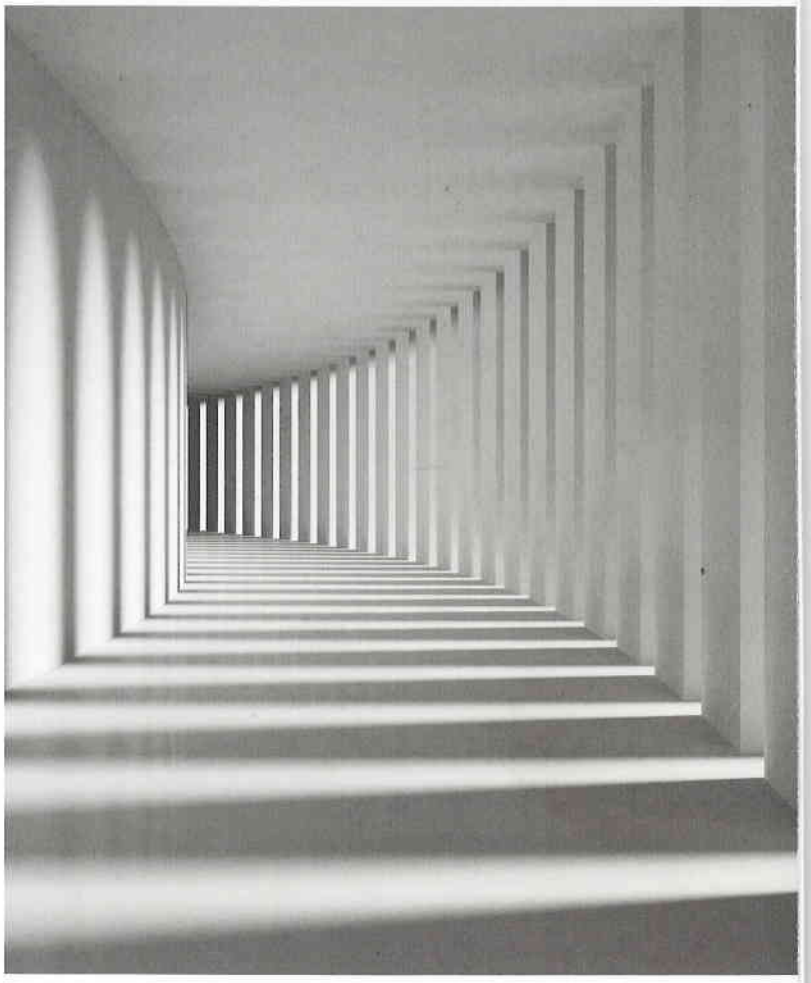
EXCLUSIVELY FOR
TDWI PREMIUM MEMBERS

VOLUME 20 • NUMBER 2

Business Intelligence Journal

THE LEADING PUBLICATION FOR BI, DATA WAREHOUSING, AND ANALYTICS PROFESSIONALS



tdwi
Advancing all things data.

VOLUME 20 • NUMBER 2

Business Intelligence JOURNAL

Price/Performance Considerations in Building Data Warehouses for Big Data

Norman T. Kutemperor

Norman T. Kutemperor is CTO at Scientel Information Technology, Inc.
scientel@scientel.com

Abstract

Information is today's currency and highly automated organizations raise the value of their enterprise with the IT muscle power derived from smart business intelligence—which ultimately comes from the smartest data warehouses. Building on a shoestring—e.g., using commodity hardware—can appear initially to be the most economical approach. However, many factors go into building a data warehouse for big data, and price is just one. This article describes several major considerations that should govern the overall approach for designers of data warehouses for big data.

Introduction

Technological advancements—such as the Internet, smartphones, cloud, mobile, super servers, tablets, social media, imaging, digital sensors/instrumentation, and advanced scanning—have led to today's big data environment and related data-handling issues.

Today, we generate more data in one day than we did in the entire 55 years from 1946 to 1999 *combined*. Big data is generally characterized as arriving in high volume, velocity, variety, and variability. "High" is relative to the organization. One thing is certain: corporate data is at least doubling every two to three years to staggering levels that eclipse the ability of today's ordinary machines and SQL database management systems (DBMSs) to handle cost-effectively and reliably.

Structured data is only 20 percent of today's big data and is easily handled by the popular SQL database (DB).

That's not the case with unstructured data, which makes up the remaining 80 percent.

Conventional SQL platforms that ruled the mainstream DBMS industry during the last three decades—prior to the advent of today's true big data—handled data acceptably or at least could be patched enough to perform in some fashion. However, big data caused SQL-based platform failures at all the Internet giants within the past decade, forcing them to switch to versions of NoSQL DBs to handle big data. This gave birth to the NoSQL industry in 2010. Only NoSQL DBs can properly handle many of today's true big data loads, and in the future, only NoSQL DBs will survive the constantly increasing big data requirements. For this reason, the NoSQL DB market is growing at 82 percent annually.

When the Internet of things (IoT) becomes fully active, most of today's infrastructure will collapse from extreme demand. This calls for ultra-efficient infrastructure uniquely designed for big data in all its forms.

A properly-designed NoSQL database is very good with unstructured data and big data, and it is highly scalable. Because information arrives in many different forms, corporations must organize, store, process, mine, and analyze data rapidly to stay competitive.

Data warehouses are efficient because they run on hardware optimized for the database. Furthermore, the DB must support distinctly different data models for unstructured as well as structured data, which most NoSQL as well as SQL DBs do not support; they are based on a single data model for all data types.

A data warehouse appliance (DWA) is a combination hardware and software product designed specifically for analytical processing. *Data warehouse appliance* is also a marketing term for an integrated set of servers, storage, operating system(s), DBMSs, and software specifically pre-installed and pre-optimized for data warehousing. An appliance allows the purchaser to deploy a high-performance data warehouse right out of the box.

In a traditional data warehouse implementation, the database administrator (DBA) can spend significant time tuning and putting structures around the data to improve performance for large sets of users. With a data warehouse appliance, however, the vendor is responsible for simplifying the physical database design layer and tuning the software for the hardware.

When a traditional data warehouse needs to scale out, the administrator must migrate all the data to a larger, more robust server. When a data warehouse *appliance* needs to scale out, he or she can simply purchase additional plug-and-play components.

A data warehouse appliance comes with its own operating system, storage, DBMS, and software. It uses massively parallel processing (MPP) and distributes data across integrated disk storage, allowing independent processors to query data in parallel with little contention and redundant components to fail gracefully without harming the entire platform.

A large data warehouse appliance (LDWA) is simply what the term implies: a DWA that is inherently constructed, both in hardware and software, to scale more than average in DWA capabilities and applications. This has many implications for the increasingly complex world of big data and related applications.

In 2012, Gartner defined big data as "high volume, high velocity, and/or high variety information assets that require new forms of processing to enable enhanced decision making, insight discovery and process optimization." Additionally, a new V (for veracity) is added by some organizations to describe it.

Gartner's definition (the 3Vs) is still widely used, and the concept is maturing, fostering a greater distinction between big data and business intelligence regarding data and its use:

- Business intelligence uses descriptive statistics with data with high information density to measure things, detect trends, etc.

- Big data uses inductive statistics and concepts from nonlinear system identification to infer laws (regressions, nonlinear relationships, and causal effects) from large data sets to reveal relationships and dependencies and to perform predictions of outcomes and behaviors

Today, organizations are collecting more data than ever but are not equipped to process it effectively. As a result, enterprises are losing out on millions of dollars in revenue. Next-generation analytics must tackle big data sets, and few companies offer integrated performance hardware incorporating high-speed interconnects that deliver unparalleled performance and solutions to all big data loads.

For optimum performance in dealing with big data, advanced LDWAs are needed. However, they come at a cost that is sometimes prohibitive, and some organizations may benefit from Scientel's experience in designing and developing its own LDWAs.

When considering the design or implementation of an efficient data warehouse, pay close attention to the type of technology utilized as well as the efficiency factor of these systems. In many versions of loosely coupled data warehouse systems and commodity-based systems, ample storage and memory may be available. However, performance may lag in spite of the abundance of hardware because of the technology used. Results delivered from today's data warehouses impact every employee from the top down as top-level management makes strategic decisions based on the results delivered from—and only after being delivered from—these systems. Although cost is a key issue, performance is a major factor as well. Following are some of the items that implementers must review before deploying a data warehouse.

Massively Parallel Processing

This is the coordinated processing of a program by multiple processors that works on different parts of the program, with each processor using its own operating system and memory. Typically, MPP processors communicate using some messaging interface. In some implementations, 200 or more processors can work on the same application; and newer systems use many more. An "interconnect" arrangement of data paths allows messages to be sent between processors. Usually, therefore, MPP is much more complicated to set up, requiring extensive and deep thought about how to partition a common database among processors and how to assign work among the processors. In modern technology systems, the MPP concept is applied in a tightly coupled system to deliver ultra-high performance.

In MPP for databases and large data warehouses, proportional components of a database are processed in parallel—that is, essentially simultaneously, in exactly the same batch—by a large number of equal (peer) nodes in the same system or server. This concept does not apply to DB systems where the DB components are only stored and/or managed by nodes that are not peers but slaves and hence not identically and fully processed by them. Designed for high volume and velocity, this concept works well on systems that scale out (NoSQL) rather than up (SQL).

Scale-Up versus Scale-Out

Some systems are well known for their expansion capabilities. In data warehousing, expansion is a term that itself implies limitation. It is synonymous with scaling up, but scaling up is limited to the constraints of a single node or server and will reach some specific limits that are pre-established. Scale-out expansion is the ultimate in preserving investment as well as ensuring that the data warehouse does not hit a brick wall in terms of expansion or obsolescence. A good scalable system can keep expanding without many predefined limits by adding new nodes of full server systems and be able to run the same or similar versions of the operating system and DB as data needs expand.

When the amount of data increases, standard procedure in the SQL environment is to scale up. This certainly increases some of the computing power and provides more memory or storage, but often these resources end up being utilized inefficiently or incompletely by slave processes. However, in scale-out environments typically supported by NoSQL DBs, additional resources are added as new nodes that are assigned to process a certain portion of data.

In-Chip Technology

With in-chip technology, data is loaded into as many as three distinct buffers for ready access without having to read it from the actual storage device. These are: (1) buffers resident on the main processor cache which is resident on the CPU, (2) buffers resident on the disc controller, and (3) buffers resident on the disk subsystem. When properly utilized, these technologies allow a fair amount of data to be retained in these structures. For example, if the CPU cache stores 10,000 records, the disk cache stores 8,000 records, and the controller cache stores 6,000 records, memory holds a total of 24,000 records. However, a weak in-chip implementation might only have 12,000 records split in a similar fashion among the equivalent in-chip buffers.

Keep in mind that in-memory is 10 times faster, so when processing transactions on the records held in memory, the system can essentially run at least that much faster. Anyone configuring their own DW should take this into consideration.

Commodity Hardware versus Vendor-Specific Hardware

Commodity hardware is generally computer hardware that is affordable, easy to obtain, and relatively low-performance—such as the PC—and is capable of running general software such as Microsoft Windows or Linux without requiring any special devices or equipment.

Some vendors configuring systems along these lines have incorporated many off-the-shelf ("commodity" only in this respect) but highly advanced pieces of an overall system. These systems are then assembled with all the physical pieces into a system that is properly configured and integrated, tested, and benchmarked against the specific DB system to provide higher performance. However, although commodity hardware is cheaper, the amount of extra commodity hardware acquired to reach a certain performance level surpasses that of the specific or even custom hardware versions.

One of the major differences between commodity and specific hardware is that commodity systems are loosely coupled. These machines' ability to share information is so critical that the speed and performance of commodity-based systems usually drags compared to specific hardware where components could be tightly coupled—i.e., where the information is shared between CPUs much faster. This tends to boost both the speed of processing and the amount of information processed.

Super DB versus Ordinary DB

A super DB can be thought of as one capable of ultimate scalability and with speed far exceeding that of mainstream DBs. Another way of thinking of this is NoSQL (super category) versus SQL (ordinary category). Super DBs can manage a very large number of records (i.e., more than trillions of rows). Not only can these systems store such large amounts of data, they can also process them in one stretch. For this to happen, these systems must store data in the most efficient manner. Efficiency in this regard is derived from the format in which the data is stored as well as the model utilized. For example, in the Scientel LDWA system, over 1 trillion transactions can be processed for more than 1 billion customers. This capability is derived from the multi-models utilized in this single DB. In this particular application, the LDWA utilizes about 160 TB of storage as opposed to an estimated 360 TB in the SQL model—assuming it is even possible to build such a DB with SQL.

Conventional versus Modern DBs

The original reason for and the benefit of the relational DB model (e.g., SQL) is that by constraining the data schema (i.e., eliminating structural complexity of the data or decomposing it into relations), you gain power and flexibility in the types of queries you can execute against. Said another way, normalized data design yields a general-purpose query language. What we lose in data structure flexibility we gain in being able to address more data. Hence, in theory, if you require extreme query flexibility, a relational model comes in handy.

This general query advantage of relational/SQL models over NoSQL can be reduced by shifting the newly-gained financial resources (e.g., less storage with NoSQL) to customizing the query. The all-around advantages of modern approaches—e.g., NoSQL—further reduce SQL-type nominal query advantages. Additionally,

advanced NoSQL DBs such as Gensonix provide support for SQL queries.

For example, we all know that "no join is faster than join." The inherent disadvantage of decomposing your data is the required re-assembly. If you are looking for speed or scalability, then de-normalizing your data is usually the first step. The disadvantage? Now you have introduced a number of potential anomalies: Updates, inserts, and deletes can cause data inconsistencies unless you keep careful accounting of all duplication. One-to-one and one-to-many relationships are usually easy to manage, but many-to-many in denormalized schemas are nothing but a recipe for disaster—that is, if you care about consistency.

Finally, because you lose the power of a general-purpose query language (SQL), you can now rely on the domain-specific language provided by your new (NoSQL) database. Various NoSQL vendors had to introduce their own query language constructs alongside MapReduce functionality to address the problem of querying arbitrarily deep records. However, NoSQL syntax is mainly procedural with respect to the exact method for solving the problem, and this is more functional than the SQL method where the system resorts to any default methods.

In any case, SQL still has a fundamental problem with unstructured data. SQL-type systems that attempt to deal with unstructured data eventually reach a point of diminishing returns, where SQL simply becomes too expensive or fails totally.

NoSQL Data Formats

Some of the attributes commonly included under the general NoSQL label are: document based, schema-free, distributed, and scalable. However, in a document DB, to do computations, almost all that is held in character format needs to be converted back to integer style. Also, in XML versions of databases the data may be stored in character format and need to be converted back to integer style before computations can be done. In the XML and document formats, space is taken up to store tags. You must calculate the amount of space consumed by tags compared to real data. In our comparison, we found that a JSON (JavaScript Object Notation) document stored in a document DB consisting of 4 fields totaled 127 bytes. However, the data only amounted to 56 bytes and the tags took up 71 bytes. This ratio does not lend itself to an efficient big data processing scenario.

In some ways, the convenience of creating a database without a schema (NoSQL) has been exaggerated. During data entry and database creation, tags have to be created, which imposes burdens. Furthermore, tag storage requires additional space. This could turn out to be a factor in big data. Often the scalability features of NoSQL overshadow the schema, implying higher efficiency in computer system resource use. Some NoSQL DBs support schema-based data models without joins—and in some data models, even without indexes.

Thus, basically what we're doing is trading programmer convenience and adding overhead and stress to the systems in processing data. In essence, we are playing down impact to the performance (in terms of speedy usability) of big data. In other words, big data processing systems are being excessively overworked each time we inquire or process data, and we have traded this for a one-time convenience on the part of the programmer in creating the schema.

Also, when we update these databases by adding or deleting tags or fields, we cause data to shift where it is or where it was stored. This creates extra work for the systems because updating the database is now forcing a certain number of records to be rearranged within it.

Managing document, XML, and key-value-based records could require similar recalculations. By working with an expert supplier of efficient DBs, it is also possible to create logical fixed-length file formats that conform more to the nature of the hardware, which eliminates record straddling and thereby avoids multiple buffer (as opposed to single) reads.

Multi-Modeling Architecture

Important efficiency comes from the multi-modeling capabilities of certain databases. Keep in mind that all data is not created equal. It comes in various shapes and

forms. One of the advantages of NoSQL DBs is their support of multiple data modeling schemes. However, it is hard to find them in a single DB, so many companies choose multiple DBs. For example, in SQL DBs, the same relational join model is utilized for transactions as well as blogs that store unstructured data contents. When processing tables for ordinary transactions, this inefficiency adds to the storage requirements and also impacts performance. Some NoSQL DBs support multiple data modeling schemes in a single database.

For statistical purposes a column model utilizing integer arrays can store much more data in the same amount of space than can a relational table of the same data. In this particular case, we have noticed up to an 80 percent reduction in required data space. Furthermore, when statistical calculations are made on this database, the column model already has data available in an integer format; in the other models, the characters have to be transformed to integer format. Thus, combining a transaction model with the column model and the content model can save a considerable amount of storage, which translates to a considerable increase in efficiency and performance. On the other hand, storage of certain data types in certain data models decreases efficiency and exacerbates instability and reliability issues.

In-Memory Technology

In certain data warehouses, in-memory technologies are utilized for speed. However, the potential improvements are often overridden by inefficiencies inherent in standard databases. Some vendors consider solid state drives (SSDs) as in-memory technologies, but they are not. They replace hard disk drives (HDDs) that happen to utilize semiconductor components instead of spindles and motors. It is expected that in most ordinary systems, SSDs will soon replace HDDs. When this happens, system performance will automatically increase, making in-memory systems less attractive and hard to justify. Some SSDs now fail at high rates and are only catching up to the pruned reliability of HDDs.

Direct Processing versus ETL

Today, systems are being deployed that render good ETL functions but fail to do the processing required. In these scenarios, ETL methodology is utilized to extract data from the main repository and move it to the processing system, and this methodology is repeated again and again. This process can be regarded as a five-step process. If we add up all the time expended within it, we find that it is ultimately slow. A powerful data warehouse system should be able to process a large number of data sets in real time without having to fetch and deliver to another system for processing. Such are the NoSQL systems that support massively parallel processing. A NoSQL DB with large storage capacity as well as processing capability can handle large data sets in real time transactions in a two-step process.

Summary

When preparing data warehouses to store massive amounts of information, we should not forget that the purpose of data storage is for some sort of analytics to be performed at some time. If the data warehouse does well in terms of data storage and is not good at processing, we may not have progressed. A good data warehouse system should also deliver good business intelligence as well as perform standard and predictive analytics.

Building a data warehouse economically should not mean that we keep the cost at the bare minimum. Essentially the old rule applies—we get what we pay for. In building the optimum data warehouses, the best rule to apply is the price/performance ratio. From a corporate standpoint, a system that can process 1,000 transactions a second for $1 is more expensive than a system that can process 1,500 transactions a second for $2 because the increased speed in the second, nominally more expensive case allows processing of more big data, which ultimately results in more BI results faster, and more (and presumably better) strategic decisions made faster, improving the bottom line faster. After all, we are truly in a global digital world, and those who muster the advanced IT capabilities of today's enterprises are winning the race.

Information is today's currency and organizations that are highly automated raise the value of their enterprise, thanks to their IT muscle power derived from smart business intelligence—which ultimately comes from the smartest data warehouses. ■