIS requests can be executed in parallel by a large number of worker processes.*

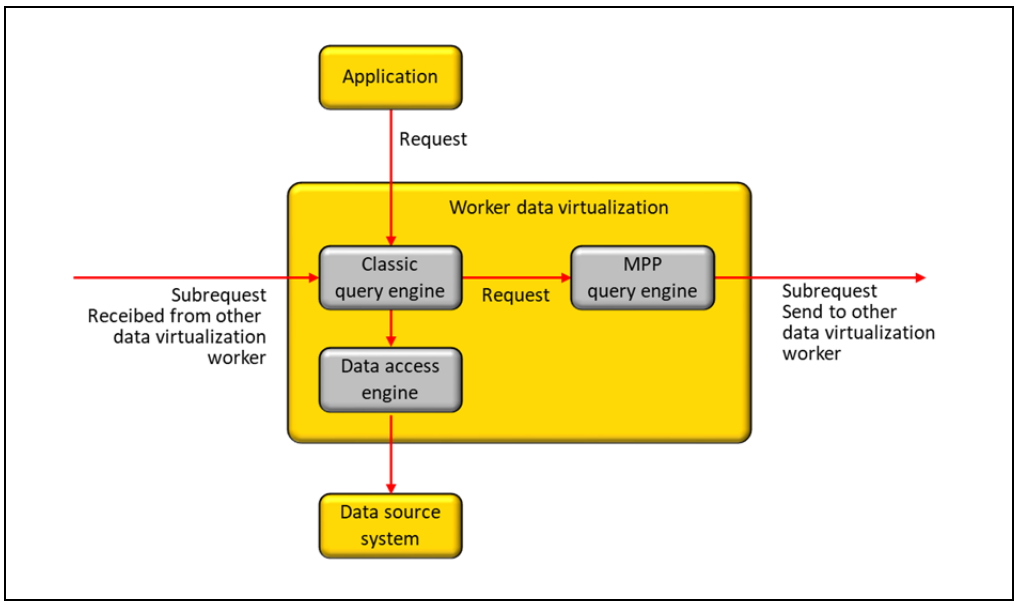

Figure 10 *Each worker process consists of the classic query engine and the massively parallel query engine.*

Statistical Information – The more information query optimizers have on the data to be accessed, the better the generated query execution plan can be. Information on the cardinality of tables or files, the distribution of values in a column, the availability of indexes, and how data is partitioned, is all very relevant input for determining the best query execution plan (QEP). Previous versions of CIS already used statistical information to determine the ideal QEP. CIS version 8 keeps track of more statistical information, especially the data needed for the parallel pushdown. For example, based on available statistical information, the MQE determines whether parallel execution of requests is useful.

The Workings of the Query Optimizer – Every application is "connected" to one worker. The application sends a request to its worker. The query is first evaluated by the CQE. It determines whether the query must be optimized by the CQE itself, or by the new MQE. This decision is based on the amount of the data and on

the data source characteristics. For accessing a few thousands of rows, parallel pushdown is an overkill. A simple QEP is probably faster. It uses the available statistical information to make the optimal decision. If the CQE determines to process the request itself, it's optimized and a QEP is generated that uses as much pushdown processing as possible.

If the request qualifies for parallel execution, it's passed on to the new MQE. The MQE evaluates the request and breaks it up into subrequests that are executed in parallel. Evidently, it tries to use as much parallel pushdown as possible. These subrequests are then send to the other CIS workers who then all start the execution of their subrequests. By the way, when these subrequests are received by the workers, they are optimized by their own CQEs that are responsible for executing the subrequests. Note that because the CQEs can use the full data access layer (read all the adapters), they have access to any type of data source.

Balancing of the Query Workload – In the optimization process, the MQE watches for skewing of the processing. For example, if one data source contains 80% of the data and the other 20%, and if five workers can be used, it's probably best to let four of them process data source one and the fifth worker the other data source. This is probably more efficient than to distribute the workers equally over the two data sources. It's a good example to show how important it is that the MQE needs detailed statistical information to make that decision.

Query Prioritization and Throttling – CIS version 8 supports two advanced features to manage the query workload: *query prioritization* and *query throttling*. The priority of a query influences how many resources are devoted by CIS to the processing. Queries with a lower priority must yield with resource consumption to queries with a higher priority. For example, the query of a batch report probably doesn't need a high priority, whereas a query from an online customer does.

With query throttling the overuse of resources, such as I/O, memory, and processing power, can be prevented. Limits can be set to how many resources a query is allowed to consume. Especially in a big data environment, this can be important. Otherwise, queries on big data may consume so many resources that they slow down the performance as experienced by other users. Query throttling minimizes this risk of interference on other users.

9 Summary

CIS version 8 enriches support for big data processing.

Analytics workloads at greater scale, with higher performance have driven demand for big data processing. The new architecture of *Cisco Information Server* version 8 enriches support for big data processing. The new MPP query engine supports parallel processing and query pushdown. The previous versions of CIS made big data processing easy, and now, extended with features such as parallel pushdown, query prioritization, and query throttling capabilities, it makes big data processing fast.

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, big data, database technology, and data virtualization. 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 annually in London). He writes for Techtarget.com[2], B-eye-Network.com[3] and other websites. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[4] all published at B-eye-Network.com. The Data Delivery Platform is an architecture based on data virtualization.

He has written several books on database technology. Published in 1987, his popular *Introduction to SQL*[5] was the first English book on the market devoted entirely to SQL. After more than twenty-five years, this book is still being sold, and has been translated in several languages, including Chinese, German, and Italian. His latest book[6] *Data Virtualization for Business Intelligence Systems* was published in 2012.

As *Ambassador of Kadenza*, Rick works closely together with the consultants of Kadenza in many projects. Kadenza is a Netherlands-based consultancy company specializing in business intelligence, data management, big data, data warehousing, data virtualization, and analytics. Joint experiences and insights are shared in seminars, webinars, blogs, and white papers.

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 Cisco Systems, Inc.

Cisco (NASDAQ: CSCO) is the worldwide technology leader that has been making the Internet work since 1984. Our people, products and partners help society securely connect and seize tomorrow's digital opportunity today. Discover more at thenetwork.cisco.com and follow us on Twitter at @Cisco.

Cisco® Information Server is data virtualization software that lets you integrate data at big data scale, with breakthrough speed and cost effectiveness. With Cisco Information Server, you can build and manage virtualized views and data services that access, transform and deliver the data your business requires to accelerate revenue, reduce costs, lessen risk, improve compliance and more.

For more information, please visit
http://www.cisco.com/c/en/us/products/cloud-systems-management/data-analytics/index.html#~overview

[2] See http://www.techtarget.com/contributor/Rick-Van-Der-Lans
[3] See http://www.b-eye-network.com/channels/5087/articles/
[4] See http://www.b-eye-network.com/channels/5087/view/12495
[5] R.F. van der Lans, *Introduction to SQL; Mastering the Relational Database Language*, fourth edition, Addison-Wesley, 2007.
[6] R.F. van der Lans, *Data Virtualization for Business Intelligence Systems*, Morgan Kaufmann Publishers, 2012.