Building a Best-Fit Data Warehouse: Why Understanding Physical Database Structures Matters

John O'Brien

John O'Brien is the chief technology officer for Dataupia.
jobrien@dataupia.com

Abstract

Do you ever blame your database for not delivering the BI capabilities you expect or for failing to handle increased volumes gracefully? Perhaps it is not inherently the database's fault. Could it be an infrastructure problem? There may be a better way to match the database and your BI workloads.

Database physical designs vary widely. Designs are always implemented within a set of requirements or constraints such as performance, budgets, or enterprise hardware standards. Over time, physical designs may be optimized for specific applications, but they can stretch only so far. One physical design might perform well in an OLTP environment but fall short of the full potential of a business intelligence implementation. A data warehouse will operate most efficiently when the BI workload and logical data models are matched with the optimal physical database architecture.

For the BI practitioner, this article will demystify physical database design for BI workloads such as operational BI, management BI, and analytics. For the DBA, it will describe techniques to optimize a database for their own workloads and deliver a best-fit data warehouse.

Introduction

Data warehouse experts are keenly aware of the role they play in bridging the abstract (business requirements for information) and the concrete (the technological resources at hand). Data warehousing projects often reflect this environment, as business analysts and data architects meet with business users to understand objectives. The same analysts and architects convene to translate the results of these conversations into a logical model. Eventually they develop technical specifications that their colleagues in the data center can parse.

This handoff between business database specialists can work smoothly, but more often than not, something is lost in translation. Designing the right technical specifications for a successful project depends on understanding how (and under what circumstances) the physical database environment bolsters or weakens the data warehouse's performance.

For data warehouse architects, there are four main models to consider: symmetric multi-processing (SMP), cluster, massively parallel processing (MPP), and grid. These different approaches all address the same goal: processing large amounts of data. The challenge is to provide enough disk space, processing power, and network bandwidth.

- SMP systems have multiple CPUs to provide scalable processing capability and have external storage servers, typically connected over the network.
- Clusters are multiple servers attached to storage, where all components operate as a single virtual server.
- MPP systems have multiple CPUs directly attached to storage that operate as coordinated yet independent components.
- Grids are collections of heterogeneous computers whose resources are invoked to work in parallel on a complex problem or massively distributed data set.

The following section examines three types of BI-related work and how they pose different challenges to the CPU power, disk space, and connectivity balance. Understanding these challenges is the key to selecting the best physical architecture model for your particular data warehousing project.

Business Requirements Expressed as BI Workload

Databases have their own architectural challenges, especially when the dimensions of time and data volume are considered. The data warehouse must serve more users

ACCESS ATTRIBUTE	PHYSICAL ARCHITECTURE	BI WORKLOAD
DATA ATTRIBUTES		
<500 GB data sets	SMP	Operational, management
Large data sets >500 GB	MPP	Management, historical
Distributed data	Grid	Operational
USER ATTRIBUTES		
<25 users	SMP	Management, historical
25–100 users	Cluster	Management
>100 users	MPP	Operational
Unlimited users	Grid	Operational
TIMING ATTRIBUTES		
Intraday updates	MPP	Operational
Ad hoc	MPP	Historical
Scheduled processes	SMP, cluster	Management, historical
24x7 active inquiries	MPP, grid	Management, operational

Table 1: Mapping usage characteristics of BI workloads to best-fit physical architecture for best performance

who are performing a greater variety of tasks. Meanwhile, the transaction database is under pressure to deliver operational reporting in near real time.

We will look at three types of workloads for data warehouses in terms of the physical design challenges they present: operational BI, enterprise BI, and historical BI. Most of the work involving the data warehouse (or variations such as operational data stores and data marts) falls into one of these three categories. These categories do not include the high-transaction, operational database, nor OLTP-only systems, although in practice, these systems are asked to handle some BI tasks.

When BI workloads are viewed from the perspective of the demands placed on the physical architecture, these characteristics stand out:

- Data volume
- Number of users
- Types of queries
- Frequency and timing of access
- Latency requirements

While there are many categories of BI activities, these five workload characteristics most clearly expose how a database's physical architecture affects the data warehouse.

Table 1 maps out types of work that data warehouses may perform and the architecture models that best match their requirements.

The "Physical Architecture Models" section later in this article provides details on how the architecture models process data and how this affects data warehouse capabilities.

Operational BI

TDWI defines operational BI this way: "Operational BI delivers information and insights to a broad range of users within hours or minutes for the purpose of managing or optimizing operational or time-sensitive business processes."

Although operational BI is an evolving term, some of the ways in which it will impact the data warehouse are clear. Data warehouses will be called upon to serve more users, keep data more current, handle different types of inquiries, and tie into operational data applications.

Data warehouses will be called upon to serve more users, keep data more current, handle different types of inquiries, and tie into operational data applications.

One characteristic of operational BI—driving information out to the edges of an organization into areas previously considered tactical—will impact the physical architecture. Users once satisfied by static Monday-morning reports are being challenged to make information-based decisions in real time.

The operational BI factors most likely to challenge data warehouse infrastructure include:

- **Data volume:** Large with rapid growth
- **Number of users:** Hundreds to thousands
- **Types of queries:** Simple, few joins, small set of fields
- **Frequency and timing of access:** Continuous with somewhat predictable spikes (that will follow the pattern of spikes in demand on operational systems)
- **Latency requirements:** Near-real-time for currency of data and report delivery

Operational BI will multiply the numbers of users (or Web services) requesting information on a daily basis and place "real-time" or "on-demand" pressures on data warehouses. Operational BI in its most basic form might be fulfilled by reporting solutions that ERP, CRM, and other operational suites already have. There is one absolute difference: data volume. The customer service representative is looking up five years' worth of history, not five months'.

A physical architecture built for OLTP simply cannot perform when handling large amounts of data (more than 500 GB).

When grid computing is ready for mass adoption, it will change many of the assumptions underlying BI today.

Shared Resources

Not many organizations have built a platform for operational BI, and few will have the opportunity to do so, since the order of the day will be to retrofit or repurpose existing infrastructure. SMP will stretch only so far in terms of concurrent requests and refresh times. The SMP architecture that had performed so well will reach scalability and performance limits very quickly when exposed to the factors just described.

Clusters might stretch further, especially if data and associated workloads can be distributed in a method that matches the way people actually use the system. The drawback here is that so much configuration and database optimization will be required that even minor changes in data or query types can become resource-intensive projects.

The optimal architecture model for this level of concurrency is one that features shared-nothing parallel processing. MPP simultaneously addresses capacity and computing needs, which removes the most significant barriers to scalability. The work of many can be processed without needing to prioritize queries or throttle performance on some to speed others.

Theoretically, if MPP were well-suited, grid would be better. However, grid computing has been implemented successfully only by a handful of the most innovative, technology-rich organizations. When grid computing is ready for mass adoption, it will change many of the assumptions underlying BI today.

Management BI

Management BI encompasses the information services required by strategic decision makers, such as dashboards and dynamic reports with drill-down and exploratory capabilities. These interfaces with their push-button views of information are creating expectations about the freshness of data. In addition, the flexibility to browse through different dimensions and drill into details could result in more ad hoc querying into various data sets.

Although management BI is the most mature type of BI workload, it is changing somewhat because the enterprise user is working with information in more varied and spontaneous ways, which poses its own challenges (especially in flexibility) to the data warehousing infrastructure. Physical architecture must support these requirements:

- **Data volume:** Slower, more controlled growth; one or two years of history
- **Number of users:** Lower hundreds
- **Types of queries:** Simple, few joins, ad hoc fetches of detail
- **Frequency and timing of access:** Varies widely, but can usually be scheduled
- **Latency requirements:** Monthly to daily

This is the most common usage scenario, and the supporting physical architecture models dominate the data warehousing infrastructure landscape.

Balanced Resources

The physical architecture that was designed for this usage context is SMP. For many organizations, it is still the best-suited architecture. For management BI, the more important problem is not which model to use but how to select the components within the architecture.

Although IT has the most experience in designing architectures to support management BI, it's still not an easy task and requires careful planning, especially around capacity planning. Forecasting growth and choosing an architecture that can accommodate just-in-time demands are important to avoid either overbuying capacity or being caught short.

Historical Analytics

Historical analytics refers to the practice of analyzing data that has been collected over a period of time. In some industries, this period may span a client's lifetime or a century. Historical analytics is characterized by large volumes of atomic, granular data that is accessed by a limited set of highly skilled users who execute complex queries. Typically, the work is scheduled and prioritized, and business cases are required to justify unplanned inquiries.

Historical BI includes data mining, predictive analytics, and pattern detection, which presents several challenges to the physical architecture:

- **Data volume:** Very large and constantly increasing
- **Number of users:** 25 or fewer
- **Types of queries:** Very complex, computing intense
- **Frequency and timing of access:** Infrequent, usually planned
- **Latency requirements:** Low

Historical analytics are so resource-intensive that even the most powerful platforms usually support low concurrency and user counts. Data loads and backups must usually be scheduled for times when queries are not running to avoid overtaxing the data warehouse.

For management BI, the more important problem is not which model to use but how to select the components within the architecture.

More Power Needed

The physical architecture best suited to this level of analytics can be summed up in one word: more. The SMP and cluster models, which start with the principle of one system and scale out from there, have too many insurmountable constraints. A computer with enough storage attached still faces a shortage of computing power and network bandwidth.

Combining computing resources in a clustered approach theoretically brings enough computing power and disk capacity together, but the complexity (and additional computing resources) needed to orchestrate this work allocation makes it unfeasible. Only MPP and grid models have succeeded in supporting historical analytics to the extent that at least large enterprises can implement them.

Physical Architecture Models

The physical architecture of a database comprises the physical data model that is adapted for a particular database and its server and storage hardware, network components, and interconnectivity. The physical architecture facilitates optimal database operations, so it must be selected carefully and designed properly to avoid database performance problems. If the physical architecture is poorly designed and doesn't match the database's expected workload, it will hamper performance.

SMP Diagram

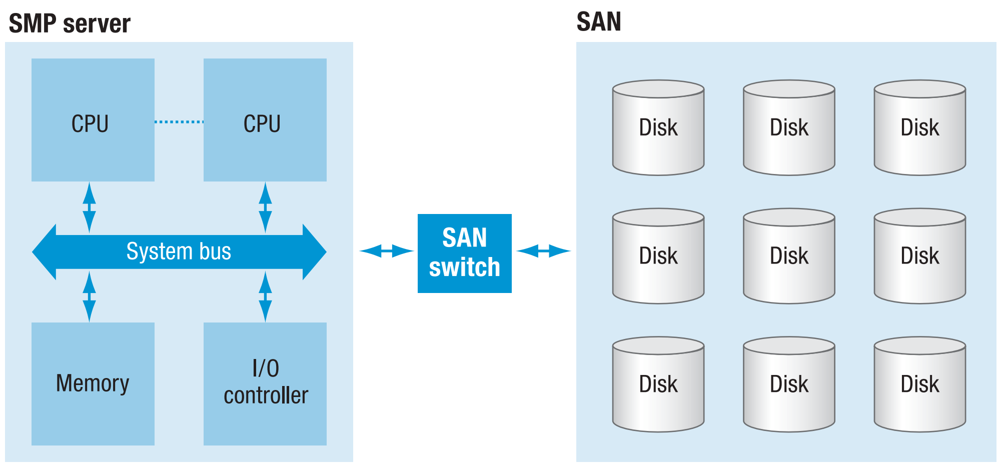

Figure 1: Symmetric multi-processing model connects CPUs in a single-server, shared-everything environment to storage

One need only review the history of RDBMSes to understand how a solid architecture might still be at the root of performance issues. The RDBMS was developed in the context of client/server computing. The client/server environment placed heavy emphasis on appropriately dividing the processing between the client and the server; indeed, it took several years to move from thin-client to fat-client and back to thin-client architectures. An enterprise client/server environment might include hundreds of clients, but commonly only one server. The approach to solving performance bottlenecks was to fine-tune the workload split (by vendors) and to acquire bigger and better servers (by users). One of the limitations inherent in this system is that the communications between server and clients at some point had a dedicated channel and had to flow in and out of a few I/O channels.

These transaction-oriented databases were designed to accommodate more users by leveraging more CPUs and shared-memory SMP servers that move a large number of small, 8K data blocks from storage.

That was 20 years ago, and since then, several physical topographies and computing models have been developed. DBMSes have kept pace to take advantage of them, mainly by including new internal mechanisms that allow a database to be adapted to or optimized for one of those environments.

On one level, these new physical blueprints can be divided into the "shared-everything" and the "shared-nothing" models. In a shared-everything model, the components operate as though they comprise a single machine. Because they contend for the same resources, work is not truly done in parallel. In the shared-nothing model, the components work independently even if they are part of a single machine and can process in parallel. Both offer ways to fully leverage all the computing resources available within an architecture, but with very different implications. In the shared-everything camp are SMP and clusters. Shared-nothing models are MPP and grid.

The following sections will review each model, point out advantages and disadvantages, and most important, describe which workloads each best supports.

SMP

Symmetric multi-processing (SMP) systems are comprised of two or more CPUs working in a shared-everything environment (see Figure 1). They primarily

Cluster Diagram

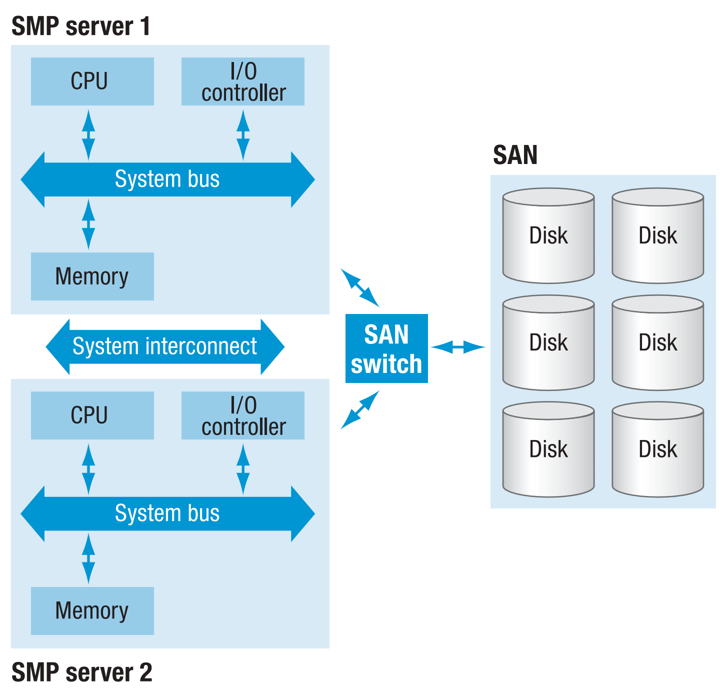

Figure 2: A cluster connects servers to a bank of disk storage

address the additional computing cycles that are required to process an increasing number of transactions.

To build out an SMP system to support a data warehouse, one typically adds storage (either NAS or SAN) to accommodate data volumes. The system architect must balance processing capacity and storage resources to ensure acceptable performance. In an environment of fast or unpredictable data growth, this is extremely challenging and involves constant tuning. Unfortunately, hardware tuning is usually not enough to keep a data warehouse operating smoothly; the database usually requires careful design and some kind of optimization.

While SMP scales on the computing side of the data warehouse infrastructure and SAN/NAS scales on the storage side, the system as a whole has significant scalability limits. Because the processing capabilities are separated from the actual data by a network, data is always being shuttled around the network. Within an SMP architecture, all data blocks must be sent to the database shared memory for processing. The network can be enhanced to provide more bandwidth and speed, but only at great expense and with uneven performance. More important, even though a network can be built out to support data warehouse traffic, other applications that rely on the network are affected to some extent when terabytes of data are moving through it.

Clusters

Clusters are groups of SMP servers that are unified into a single computing environment. Clusters are a shared-everything environment that supports close coordination of processing, but just as in SMP, they do compete for resources and can step on each other.

At the risk of oversimplifying the dynamics of innovation, the shortcomings of SMP in certain applications led to clusters. Clusters do provide the additional resources to keep pace with heavier processing loads. In an SMP environment, one can scale storage easily, and network throughput can be increased to a certain extent. Clustering allowed scaling of the database server by making available more CPUs, memory, and I/O cycles.

Allocating more resources to a task does not necessarily speed its completion. As Figure 2 shows, the clustered servers require an interconnect layer. The layer's sophistication determines the degree to which the clustered servers can interoperate or effectively share workloads. The system interconnect can be as basic as round-robin load balancing, where tasks are assigned to the next available resource in queue order. More common in a data center environment is an interconnect that allows rules-based task prioritization or criteria-based selection of "best-fit" resources.

These server-level interconnects could only deliver some intelligence to resource allocation, so the major

MPP Diagram

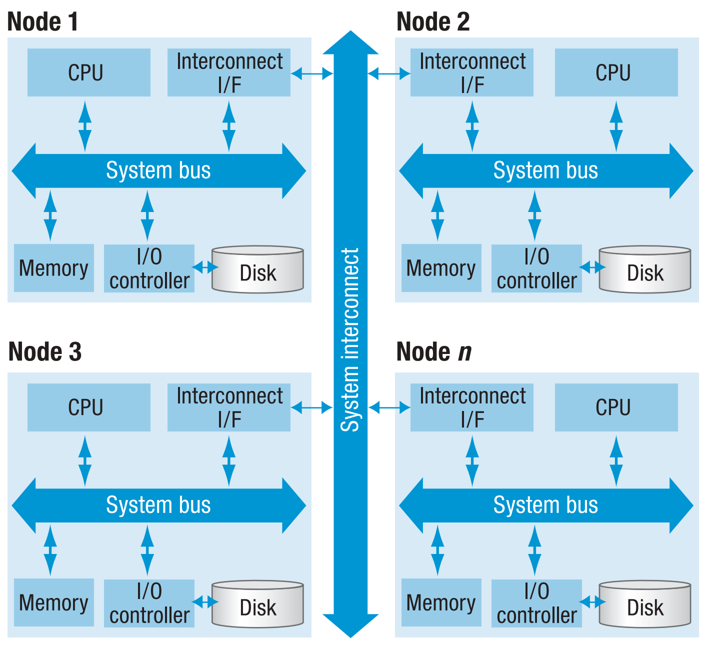

Figure 3: Massively parallel processing collocates CPUs with disk

database vendors built interconnect mechanisms that were designed to distribute data processing tasks. The interconnect capabilities of distributed databases are sophisticated extensions of the database management system itself. They preserve its atomic nature (even of distributed transactions) and auditing. They ensure recovery and security at the database level across the cluster.

Clusters have solved the challenges of accessing dispersed data and increasing DBMS throughput, especially for most enterprise transactional systems. However, because a cluster is a networked solution, bandwidth can still be a stumbling block whenever large data sets are sent over the network. In fact, such large file transfers from storage servers to a single machine in the cluster can bog down the performance of all the servers in the cluster—not by taking up CPU resources, but by consuming bandwidth. The other servers cannot return results until bandwidth is once again available.

MPP

SMP and cluster architectures use a shared-everything model, which shields applications from some of the inherent complexity. At the same time, shared-everything can result in significant resource contention that can't be resolved by installing more hardware. The nature of data warehousing involves working with massive amounts of data, which can strain any resource pool. Massively parallel processing (MPP) architecture was developed to address this need to combine resources while simultaneously dividing the work as efficiently as possible. Data warehouse appliance vendors have all gravitated to this architecture as the most efficient model for supporting large databases.

Many of the hardware components found in a cluster are also found in an MPP architecture—balanced CPU and disk capacity as well as a system interconnect mechanism. The significant difference is that the work is divided across these resources in a very different manner. The work is always delegated to the CPU associated with a particular data range. In a data warehouse appliance, where CPUs are physically located next to the disk, the association is straightforward.

Once the work (a query or a subset of a query) is allocated, the CPU(s), memory, and I/O resources work exclusively on its results. Only the results are then sent through the interconnect and combined with other results before being sent over the network to the client. The coordination of the independently functioning components occurs via the system interconnect and software modules sitting on each of the subsystems to provide even more sophisticated orchestration of work than clusters offer.

MPP ensures equilibrium between disk capacity and processing power, just as clusters do. However, MPP also eliminates the last bottleneck—network bandwidth—

Grid Diagram

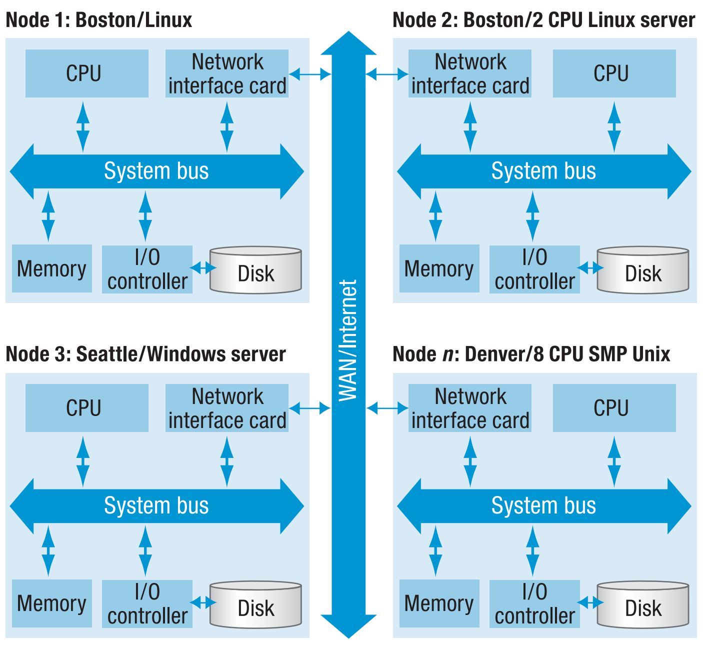
Node 1: Boston/Linux
Node 2: Boston/2 CPU Linux server
Node 3: Seattle/Windows server
Node *n*: Denver/8 CPU SMP Unix
CPU
Network interface card
System bus
Memory
I/O controller
Disk
WAN/Internet

Figure 4: Grid assembles available computing and disk resources into a single computing system

thanks to two attributes. First, because the MPP architecture creates a virtual subsystem, none of its internal communications consumes bandwidth on the remainder of the LAN or WAN. Second, MPP does not involve shifting massive volumes of data in order to process it. It does not fetch multiple terabytes of data from a storage farm, because it enables processing data where it is physically located. After processing, MPP sends only results over the network. Compared to the SMP or cluster architectures, its use of networking resources is minimal—unless, of course, your queries are returning multi-terabytes of data.

We've described how MPP facilitates dividing queries for efficient processing, but it can also allocate work among components according to the type of work. It's analogous to running background processes. In an MPP environment, you can dedicate some components to load or ETL operations while the remainder is available to process queries. This aspect of MPP is gaining importance as 24x7 operations demand that systems keep serving users while maintaining currency and data quality.

Grid

There are many ways to define a grid. The ambiguity and confusion around what a grid is and how it's supposed to function are probably signs of how early grids are in their development.

A generally accepted definition of a grid is a large, heterogeneous collection of geographically dispersed computers whose resources are shared over a network to create a supercomputer. If one computer in the grid fails, the grid software simply reassigns its job to another computer in the grid. Failed computers in a grid are repaired or replaced whenever it is convenient, as no single computer has a significant impact on the grid.

Computation-intensive applications that are designed for parallel processing can draw on otherwise unused processing capacity. Using this definition, examples of grids are Google, IBM Blue Gene, and Livermore Labs' infrastructure. The power of grid computing is evident—Blue Gene topped the list of the world's fastest computers for a third year in January 2008 by performing 478.2 trillion calculations per second.

Figure 4 takes components that support other applications and shows how they would translate into a data warehouse infrastructure grid.

In Figure 4, the system interconnect is the outstanding difference from the previous model diagrams. In a grid, the system interconnect is a general-purpose WAN/Internet connection to which the participants have no specialized connection mechanism. In other words, the connections that combine the member components into a single system are not physical. They are virtual—that is, invoked through software. Another difference in the diagram is that the members are constructed on different architectures, use incompatible hardware, and run different operating systems.

Other differences are not apparent and have to do with how the services of the members are used. In a grid system, resources are located that have unused capacity, usually processing capacity. The unused capacity is then allocated to processing tasks that require multiple CPUs. Meanwhile, the member continues to operate to the same level on its "own" tasks. The members are not specialized. The orchestration and aggregation of results occurs outside of the member systems.

Data warehouses built on a grid infrastructure are still the subject of theory or experiments at best. While the Google grid is an example of accessible data located in a grid environment, the industry won't be seeing pooled disk space supporting a virtualized data warehouse in the near term.

There will always be a gap between the capabilities of the physical architecture and the requirements of the data warehouse.

Adjusting DB Internals to Manage BI Workloads

Even if there were a perfect physical architecture for data warehousing, chances are it would still not be perfectly suited to how your organization uses its data warehouse. There will always be a gap between the capabilities of the physical architecture and the requirements of the data warehouse. Most often the gap shows up in scalability issues, especially in the ability to accommodate data growth or changes in workload requirements.

A data warehouse on SMP might be able to house 40 TBs of data, but accessing or manipulating data might be very challenging. For example, an SMP data warehouse with 40 TBs might still operate well if the scope of its work is focused on a subset of data (less than 1 TB) or one type of workload. Clustering will improve performance over an SMP architecture working with larger data sets, but workload is still limited. The MPP architecture will stretch the most, but it, too, must be adjusted to maintain performance as data grows and as workloads vary.

To bridge that gap and keep pace with change, techniques can be executed at the database internals level that will extend the architecture's capacity. Because each architecture faces different stresses from massive data volumes, it responds best to different optimization strategies. Some of these strategies involve hardware-based adjustments, while others are schema- or index-based. Typically we find these architecture and optimization pairings:

- Partitioning with SMP (and clusters)
- Data distribution with MPP

Partitioning

Partitioning was developed in response to data sets that were too large for DBMSes to manage. One technique for handling large amounts of data is to split it up into a collection of manageable parts. Partitioning first allows you to isolate data to a subset to avoid scanning large amounts of irrelevant data. Then the database can run its operations in parallel within that partition. A good partition design coupled with database parallelism can outperform standard indexing.

Common practice is to partition so as to enable aging data across storage tiers. For example, partitioning a year of data into 12 partitions simplifies the task of keeping one year of data in the data warehouse; at the end of every month, roll off one month and migrate it to other tiers for a cost-effective storage management solution.

Another benefit of partitioning is the ability to manage data subsets by time. The most common partition key specified in a data warehouse is a date range. A database partition, and its locally managed index, can be loaded into the data warehouse without interfering with other partitions. The result is parallel operations, with the associated performance and flexibility benefits.

Effective partitioning requires a deep understanding of how applications access data, how this access relates to the data models, and how the models are defined within the schema. Most important is to understand how isolating a subset of data can improve a query's performance. If you just go by the size of certain tables or other design elements, your partitions might actually slow performance. Similarly, if you don't understand when certain data must be accessed, you could create concurrency problems where none existed before.

Whether you isolate by day, week, or month depends on how you want to access the data. You must balance the manageability of a partition with the frequency or context of its access. For example, partitioning a year into 365 subsets has its benefits as it maps to operational data. You get optimal performance when you analyze a day's worth of data. However, if you need to work with 365 partitions to pull together different information, you will incur considerable overhead that will slow performance.

Ironically, partitioning to align with access to data and partitioning for aging data are diametrically opposed. An organization must choose one approach or the other for a given data set. Once the choice has been made, a master partitioner can use partitions and sub-partitions to manipulate workloads and orchestrate resources to bolster the performance of the data warehouse's most critical operations.

Once data is isolated to enhance query performance, other benefits follow. Techniques such as immediately marking a partition of static data as read-only can improve performance. On the other hand, partitioning requires a high level of expertise and administration. Sizing, creating, and maintaining partitions are ongoing tasks that can become minor projects if there are changes to types of data sets, queries, or workloads.

Data Distribution

Data distribution is at the foundation of MPP. SMP strives to isolate data with partitioning to gain performance; the MPP environment strives to spread the data evenly across all processing nodes to optimize parallel database operations, including data loads.

With MPP, data distribution is handled as part of the process of physically writing data onto the disks and managed by software that is distinct from data management or the operating system. This last aspect is another factor in MPP's performance advantage over SMP—the DBMS and the OS are spared from some of the management overhead incurred in the SMP architecture. In addition, more flexibility is gained to accommodate changing workloads, as adjustments can be made at the point where data is loaded.

Ironically, partitioning to align with access to data and partitioning for aging data are diametrically opposed.

If you were to engineer your own MPP system, creating the data distribution algorithms and programming the write, update, and redistribution procedures would be a monumental task. Vendors that deliver MPP systems have embedded this functionality into their appliances and provided utilities for configuring data distribution.

Once again, the challenge for the DBA is to identify the best distribution key to spread the data evenly for loads and queries. MPP database vendors typically offer several options to distribute: round robin, hash-based, and specified key. You must understand the data coming into the MPP system and the queries that will be executed to select the best approach.

To ensure that an MPP architecture supports the type of work your data warehouse performs, you must consider whether you are served better by distributing data evenly across disks or distributing data according to access patterns or table sizes, especially as the data warehouse transitions from one type of primary workload to another. Trade-offs are part of the decision-making process. Will you select a distribution key that offers the best performance for a percentage of your BI workload, or will you select a distribution method (hash or round robin) that offers the highest overall average for a mixed

BI workload? Will you decide to physically design the database to do both and just purchase the additional storage space?

Conclusion

The many physical architectures available to today's data warehouse evolved to address problems arising from limited CPU or disk resources. Their emphasis on different aspects of the architecture can have a significant impact on how a data warehouse performs. Under a set of conditions for which its architecture was designed, the data warehouse performs well. When conditions change, the data warehouse may not perform at the same level.

How does your data warehouse project map to the three types of BI workloads? If it maps neatly today, what does the future look like? Is there a movement toward having more people query the data warehouse directly? Is data growing, especially data that has to remain easily retrievable?

The key to building the best infrastructure for your data warehouse is to match your organization's access needs with the most suitable physical architecture. It sounds simple. The challenge is that your data warehouse probably needs to execute most, if not all, BI workloads. If it doesn't currently operate under these conditions, you can rest assured that at some point it will have to keep pace with business dynamics, more sophisticated users, and technology innovation.

My advice is to use the simple architecture/grid matrix (Table 1) as a guide—not for projecting the status quo five years forward, but for anticipating future business requirements. In other words, don't design for the probable—design for the possible. ■