

DEPLOYING EMC VNX UNIFIED STORAGE SYSTEMS FOR DATA WAREHOUSE APPLICATIONS

Best Practices for Adoption and Deployment

EMC Solutions Group

Abstract

This white paper discusses how the newly introduced EMC® VNX™ unified storage systems address the challenges of the rapidly growing data warehouse deployments in many businesses by offering significantly increased performance and capacity, at reduced footprint and cost. Best practice guidelines based on engineering testing of typical use cases are provided to assist IT staff in adopting and deploying these systems quickly and to their best advantage.

December 2011

Copyright © 2011 EMC Corporation. All Rights Reserved.

EMC believes the information in this publication is accurate as of its publication date. The information is subject to change without notice.

The information in this publication is provided “as is.” EMC Corporation makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any EMC software described in this publication requires an applicable software license.

For the most up-to-date listing of EMC product names, see EMC Corporation Trademarks on EMC.com.

All trademarks used herein are the property of their respective owners.

Part Number H8177.3

Table of contents

Executive summary	5
Overview.....	5
Key VNX family product features	5
EMC VNX family: unified storage platforms.....	7
Overview.....	7
Data Warehouse deployment trends and VNX systems.....	7
Data density.....	8
DW deployments and VNX storage systems.....	9
Overview.....	9
The “massively parallel” data warehouse engine trend	10
Overview.....	10
Using MPP for queries involving large data sets	10
Example of using MPP for a business query involving large volume of data	10
Processing with proper I/O subsystem data feed	11
Guidelines for sizing the right VNX storage system	12
Overview.....	12
Redundant data	12
Applying Flash drives to improve access for derived data.....	12
Row versus column data storing format and storage system I/O bandwidth implications.....	13
Applying bandwidth sizing guidelines to VNX storage systems	16
Overview.....	16
Hardware components.....	16
VNX bandwidth sizing approach.....	16
Read bandwidth of different models	16
VNX5100	16
VNX5300	17
VNX5500	17
Release 31.5 VNX5500 enhancement option	17
VNX5500 FC read bandwidth enhanced option	18
Read bandwidth for VNX5700 and VNX7500.....	18
Choosing DAEs and drive types.....	19
Front-side connectivity choices for DW deployments.....	22
Storage pools and auto-tiering.....	25
Absolute bandwidth consideration between virtual pool LUNs and traditional fixed LUNs	27
Virtual pool LUNs.....	28

VNX system cache setting for DW deployments	30
Storage processor cache memory consideration.....	30
FAST Cache leverages Flash drives	32
Choosing between deploying for Flash drives for FAST caching versus FAST auto-tiering use.....	33
EMC engineering case studies deploying VNX5300 with an Oracle DW test workload	35
Overview.....	35
Case Study 1: Dell rack servers with unified storage.....	36
Case Study 2: Deploying with UCS blade servers.....	37
Leveraging the R31.5 FC enhanced read bandwidth feature for VNX5500	39
Conclusion	42
Overview.....	42
Key findings from the use cases.....	42
References.....	43
EMC documents.....	43

Executive summary

Overview

Data warehouses are able to offer rich insight into daily business decisions and can help users radically change and improve their business model. However, the complexity of these data warehouses is increasing, with very diversified data details and rapid growth.

The trend of deep business analytics has changed. There is no longer a need to carefully plan and map out the data for efficient data querying. Because of an increase in processing power from processors, such as the multi-core processors, which can pour through data “by brute force” in parallel, warehouses are now built around data management systems that are Massively Parallel Processing (MPP)-based. Business data is frequently partitioned, and complex queries are organized into parallel subtasks that can optimally engage as much concurrent processing resources as possible.

In order to support the parallel processing model effectively, scalability and performance are critical characteristics of a storage platform. Additionally, because the data is the prized asset, it must be protected against corruption to enable queries against the data to continue, thus enabling the continuation of sound business decisions based on data. The introduction of the EMC® VNX™ family of unified storage platforms continues the tradition of providing one of the highest industry standards in data reliability and availability. The storage can also process high IOPS and meet bandwidth requirements to support the sustained data access bandwidth rates demanded by the new MPP-based data analytic approaches. The new system design has also placed heavy emphasis on storage efficiencies and density, as well as crucial green storage factors, such as data center footprint, lower power consumption, and improvement in power reporting.

This white paper covers key features in the VNX family specifically relevant to deploying next-generation data warehouses, supported by EMC Engineering case study findings.

Key VNX family product features

The key hardware features of the VNX family are as follows:

- Storage processors based on Intel Westmere multi-core processors
- Multi-channel 6 Gb SAS backend I/O buses
- PCI/E GEN II connection adapters for both front and backend data path connections
- Support for multiple types of drives including:
 - Flash drives
 - SAS 10k/15k drives of various capacity, 2.5”/3.5” form factors and capacity
 - Near Line SAS drives with maximum capacity of up to 3TB per drive

The key software features of the VNX family are as follows:

- The Fully Automated Storage Tiering (FAST) suite enabling the support for automatic data storage tiering, and I/O performance boosting through extended storage system caching with FLASH drives in the system

- Replication suite including SnapView, MirrorView, RecoverPoint splitter, and SANCopy
- Array management suite—Unisphere, a single management window for the provisioning, management, and control of both block and file data storage

This white paper covers the key features in the VNX family specifically relevant to deploying the next-generation data warehouses, supported by EMC Engineering case study findings.

EMC VNX family: unified storage platforms

Overview

The new generation of EMC unified storage offers a range of choices for meeting the diversified business requirements of the enterprise, including performance, capacity, and protection, at the best total cost of ownership.

Figure 1 shows the product line of EMC VNX family.



Figure 1. The VNX family of unified storage platforms

A key distinction of this product line is the support for block-based and file-based external storage access over a variety of access protocols: Fibre Channel (FC), iSCSI, Fibre Channel over Ethernet (FCoE), NFS, and CIFS network shared file access. EMC Unisphere™ is an easy-to-use, web-based application, which provides complete management capabilities. For more information, refer to the following EMC white paper or contact EMC for a demonstration:

[Introducing EMC Unisphere: A Common Midrange Element Manager](#)

Data Warehouse deployment trends and VNX systems

Data warehouse deployments are rapidly increasing the need for storage. However, more storage is not the only answer—scalability and bandwidth are also critical to ensure that market data and other valuable information can be obtained quickly.

With this generation of storage, EMC has boosted both sustained read and write bandwidth. Sustained read/write bandwidths are typically crucial for data warehouse deployments contending with ever-increasing data volumes. The boost in bandwidth results from innovative system re-architecting, including the adoption of high-power multi-core processors, fast memory chips, multichannel 6 Gb/s SAS backend buses, and the use of new GEN II PCI/E buses. Additionally, advanced storage software features such as Fully Automated Storage Tiering (FAST) deliver performance balanced with lower costs. Flash drives can be used to support extended hot data caching, and multi-threaded code enhancements can take full advantage of the high power offered by the multi-core processors.

Data density

Another consequence of the rapid data growth is the need to store data more compactly. Data centers are continually contending with physical facility space, power, and cooling challenges. The VNX family supports 2.5-inch SAS drives in a 2U disk array enclosure (DAE) that can hold up to 25 drives, which is one of the densest offerings in the industry. This offers 2.5 times the storage capacity in the same rack space compared to previous generation products. The power efficiency of the new DAEs also makes it more cost-effective to store the increased data in this much more compact footprint without increasing power consumption and cooling.

DW deployments and VNX storage systems

Overview

Many customers are planning to deploy new data warehouses. And while most people relate to the term “data warehouses” as a methodology for collecting and managing a large set of data objects with information that is relevant or valuable to the business, there can be quite a diverse approach to actually implementing the actual dataset collection. Just as building a warehouse to stock grocery store products to supply grocery stores and supermarkets would have quite a different approach compared to one used by Home Depot for stocking home building material, the business data model that governs what data is relevant to the business, and how the data needs to be stored and interpreted to turn into information useful for the business, would often dictate the approach required or desired.

The underlying storage system support is an integral part of physical implementation of a data warehouse. Yet, the right choice of storage infrastructure has to be made with some understanding of what type of data warehouses need to be built. It is therefore worthwhile to digress a bit to examine how the changes in data warehouse deployment trends have affected the way data may be stored in the supporting storage infrastructure, and the I/O that database systems are now driving down to the storage system.

The “massively parallel” data warehouse engine trend

Overview

Information generation and capturing have been growing at an exponential rate, in no small part due to the technological advances in electronic devices supporting a wide range of messaging and communications, empowered by advanced communication technologies such as wireless networking, high-speed network support, and others. Social electronic networks such as Facebook, along with rapid adoption of information exchange facilities such as email and instant messaging, add to the volume of information generation. Computer systems and smart meters capture information about key clicks, usage, and access patterns of resources on an ongoing basis (as opposed to, for example, the utility service person going onsite to a customer’s house to do a manual meter reading every month).

Because so much of the actual information details are generated automatically by machines and electronic device as opposed to by human hands, there are many more opportunities to do more with the information available. A classic example is the minute-by-minute power usage in a household monitored and recorded by the new smart meters that utility companies are starting to adopt and standardize to. With the added details, utility companies can develop a far more comprehensive understanding of the actual power usage pattern in a neighborhood, and then optimize the power generation and supply process to ensure higher quality service without having to overbudget the power reserve. Special incentive programs can be structured and offered to customers to allow power to be distributed and supplied in the most cost-effective manner to help control cost for both the utility company and customers.

Innovative companies recognize the vast potential of leveraging the available information to strengthen their business. As a result, there is a wave of new data warehouse initiatives from many companies across different disciplines springing up very quickly.

Using MPP for queries involving large data sets

To effectively process the massive (and still growing) volume of data, most DBMS engines are going to a Massively Parallel Processing (MPP) model for supporting heavy-duty, deep business data analytics, timely information reporting, and dashboard updates. Since a primary focus is on quickly pouring through a lot of business-related information pieces to provide insights into how effectively the business is operating, and to mine for new insights that can radically transform the business operation into a much more efficient or competitive execution, it is often the case that complex business queries can be broken down into parallelizable subtasks. By leveraging the relative cheap and abundant processing cycles afforded by the new-generation multisoocket, multi-core commodity servers, the subtasks can be distributed and run on many servers in parallel, properly distributed, coordinated, and managed by the DBMS query optimization engine, and the answers to such queries can still be realistically reached in minutes as opposed to hours or days.

Example of using MPP for a business query involving large volume of data

Figure 2 shows an example of using MPP for a business query involving large volume of data.

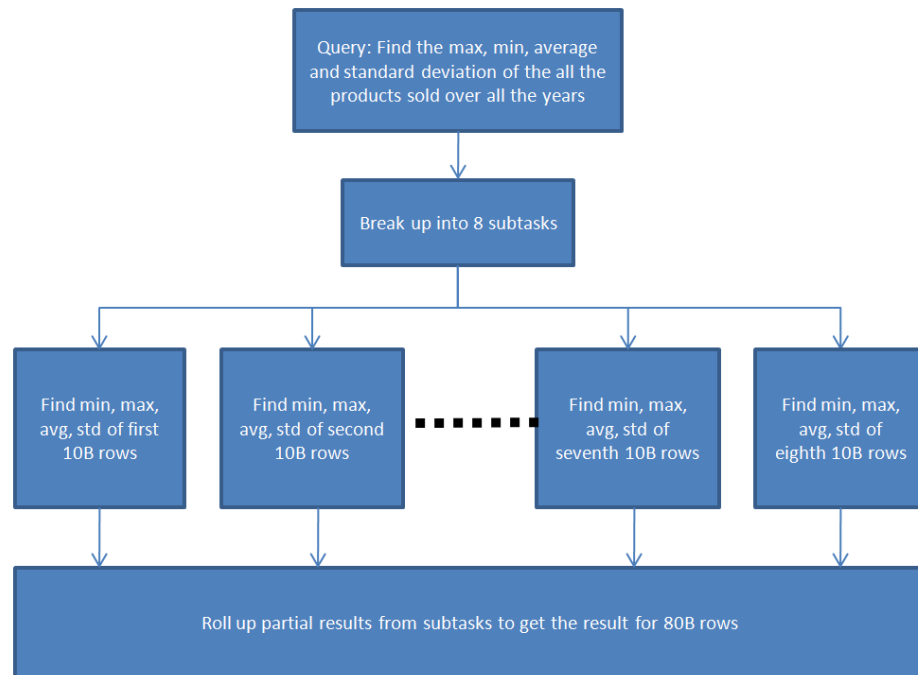


Figure 2. Using MPP for a business query involving large volume of data

All the key DBMS players in this space offer their unique technology for supporting parallel complex queries, including Oracle, IBM (DB2), Microsoft (SQL Server), Sybase (IQ), and Teradata. There are also a number of very successful new players in this space, also favoring such a model, including Netezza, [EMC Greenplum®](#), Vertica, Aster, and others.

Processing with proper I/O subsystem data feed

Regardless of how each system approaches the parallel query support model, there is a common theme. The success of such a model depends on:

- The ability to reasonably break the complex query job into a number of subtasks that can run in parallel with other subtasks without getting into each other’s way; this is typically achieved by having each subtask assigned a separate resource, to work against a different “segment/partition” of the dataset needed to be processed.
- The ability to balance the processing resource allocated to each subtask with an adequate rate of data feed from the underlying storage supporting infrastructure to allow each subtask to proceed smoothly as a data processing pipeline.

The key factor is *balance*. In the parallel model, the ideal is that all subtasks do roughly the same amount of work, run independent of each other for roughly the same amount of time, and each subtask can continue to proceed smoothly without experiencing significant delays waiting for the next batch of data to be delivered from storage.

The last factor is often the governing factor in sizing the best storage supporting infrastructure fit for a new data warehouse project.

Guidelines for sizing the right VNX storage system

Overview

It should be remembered that not all business queries run against the stored data in the data warehouse, or against data marts spun off from a central enterprise data warehouse, need to constantly do full data scans. In fact, it is frequently a practical implementation practice to create some form of redundant data, when the data is initially admitted into the warehouse, in anticipation of the “frequently asked questions” from the business standpoint, to avoid repeated data scans. Typical examples may include special summary data tables that are generated to “summarize” the aggregate effect of a number of new data rows inserted, such as keeping a year-to-date running revenue total for each store in the enterprise as each store product item is sold. The information about goods sold in a particular store throughout the day is already captured in each sales receipt. But rather than going through each sales receipt to re-compute what the store has sold during the day, a special table can be maintained to keep a running total, thereby avoiding the need to keep recalculating that store total, if this is a piece of information that is often needed by different user communities in the store or the company.

Other examples may include indices that are created to optimize for target value data searches, materialized views, and dimension cubes.

Redundant data

When the warehouse system is implemented with such extra and “redundant” data being stored, there is a cost associated with initially deriving the extra data from the raw data, or what is commonly called the “fact table” data in the DW business. There is more processing overhead to extract, duplicate, calculate, or derive these sets of data, typically during data insertion time. There is management overhead to manage the data, and extra application design and implementation tradeoffs to leverage the data. And of course, extra storage is needed to store the extra data. The main reason for paying all of this extra overhead is specifically to improve the efficiency of retrieving this information or to improve business queries that typically need the data in its derived form.

Applying Flash drives to improve access for derived data

This is where the automatic multitiering feature in the VNX is an ideal fit. By deploying a VNX storage unit with both Flash drives and the high-density, small form factor (2.5 inch) SAS drives, the bulk of the fact data (the data with the details) will be kept by the tiering algorithm on the SAS drives, while the “redundantly created” data for the primary purpose of accelerating query access will be positioned by the storage tiering algorithm to the Flash drives, ensuring the lowest access latency when the special data is needed.

More and more, it is no longer possible to completely avoid relying on “brute force,” pouring through a significant amount of the detail fact data to answer complex business queries, especially those ad hoc queries. A new generation of data analysts now specializes in exploring a new and rich information set, looking for new business insights that the rich datasets may be revealing or exposing that they have not thought of. With the electronic age and the social network through the Internet, there is a lot more data sharing, data modeling, usage sharing, and collaboration. So, most DBMS engine vendors have established general rule-of-thumb guidelines for what the supporting storage infrastructure needs to deliver for parallel data processing to work efficiently.

Row versus column data storing format and storage system I/O bandwidth implications

Most engines that store data in the more traditional relational table row form have generally recommended about 150-200 MB/s of sustained data read rate, reading with typically large I/O sizes, such as 256 KB per host read or higher, to match up with the engine code executing on a commodity CPU processor like the Intel Nehalem processors. When the row data is stored with compression, the requirement would be more about 150 MB/s, while it would be more toward 200 MB/s if the data is not stored compressed.

Many engines now also offer either a standard feature, or logical data storage option, to store some of the data in a *column* orientation.

For example, a sales receipt may have information about the customer, the number of units of items the customer bought in that transaction, the prices of the different items, the item descriptions, and other data. In a traditional relational table row storage format, the data stored may look as shown in Figure 3.

Receipt # 1	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 2	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 3	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 4	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 5	Customer name	Date of sale	Item description	Qty sold	Sales price

...

Receipt # N	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price

Figure 3. Row-based data store

Receipt # 1	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 2	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 3	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 4	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # 5	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt # N	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price
Receipt...	Customer name	Date of sale	Item description	Qty sold	Sales price

Figure 4. Column-based data store

When the DBMS engine employs a column-based data storing model, the data from the same *column* for all the different data rows is being stored packed together adjacent to each other in storage (the same color grouped columns in Figure 4).

Now suppose we exercise a business query that is interested in finding out how many total units we have sold of a particular item, and how much revenue was generated. Basically, we need the information from the last three columns to get the answer. With the data stored by columns, we would be able to pull across exactly just the three columns' worth of data from the I/O subsystem. When the data is stored in row orientation, we have to bring all rows across, and basically discard the fields from each row relating to the first three columns, which are not relevant to the query we are posting.

Hence, DBMS vendors supporting column-based store options (for example, Sybase, EMC Greenplum, Vertica, ParAccel, and now, to a degree, Oracle) frequently use a CP core to storage read data feed ratio that is lower, such as 50-100 MB/s. They are banking on the fact that most reporting, business intelligence, and business analytic applications tend to work with a subset of the fields carried in a data row (such as columns with numeric values, dates, and so on) for analytic-focused work, and that most of the other auxiliary data (for example, customer name) is not as commonly involved.

Also, when a series of like content (such as numbers) for the sales value is stored adjacent to each other on disk, the similarity in the data type frequently offers much more opportunity for compression. (For example, 100 sales receipts for \$5 sales can be readily compressed into a coded representation of “the next 100 receipts are all \$5,” as opposed to storing the number \$5 100 different times, one for each row).

While these are generalized guidelines, and where queries involving large table scans tend to be less demanding on storage data read bandwidth for columnar store formats, ultimately it still depends on understanding the actual business data usage pattern. If the bulk of the business queries that require large table data scans end up

using most of the data fields in each row, columnar data store format can in fact be *more* I/O bandwidth-intensive than row-based store. In planning for the right storage system choice, it is important for the IT team responsible for the supporting storage infrastructure to work closely with the warehouse data model designers and implementers to better understand the most likely I/O pattern that the ultimate deployment will be driving, so the correct storage system configuration choices and tradeoffs can be made.

Applying bandwidth sizing guidelines to VNX storage systems

Overview

The maximum sustainable data read and write bandwidth of each VMX system is a combination of many interoperating parts of the storage system.

Hardware components

There are at least five hardware components that factor into the bandwidth equation:

- The front-side connection ports (and protocol) used to connect servers to the storage system
- The number of physical drives holding the needed data that participate in delivering data from the drives; the speed and type of drives affect what can be practically delivered by the design of the drives
- The storage system's CPU and memory capability for accepting incoming requests from the host, mapping and dispatching the I/O to the drives in the back end, and returning the data retrieved from the drives back to the servers through the front-side connections
- The number of buses/channels over which the storage system processors can dispatch I/O to the backend drives, and to pull the data bits up from the drives
- The bandwidth supported by the DAE holding the drives

With a modular design the number of front-side connections, and the number of DAEs and drives that can be configured in each VNX module, can vary. The other two components, namely, the storage processors' CPU and memory capabilities, and the number of backend buses/channels for carrying data I/O to/from the physical drives, are more or less fixed within a particular storage system.

VNX bandwidth sizing approach

When faced with sizing the right VNX storage configuration for a particular deployment expectation, it is therefore often the right approach to start by choosing the VNX product with the storage processors and backend bus maximum theoretical capacity that will satisfy the expected business requirement. Then, choose the number of front-side modules that would provide the needed host connectivity to drive the needed bandwidth. Finally, configure enough drives to accommodate not only the capacity needed, but also to make sure that the data is spread over enough drives to deliver the performance needed.

Read bandwidth of different models

Models of the VNX family are designed with different maximum memory configurations, and use somewhat different classes of multi-core Intel CPU chips. The three lower-end systems — the VNX5100™, VNX5300™, and VNX5500™ — are designed to support primarily around over 4,200 MB/s of single direction data transfer from the backend buses. (Typically, only a single bus connection is needed, though the systems do support the use of both backend connection buses to more evenly distribute the load.)

VNX5100

The low-end VNX5100 is a system designed to support FC connection only. Even though the physical hardware supports up to eight separate FC connections of 8 Gb/s, the integrated connection modules (one in each storage processor) are rated to deliver about 1,500 MB/s sustained read rate per processor, or 3,000 MB/s for both processors, for the most typical I/O pattern pushed by the DBMS engines when doing parallel table partition scans.

VNX5300

In addition to the integrated FC front-side connections, the VNX5300 supports two expansion connection modules in each storage processor. These UltraFlex™-based I/O connection modules support additional FC connections (4 x 8 Gb modules), dual-port iSCSI (1 Gb and 10 Gb), and now, also dual-port 10 Gb FCoE connections. EMC Engineering recommends that a good practical sizing rule for DW deployment is around 3,700 MB/s.

VNX5500

VNX5500, as with VNX5300, leverages the integrated connectors to support up to two backend bus connections, as well as two 8 Gb FC connections, for each storage processor. VNX5500 also accommodates two expansion connection modules for potential additional connections. Compared to VNX5300, the CPU and memory subsystem for VNX5500 supports considerably more sustained data read bandwidth. It can deliver over 6,000 MB/s of read bandwidth for typical large chunk data read type I/O patterns in DW/DSS deployments. However, if only the integrated backend bus ports are used from the integrated module, the total backend bus bandwidth that the integrated ports can sustain is practically limited to around 2100 MB/s from each SP, or 4,200 MB/s in total for VNX5500.

Release 31.5 VNX5500 enhancement option

Flare Release 31.5 introduced a new enhancement in the storage system firmware. VNX5500 can recognize and allow a second backend FC connection module to be used in one of the two I/O expansion slots. Therefore, unlike VNX5300, where the two expansion slots are used exclusively for adding more front side connection ports, VNX5500 allows one of the two expansion slots to be used for supporting either additional front-side, or additional backend bus connections.

By inserting a backend bus I/O module into one of the two expansion slots, two more backend disk bus loops can be added. This can bring the backend sustained read bandwidth to over 6,000 MB/s, the theoretical maximum that the SP memory subsystem can support. With this new connection option, VNX5500 can support 1.6 times the practical read bandwidth that can be derived from VNX5300, within a comparable storage footprint.

Since one of the two expansion slots in each storage processor must be used to support additional backend bus connections, only one remaining slot is able for the front side connection module. Customers cannot enable enough iSCSI or FCoE front side connections to drive the array to achieve the 6,000MB/s of read bandwidth using those connection protocols.

To fully realize the effect of the 6,000+ MB/s read bandwidth advantage, the remaining I/O module must be configured for FC 8 Gb connections, if a single connection protocol is used. The FC module, together with the FC integrated connection ports from the SP, can support over 3,000 MB/s per SP (two integrated 8 Gb FC ports, and four other deployable FC connection ports from the expansion FC connection I/O module). Otherwise, a combination of 8 Gb FC ports from the integrated ports (two ports) and two other iSCSI or FCoE ports in the expansion modules would be the only way to have enough front-side connectivity to fully drive the 6,000+ MB/s of read bandwidth. In that case, multiple server-to-storage connection protocols are needed. This increases the complexity of deploying multiple I/O drivers on the server.

Therefore, this is a deployment tradeoff. If the deployment is fundamentally FC based, customers need to configure VNX5500 with the additional backend port connections

by committing one of the two available I/O module expansion slots for additional backend bus connectivity, and deploy additional FC 8 Gb connections from the remaining I/O module on each SP.

Alternatively, if the deployment uses iSCSI and/or FCoE, the I/O modules should be reserved for adding more front-side connections. Practically, around 4,200 MB/s of sustained read bandwidth from that single pair of integrated backend bus port connections should be expected.

VNX5500 FC read bandwidth enhanced option

Figure 5 shows the recommended connection model that can drive VNX5500 to the 6,000 MB/s read level using 8 Gb FC connections:

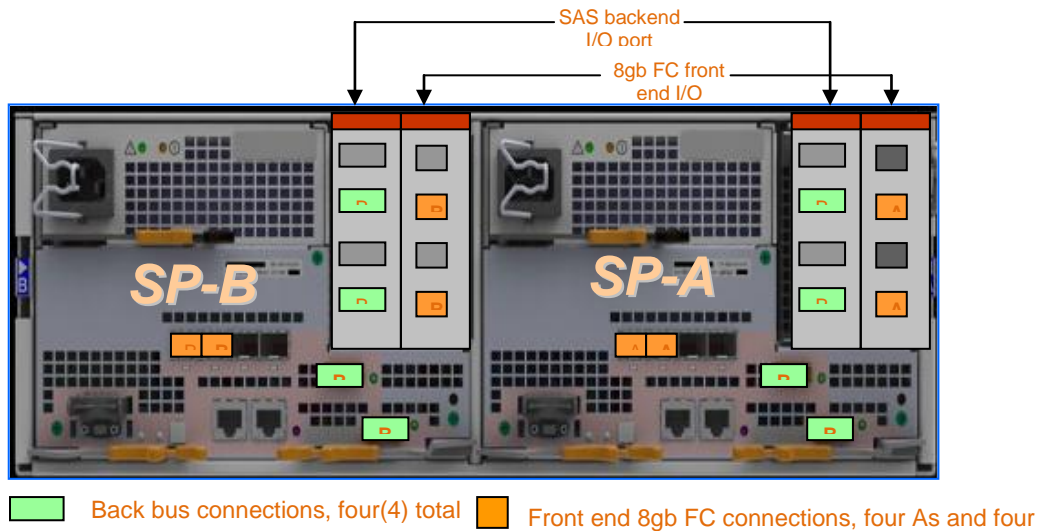


Figure 5. Recommended connection model of VNX5500

Read bandwidth for VNX5700 and VNX7500

For the high-end products in the family, the backend bus architecture supports multiple four-channel, 6 Gb SAS buses through I/O modules (as opposed to an integrated fixed number of backend buses). Hence, backend bus scalability is generally not what limits the maximum bandwidth achievable from these systems. These also support multiple I/O modules for different protocol front-side connections, which means that front-side connectivity capacity can also be modularly scaled. So, the theoretical limit is generally what the storage system’s processors and memory system can support, assuming enough drives are deployed. For bandwidth-focused deployments, the maximum number of drives supported by these systems’ software is generally more than sufficient to fully engage all the bandwidth power that the storage processing system components are designed to deliver.

For DW-oriented deployments, based on EMC Engineering experience, we recommend using around 6,000 MB/s as a practical expectation for a VNX5700, and between 9,200 – 10,000 MB/s for the top-end VNX7500™.

As stated earlier, in as much as the trend of many of the current DSS/BI deployments against large data warehouses tend to emphasize the importance of sustained read bandwidth from the supporting infrastructure, the actual need is still often very much dependent on how the data model and the user applications are designed and implemented. Depending on the DBMSs, the actual data model and implementation, and the actual server(s) used for driving the DBMS system and the associated

applications, the actual I/O pattern may vary, and the host I/O may or may not in fact need to drive a VNX system right up to the practical limit. The practical bandwidth guidelines provided here are useful to help with planning the right VNX system, and the right number of systems, to be factored into a new deployment, or to upgrade/augment an existing deployment.

Choosing DAEs and drive types

The VNX system supports two types of DAEs. The 3U form factor DAE continues to support 15 drives with a 3.5-inch drive form factor.

The new 2U DAEs are designed to support the new 2.5-inch disk drives that are becoming more popular in the industry. Because of the smaller drive form factor, the same-width DAE design holds up to 25 drives within one enclosure, and therefore provides much higher storage density and compactness.

The DAE structures are designed to house different drive types, as long as they fit in the 3.5-inch or 2.5-inch drive housing. These include Flash drives, and SAS 10k or 15k drives, as well as high-density SATA or Near Line (NL) SAS drives at lower rotational speeds of 7,200 or 5,400 rpm.

The read (and write) bandwidth that is achievable from the different drives, to a large extent, depends on the size of I/O that is being driven down from the host. Without any rotational parts, Flash drives tend to be able to achieve a more constant level of peak bandwidth. However, even with Flash drives, the larger I/O sizes allow the data streaming through the Flash memory data channels to be steadier, thereby ensuring that the peak bandwidth is more predictably achieved.

With rotating drives, data retrieval starts only after the drive head is correctly positioned to the track and sector where the needed data is located. Hence, the best bandwidth achievable from a drive is attained when we pull as many bits as we can off the drive before repositioning the drive head.

With the amount of data that can be stored within today's high-capacity drives, it is rare that there is only one single database table packed entirely within a single drive (or a small number of drives). And even if the physical data implementation model was done to ensure that rows from the same table are packed tightly together inside the physical drive, the fact that in most real-life deployments, there are now frequently multiple users running concurrent queries accessing perhaps a different set of rows within the same table at a particular point in time, the net result is that the most realistic data read pattern from the drives is going to be "chunk" random, as opposed to a true sequential access. A single user may be reading the table from the beginning. So, the storage read pattern may be 256 KB for the first 32 pages of 8 KB from this table, followed by the next 256 KB, and so on.

However, when there is the presence of a second user, it is now very likely that the second user stream may be asking for the data at 1 GB into the table, for the set of 32 pages (256 KB reads), followed by the next 256 KB (reading at a disk offset of 1 GB + 256 KB, and so on). To accommodate the data need of both users, the DBMS will be issuing I/O to the storage system that *requires* the disk I/O to alternate between the beginning of the drive, and the sectors 1 GB into the drive. So, the drive head would have to seek back and forth, then pulling up 256 KB each time. If we assume we can move the drive head 200 times a second, and we pull up 256 KB from the drive each time, we can get 50 MB/s out of the drive. Because of the physics of the drives, it is

usually not possible to move the head mechanically, then read the disk sectors, much more than 200-250 times a second even on the 15k SAS drives.

How much data we typically will try to pull off a drive before moving the head again is a function of the I/O request size from the host. While the storage system may sometimes leverage statistics of observed data access patterns to trigger predictive LUN data prefetching (causing more data to be read from the drive as the head arrives at the right position than what the host calls for), most of the time the storage tries to retrieve and deliver *exactly* what the host application asks for (and needs). Storage predictive “read-ahead” heuristic algorithms can be a two-edge sword for the type of “chunk” random read I/O mix in the DSS/BI application environments. If the storage attempts to “read-ahead” and gets more table pages into the storage cache memory in anticipation of User 1 needing those table pages eventually, that will come at the expense of delaying the pages needed by User 2’s query to be delivered. And since the parallel processing model favors a balance flow and execution of subtasks that are running in parallel, rather than expediting some of the parallel tasks’ executions, implicit storage read-ahead can sometimes net a negative impact to the total smooth parallelized query execution time.

In a sense, for most of the DBMS offerings handling large complex queries as parallel subtasks, each system is essentially orchestrating its own “table data read-ahead.” The DBMS organizes the subtasks in such a way that one sub-process, or process thread, is engaged with pushing a data read I/O to the supporting storage system “to get the next batch of needed table data ready,” while another process or thread is assigned to process the data already retrieved. The DBMS attempts to balance the amount of table data being prefetched against how fast it can “consume” the data already fetched. It also has to manage and arrange for data buffers to be created and allocated to ensure the data flow and processing pipeline can be maintained. Especially when contending with the DBMS running multiple parallelized queries for many concurrent users, it is often the best idea to keep the storage processing as straightforward as possible, rather than second-guessing what the DBMS engine is trying to do next.

Different DBMS engines have different configuration/implementation options to influence the typical single biggest “chunk” I/O that it will drive down to the underlying storage.

For example, with Oracle, the database instance’s configuration parameter, `db_file_multiblock_read_count`, can be set in the instance’s `init.ora` parameter file to designate how “aggressively” the DBMS should arrange to “prefetch” data tables needed to run user queries. If a site is using Oracle Automatic Storage Management (ASM) facilities to manage the data storing, a common and natural setting for this parameter is to set it to a value of:

```
Db_file_multiblock_read_count = ASM_allocation_unit_size /  
DB_page_size
```

Typically, the default ASM allocation unit size is 1 MB. That means up to 1 MB worth of database pages for the same table file will be packed together on the ASM disk. So, if the DB page size used for the database is 16 KB, the parameter should be set to $1\text{MB}/16\text{K} = 64$ pages’ worth.

With Microsoft SQL Server, the default EXTENT size for how table file space gets allocated on an NTFS file system used for holding the data is 64 KB. However, a SQL Server engine has the knowledge about the actual space layout of the table in the NTFS file system, and can in fact issue up to eight consecutive extents of 64 KB (512 KB) as a single host I/O to get as much consecutively stored data that is relevant for the query executing. The ability to pack eight extents physically together on underlying storage supporting that NTFS file system depends on how the different DB files are created and loaded. If multiple DB files are loaded concurrently, all going into the same file system, the likelihood of having all eight extents of data pages from the same table being packed together is reduced.

DB2 up to this point typically works with 256 KB extents. There is an instance configuration parameter that would try to logically prefetch up to as many 256 KB extents as there are physical drives supporting the storage LUN that holds the table file. So, on a traditional 3+P RAID configuration on IBM storage, their recommendation would be to set logical data prefetching to 1 MB. This also happens to be the right setting for VNX LUNs that are organized as RAID striping over a number of drives that are powers of 2, such as 2+1R5, 4+1R5, 2+2R10, and so on.

EMC Greenplum logically does not try to anticipate and issue large I/O to “prefetch” data from inside the DBMS kernel. Instead, Greenplum’s deployment best practice recommends certain supporting OS file system prefetch settings (such as using `blockdev –setra` on the OS disk device supporting the XFS file system that holds the Greenplum database files) to achieve the same purpose. The XFS file system block device read-ahead is typically set at 4 MB – 8 MB.

But even with the different methods of setting DBMS logical data prefetching, how the OS I/O actually translates down to storage I/O may still vary from deployment to deployment, or for that matter, from run to run of business queries. When there are only a few concurrent “heavy duty” queries running in the system, with each query logically reading up significant size chunks of stored data in serial fashion, the resulting accesses to the physical rotating drives would be more clustered together. As a result, the drives may deliver *more* sustained MB/s per drive under this user load, because there is less drive head seeking back and forth to find the right piece of data that needs to be delivered next. As the concurrent user load is increased, and the DBMS tries to accommodate running more user work in parallel, there may naturally be more data seeks back and forth to provide the data needed by the higher number of concurrent users and queries being processed together. The net MB/s of data delivered from the drive will drop somewhat.

For a general sizing guideline, EMC Engineering experience suggests using a rule of thumb of a more conservative 35 MB/s on the 10k SAS drives, and about 42 MB/s on a 15k SAS drive. For SATA and NL-SAS drives, 20-25 MB/s is about the most one should assume with a mix of read/write I/O hitting the storage.

The numbers quoted here are somewhat higher compared to the guideline numbers cited in [EMC CLARiiON Best Practices for Performance and Availability: Release 30 Firmware Upgrade](#). This is primarily due to the fact that in DSS/BI-oriented deployments, typical user work streams tend to be pushing large chunk size I/O in the range of 256 KB to 512 KB against the underlying storage systems, and each user stream tends to be accessing and adding data in relatively sequential manner. The enhanced backend I/O dispatch through the multilane, high-bandwidth SAS backend

bus, and the generally more efficient data buffering and CPU processing power of the VNX system hardware, favor the bigger I/O requests common in this type of deployments. Hence, one can be more optimistic in terms of the numbers used as a bandwidth estimate per drive in the VNX systems targeting primarily DSS/BI-type deployment.

Keep in mind that even if the storage RAID choice is parity RAID (for example, 4+1R5) or mirrored RAID (for example, 2+2R10), all drives in the RAID set would contribute to deliver read data. In the case of parity RAID, all five drives will be holding user data in that 4+1R5 group of drives, while in a 4+4R10 LUN, the storage firmware is frequently able to leverage the mirrored data to simultaneously satisfy two separate user I/O requests going after reading a different part of the LUN (that is, reading the data needed by User 1 from drives 1 and 3, and the data needed by User 2 from drives 2 and 4, simultaneously).

As an example, for a VNX5100 set of processors, which is expected to practically deliver a sustained 3,000 MB/s using a 4 x 8 Gb front-side FC connect, if we deploy a pool of 70 drives of 15k rpm using the default 4+1R5 protection, that would be about the right configuration to ensure that there are enough drives to support pushing the entire 3,000 MB/s that the storage system front side can be expected to deliver. Of course, the drive usage capacity would have to satisfy what is needed for the deployment purpose, but with drive density today typically at 600 GB or more, it is frequently the case that if there are enough physical drives configured to deliver to the performance, usable capacity is not an issue.

Flash drives can be counted on to deliver about 220 MB/s per drive. In that regard, when it comes to providing a sustained bandwidth feed, a single Flash drive will support about the equivalent of five to six SAS drives' worth of sustainable data read feed. In addition to the higher cost of additional Flash drives to support RAID protection (mirrored or parity), the more common problem is that the Flash drives will not provide enough usable capacity. Logically speaking, it would take no more than 15 Flash drives to deliver more than the 3,000 MB/s that the VNX5100 processor heads can move. Even when choosing to employ a wide parity RAID, such as 14+1R5, using 200 GB drives, in order to minimize the cost of the extra drives needed to support storing the RAID protection data, the total usable capacity that can be held in the drives would be a little over 2 TB.

For this reason, Flash drives are generally not recommended for holding a massive volume of detail facts data that may need to be scanned through. Flash drives should be selectively used to target database objects such as summary tables, temporary tables, cubes, and indices that have a more random accessed pattern, where usually the volume of data involved is not massive (meaning in the range of terabytes to hundreds of gigabytes), but where access service latency is important. (For example, a set of data is not usable as a sorted list of data until the last entry of the dataset has been installed into the SORTED DATA table in the right order. So, any query that includes GROUP BY or ORDERED BY cannot be processed correctly until the data sort is completed.)

The VNX5100, VNX5300, and VNX5500 storage processors all support integrated 4 x 8 Gb FC front-side connections per SP. Typically, two of the four integrated ports are intended to support host FC block connections, with the other two are reserved for file-based accessed connections. It is possible to use more than two of the four

Front-side
connectivity
choices for DW
deployments

integrated ports in each SP, but there is no significant performance that can be gained from deploying more than two ports on each SP. One can choose to deploy all four ports in each SP to simplify host-to-storage connection cabling, and perhaps bypass the need to other SAN switching networking.

The VNX5300 and VNX5500 allow two additional I/O front-side connection modules per storage processor to be configured in. The following three connection modules are available:

- Four ports of FC connections, with auto-speed recognized to support up to an 8 Gb speed for each port
- Dual-ported iSCSI connections running up to a 10 Gb speed per port
- Dual-ported FCoE connections running up to a 10 Gb speed per port

The connection modules are expected to be configured in pairs. If a FC connection module is configured into the first expansion slot of SP-A, a like kind FC connection module should be configured into the same expansion slot of SP-B; similarly for the other connection types.

When deploying with FC protocols, a practical bandwidth assumption is to work with an estimate of about 1,500 MB/s per I/O module on the VNX5100/5300/5500, in spite of the theoretical 4 x 8 Gb of FC connections that can be run from the servers to the I/O module's four ports. Again, the extra ports can be used to simplify server-to-storage connections, but as a performance working rule, stay with that general guideline for the FC module. On the high-end VNX5700/7500, the same FC I/O module can be expected to drive all the way close to 3,000 MB/s. So, if all four FC front-side connection ports on that module are used, we can expect reasonably close to 8 Gb wire speed on each FC connection port.

A similar sustained bandwidth guideline of 1,500 MB/s for each module should be used for the dual-ported FCoE or iSCSI connection modules, even though each of the ports by itself in the module can drive close to the full 10 Gb, given the right driving load from the host. In other words, if we deploy four FCoE modules in a VNX unit, and connect to one port only on each of the four modules, we can expect to drive each connection close to the 10 Gb wire speed. If we try to connect to all eight possible ports on those four modules, we will typically drive closer to 1,500 x 4 (or 6,000 MB/s) pushing I/O through all eight connections (within the limits of the SP system maximum).

Between the FCoE and iSCSI port drivers that are running inside the VNX systems to support the two different front-side connection protocols, the current iSCSI driver stack tends to be more memory bandwidth-intensive (more stack frame memory copying and bit moving involved to support the iSCSI protocol) compared to FCoE. So, as a generalization, FCoE connections tend to drive bandwidth-focused workloads more efficiently compared to the same number of iSCSI connections.

The iSCSI connections, however, generally allow a 10 Gb IP connection to direct-connect a server 10 Gb NIC or iSCSI offload interface card to a VNX front-side iSCSI port, while a FCoE connection more typically will go through a switch such as the Cisco Nexus 5000 class switch to support zoning. Access port zoning adds a bit more operational configuration complexity but helps to secure server-to-storage access paths. So, it is probably still a reasonable tradeoff to go with the FCoE if possible.

For a 10 Gb network connect, it is useful to have jumbo frame enabled for those ports on the server's NICs, as well as for the connections into the array's iSCSI front-side ports.

However, if converged network adapters (CNAs) such as those from Emulex, QLogic, or Brocade are used, the maximum frame size for protocols like FCoE is often negotiated between the switches (such as the Brocade 8000, or the Cisco Nexus 5000 series switches), and may not get anywhere as large as the 8,000 or 9,000 bytes that jumbo frame support may allow the switch ports to be configured for. However, those are typically bigger than the default MTU of 1,500 for most IP switch ports, so the IP switch ports may have to be configured for jumbo frame usage.

A mixture of connection protocols in the same array is in fact supported. It is possible, for example on the VNX5300, to connect one server via the integrated FC ports to the array, use one pair of iSCSI expansion modules to connect a second server to the same array, and with an FCoE pair for a third server to the same array. The three servers can all share the array, including sharing the same set of LUNs from the same array, such as running a three-node Oracle RAC cluster, with each node connecting to the server using a different block-based storage access protocol.

It has been a general engineering experience that deploying an uneven number of ports in these modules of the same type for connecting a single server to the array is often not the best thing to do. For example, we can connect up six FCoE connections from a server into the FCoE expansion modules, with two connections each going into Module 1 of each SP, and one connection each going into Module 2 of each SP. While in theory we can drive up to 1,500 MB/s from the two Module 1s, and 1,000 MB/s from the single port in Module 2s (for a total of 5,000 MB/s in theory across the two SPs), this actually does not always work out the best for the parallel query model. Again, remember that when the DBMS divides up the work, it assumes that all subtasks are going to be processing and doing I/O in the background at comparable speed. When some I/Os are pushed down the paths that double up on the SP front-side connection modules, versus other I/Os being sent down the path with a single connection only, the hardware queuing and service latency variations introduced would in fact be reflected in certain portions of the parallelized subtasks running faster (or slower) than others. It may not always hurt the overall query execution, but adding the two extra ports and the two modules with the single added port connection may also fail to show significant improvement as a result (since the overall query is still running at the speed of the slower, or slowest, execution paths).

It is not necessarily always a disastrous practice to run with uneven numbers of connections of the same type spread over different I/O modules. Using similar numbers of connections of the same type in different I/O modules (for example, using one 10 Gb FCoE port per FCoE module for a system with two FCoE modules on each SP) in general ensures a more balanced data push, even though in theory, the server should not really need as many connections to drive the I/O bandwidth needed.

There are tradeoffs in choosing one connection type versus another. VNX systems generally offer a number of FC block connections in most default configurations, because in the previous unified storage system customer base, block mode data stores and accesses had been through FC SAN switching infrastructures. The FC connections allow VNX systems to be deployed in an existing FC SAN infrastructure

with the minimal amount of adjustments required, and deliver an immediate boost for the SAN infrastructure.

Both iSCSI connections, as well as the new FCoE connection support, simplify the hardware needed to support both IP network traffic, as well as block I/O traffic, going into and out of the servers, leveraging CNAs that have built-in FCoE offloading, or iSCSI offloading support. The consolidation of both types of traffic into fewer PCI devices allows small form factors (such as 1U servers like the Dell R610, or the UCS blade servers) to be densely packed to support the ever-increasing massive data volume processing in the data warehouses growing into terabytes and petabytes.

Storage pools and auto-tiering

Organizing your physical drives into storage pools is the recommended approach for VNX storage systems, although storage concepts such as RAID groups and LUNs from a selected set of physical drives, as supported in previous generations of EMC unified storage, continue to be supported for VNX.

A key advantage with LUNs created out of storage pools is that the physical data layout of the LUN content is now under automatic management of the VNX storage system. Whereas a LUN of 1 TB that is allocated to the DBMS engine to be used to hold up to 1 TB of the data kept in the data warehouse used to be associated with a fixed set of physical drives in the storage system, that same 1 TB of stored data will be managed as even-sized (for example, 1 GB) slices, distributed in an optimal fashion, evenly among all drives assigned to make up the pool.

When a LUN of 1 TB is presented to the DBMS as “bit containers” to store the different database table files, there will then often be many different tables being stored inside the LUN. Even if the LUN is only holding different pieces of a large table, such as the sale receipt details of a product item from a particular business unit in the last five years, the nature of the application is such that a certain set of DB pages and table rows will get referenced more often than the other DB pages and table rows (for example, sales from last three months representing the current quarter will likely be queried with much high frequency than sales from months that were five years ago).

VNX systems continue to support the concept of RAID groups and traditional “fixed” LUNs in previous generations of the EMC midrange block storage systems. With the traditional LUN approach, with the LUN tied explicitly to a RAID group that spans a particular set of physical drives, the DBA or application programmers will have to identify the “hotter” tables, or table rows, extract them from the table files, and perhaps create a new table partition in a different LUN to try to segregate off those frequently accessed data pages or rows, in order to speed up the I/O part for the queries that are using the “hot” data repeatedly.

With virtual pool-based LUNs, VNX system software will be able to track and identify the different LUN data “slices” that are receiving the repeated references. The storage system will then automatically relocate those “slices” of that 1 TB LUN to drives that can provide improved I/O access service to the frequently used data.

As an example, as the business moves forward in time, the sales details relating to April, May, and June become the “most frequently accessed” data as opposed to the January, February, and March details (as the business moves from first quarter of the year into second quarter). The storage pool managed tiering mechanism picks up the

new data slices of the sales table with the current quarter’s data, migrating them into the high-performance drives and displacing those from the previous quarter.

Similarly, if we have two tables sitting together inside this 1 TB container, one used to hold sales details, and the other for building up a dimension cube, the typical business queries will tend to leverage the cube data in many of the frequent reports as opposed to going back to the sales detail data. Again, storage system auto-tiering will be able to detect the usage pattern, migrate the slices in the LUN associated with the cube to the higher-performing drives (for example, Flash drives used in the pool), thereby providing significantly improved service to the queries heavily leveraging the cube data.

For applications where it is operationally more convenient to divide certain data objects into separately addressable OS storage “containers,” the pool-based virtual LUN approach can be used to create *multiple* distinct OS disk LUNs. All LUNs created in the pool share all usable space provided by all drives, including drives of different tiers.

Through the virtual LUN data placement policy control, the LUNs that are expected to be holding the more frequently accessed data objects, or objects that need higher I/O service preferences, such as indices used frequently for JOINS, can be assigned a preference to be stored, or migrated over time, based on frequency of usage pattern, to the fastest storage tiers. For example, slices of data would be moved to the Flash drives. As a consequence, data in LUNs that are aged, and infrequently used, will, by natural progression, be displaced from the faster storage tier and moved into the slower tier, such as the slower, bulkier SATA or NL-SAS drives.

Virtual LUN drive pool

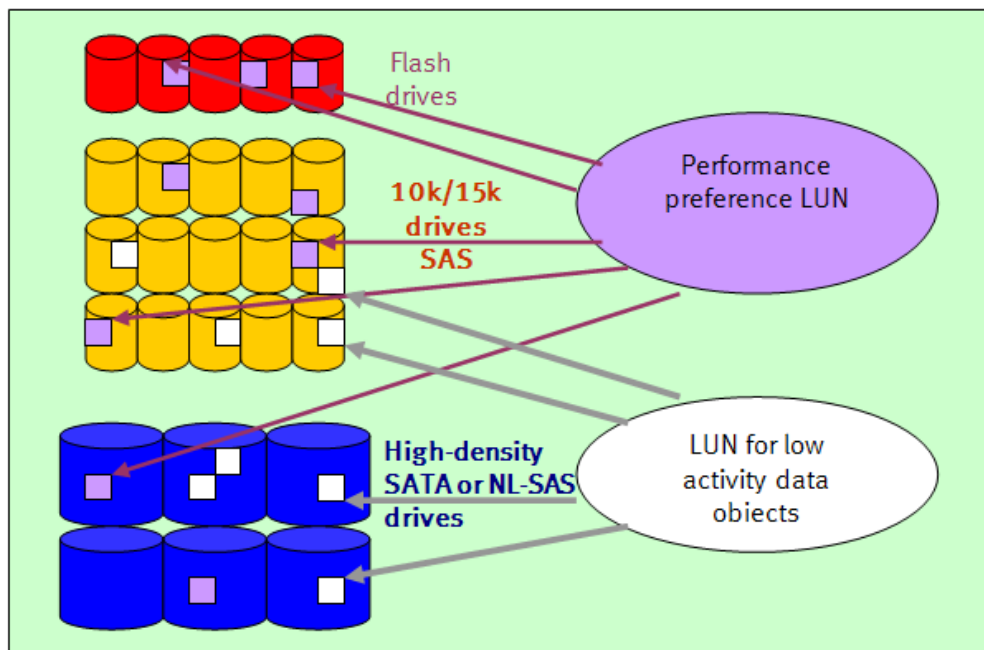


Figure 6. Multitier pool LUNs

How much Flash tier storage should be made available to the pool for supporting a data warehouse deployment? As usual, the right answer depends on understanding how the usage of the data stored in the warehouse changes over time, and when new users and applications are introduced or added. Two general guidelines are sensible to follow.

- Flash drives are generally *not* the option for holding detail fact data that is often accessed in bulk from the business applications. When gigabytes of raw fact data is read up from storage as part of a large table scan executed by a query, it is usually going to take time on the server to process this many gigabytes. Many DBMS vendors may suggest that a pure storage system read bandwidth calibration be done by executing a simple query like this:

```
SELECT COUNT(*) FROM a_very_large_fact_table;
```

The idea is that the DBMS will hit the storage system, pull up all pages of a table with many rows of data up as quickly as possible, and do practically nothing with the data (but incrementing a simple row counter). This is often used as a pure bandwidth gauge of the storage system. In reality, this type of simple query is relatively meaningless in any real-world business application. Putting all data of that table on Flash drives may make that query run very fast, but it would be a serious waste of money to do so. In fact, if `very_large_fact_table` is stored packed in a rotating 7,200 rpm SATA drive, it is readily possible to pull the table pages off the big drives at 40 MB/s or better. And by comparison, a Flash drive is going to do no more than 200 MB/s pulling the same data up.

- Implementations of business queries and reports that tend to create redundant data structures as part of the database schema to help with optimizing the amount of data, and the associated I/O needed to speed up queries, such as indices, summary tables, materialized views, and so on, are frequently the right targets for being housed in or migrated to Flash drives to improve query performance because they are used with high frequency. Since additional effort has to be made to create and maintain such additional data structures for the purpose of leveraging them to improve recurring usage performance, it stands to reason that these are suitable targets to move to Flash drives. Hence, if the DBA team and application developers can provide a reasonable estimate of how much “extra redundant data” has been planned for, then that would serve as a good guide for storage administrators to plan the Flash drives needed. Because of the high performance ratio to usable GB of capacity, it is often easier to size for capacity first, then verify that the number of drives planned is sufficient to deliver the performance expected (opposite to how one would often plan rotating drives to hold DW data).

Absolute bandwidth consideration between virtual pool LUNs and traditional fixed LUNs

With virtual pool LUNs, as mentioned above, slices of the LUN are evenly distributed among the drives making up the pool, including drives of different tiers based on the performance preference settings that the LUN is created with (and which can be subsequently adjusted).

Especially for rotating disks, it may appear that the fastest way to access a large amount of table data is to organize the different partitions of the table, and keep each partition on a fixed set of drives. By scanning the partitions all in parallel, but with each partition completely stored in a fixed set of drives, it would be the most efficient way to scan through the entire dataset without any spindle data access interference between the partitions.

Testing specifically in the lab simulating this very mode of data scans, it is in fact the case that by dividing the partitions statically into the traditional fixed LUNs on RAID group sets (and in particular, using narrow RAID configurations such as 2+1R5 and 2+2R10), that the scanning can be achieved more quickly by I/O simulation programs such as IOMETER, SQLIO, or ORION from DBMS vendors, compared to using a similar number of virtual pool LUNs.

But stepping back and looking at the more realistic view, it is becoming rare that stored data is scanned exclusively from a single user running a query. Once we take into account that most of the time there would be multiple user queries running, going to the data warehouse looking for different portions of data stored, the ultimate data read pattern from the mix of concurrent user queries is going to be randomly accessing different parts of all drives holding the data partitions in any case. The more concurrent users there are, the more random the actual I/O access pattern becomes against all drives holding the data. The advantage of being able to stream data steadily off the rotating drives supporting the traditional fixed LUNs is diluted when the concurrency of data usage increases. The convenience and flexibility offered by using pools and virtual LUNs will then outweigh the percentage of pure bandwidth advantage that a carefully planned layout using fixed LUNs may have over the virtual LUNs in a pool. Most often, the key difference is that a pool may have to be configured using more physical drives (to provide for a wider data spread) as compared to the fixed LUN over RAID group layouts in order to drive to the same level of sustained bandwidth. But from a host application perspective, the same number of virtual “LUNs” from the pool can match performance against a similar number of fixed LUNs using fewer physical drives. The more random the actual I/O access pattern is (which dilutes the advantage of planning for tight physical storage adjacencies to improve access bandwidth to drives), the smaller the difference will be between pool-based virtual LUNs versus fixed LUNs.

Virtual pool LUNs

Virtual pool LUNs can be created either as “thin” LUNs, or fully allocated LUNs, sometimes referred to as “thick” LUNs. If a pool LUN of 1 TB is created, only an initial slice of a couple of gigabytes is actually allocated and consumed from the pool. As data is written into the LUN, more space will be allocated from the pool to accommodate the added data.

Thin LUNs provide for the most optimal use of available space in the pool. The space in the pool is only assigned and used when the host actually writes data into the LUNs. Hence, it is logically possible to accommodate 10 application user requests for 10 x 1 TB LUNs, even though the pool has no more than 5 TB worth of assigned space. Assuming all 10 LUNs grow steadily in capacity, storage administrators may be able to defer expanding the storage system with more physical capacity until the total usage starts to get close to the 5 TB available.

Because the key objective for thin LUNs is to maximize usable space capacity, incremental “extents” of space assigned to each LUN when the LUN grows are by design added in relatively small space chunks. While this helps to optimize space usage, it does have the side effect of breaking up the LUN into many small space fragments spread over many physical drives. This hurts the rate by which large chunks of data stored in the LUN can be retrieved when the queries are driving large chunk “random” data scans.

Fully allocated LUNs, or thick LUNs, will immediately reserve all the space needed for each LUN created from the pool. So, it is *not* possible to create 10 thick LUNs of 10 TB each from a pool with a total usable capacity of 5 TB. The pool must first be expanded with more physical drives to be able to hold the entire 10 TB before all the LUNs can be created.

Because the thick LUN space is reserved, it is generally assumed that the application expects to be using all the requested space quickly. Hence, when a new “extent” is allocated, a new slice of 1 GB worth of space is allocated from the drives in the pool. Because of this large allocation extent size, one can expect that each 1 GB worth of LUN data would end up physically residing in and spanning some physical drives. Large data scans will get the benefit of being able to pull up good size data chunks from each drive in the pool before needing to reposition the head, thereby achieving the more optimal read bandwidth from each drive.

Hence, for VNX systems, the following would be guidelines for choosing between using pool virtual LUNs versus traditional fixed LUNs:

- First preference should be given to creating a pool, and deploying with fully allocated, or “thick” LUNs.
- Generally expect a slightly lower MB/s estimate per drive when using pool-based “thick” LUNs compared to carefully planned traditional fixed LUNs. Deploy more physical drives (for example, 10 percent to 15 percent) if necessary to address any performance concern about the maximum data retrieval rate that can be achieved from rotating disk drives.
- If the stored data clearly includes the “redundant data” created for the purpose of speeding up high impact queries, consider boosting the pool with Flash drives with enough capacity to effectively house the expected “extra data.” The amount of Flash drive needed should be sized based on the size of the “extra data” expected.
- Examine the expected user processing model. If the stored data is intended to be primarily scanned in bulk by a small number of user sessions that are scheduled, to do very heavy duty and focused deep data exploration or analysis, it may be a deployment situation where using traditional fixed LUNs, or perhaps a combination of fixed LUNs and pool LUNs, may be sensible. Traditional fixed LUNs from fixed RAID groups of drives generally can deliver equal or better bandwidth from the VNX system compared to a pool with a similar number of drives, but this is practically realizable only if the host read pattern is not so randomized by concurrent user queries that turn what is logically a relatively sequential data access pattern into a highly random one.

VNX system cache setting for DW deployments

VNX systems have two separate levels of data caching that can be configured for the system — the storage processors' memory cache and, leveraging Flash drives, an extended FAST Cache.

Even with caching enabled (SP memory cache and Flash-enabled extend FAST Cache), the storage administrator can coordinate with the DBA and application developers to decide if data caching should be applied to specific data objects stored in the VNX system. For traditional fixed LUNs, read/write caching, with SP memory, or with FAST Cache, can be selected for each LUN. For pool-based virtual LUNs, caching attributes are associated with the pool that the LUNs are created from. If it is desired to create data objects from virtual LUNs that use different caching attributes, this can be achieved by creating different pools from a different set of physical drives.

Storage processor cache memory consideration

Different models of the VNX family support different amounts of storage processor memory, ranging from 4 GB per processor for the VNX5100, to 32 GB per processor for the VNX7500.

A portion of memory in each storage processor is taken up by the running system software, and data structures supporting the various storage features, such as FAST tiering, FAST Cache support, virtual LUN mapping, and data replication software such as SnapView™, MirrorView™, and RecoverPoint splitter drivers. Some memory is used as data move and copy buffers. The remainder of memory left over can be designated as data caching buffers.

Data caching buffers can be configured from 2 KB per slot up to 16 KB. A simple and logical rule is to configure the cache buffer size to match as closely as possible to the database logical data page size. For example, for SQL Server, that would normally be 8 KB. For DBMSs that allow the DB page size to be configured for higher than 16 KB, use 16 KB, the largest size supported currently on VNX systems.

The amount of available SP memory to be configured for data caching purposes can be partitioned into a portion for read caching, and the other portion for write caching.

With the typical DSS/BI processing typically going through a considerable volume of data, the benefit of SP read caching is generally minimal. If a query scans through 1 TB of data, trying to hold on to the first 4 GB of that 1 TB table is not particularly helpful to speed up another query that needs to pour through the same 1 TB again.

SP read caching will usually only help if, as we are reading up the chunks of this 1 TB dataset to the host per the DBMS request, that the storage can “anticipate” and start reading ahead the next chunk of this dataset to get it “ready” in the read cache to speed up the next host request for this next chunk. This assumes that a) the entire 1 TB is stored in the same LUN, and b) there are no interfering requests from the host that needs to read other chunks simultaneously from either the *same* LUN, or different LUNs, creating a cache memory contention for the storage read-ahead data. If an SP tries to trigger heuristic data read-ahead for many LUNs that the host appears to be all reading sequentially, we can potentially get into a prefetched data thrashing situation, resulting in prefetched data being displaced before the host applications ask for them, creating wasted effort and slowing actual data delivery (because the displaced data will have to be read again from the drives).

Therefore, it is generally better to limit the amount of read cache memory to no more than 300-500 MB per SP. If the SP is truly contending with a single stream (or very few concurrent streams) of true LUN sequential reads, the 300-500 MB of memory is generally sufficient to provide the heuristic read-ahead buffer space. If the deployment expects high concurrent user queries simultaneously scanning different tables in different LUNs, it may be advisable to forget configuring read cache memory at all.

Unlike Online Transaction Processing (OLTP) applications, data rehit from the SP write cache for DB data is also not common. Most of the data writes in DW-related deployments are from new data being ingested into the warehouse. But in many practical DSS/BI queries, these are implemented without requiring real-time data accuracy to simplify the implementation.

For example, if a business query is trying to determine the sales trend of a newly introduced product within the last month after the product came into the market, that business query or report can often be performed *without* counting the number of units that were sold within the last five seconds. The trending answer is typically not materially affected by what happens in the last five seconds, and an accurate, real-time answer is not needed. Furthermore, if the query needs all last 30 days' worth of sales details, the fact that the last five seconds of real-time data can be retrieved immediately back from write cache will not materially affect the time it would take to perform the entire query, since most of the data needed will still have to be read back from disks.

Typically, there are two key areas where the write cache memory in the SP helps. First, during a new data ingest, the newly created DB pages of data can be quickly absorbed into the write cache memory. As newly ingested data logically tends to go into a table as consecutive new DB pages that reside together in the LUNs used as DB containers, the write cache offers an “new data staging” area where the SP would have a chance to try to coalesce some of the new DB pages together, and dispatch them more efficiently to the physical drives on the back end as larger single drive write requests. This is especially important if the LUNs are sitting in a pool, or in a fixed RAID group, using parity RAID.

Holding the newly ingested DB pages in SP write cache, deferring the physical backend drive writes to allow for coalescing into more efficient large writes, also has the effect of minimizing the backend drive data write contention against the data reads from those same drives. It is becoming increasingly difficult for any operation to completely separate out the new data load operation from normal user query operations. There is just no longer the luxury of an exclusive operation time window where the data warehouse database is “not open to user queries” just to accommodate new data load, and database maintenance. By leveraging the write cache, foreground user queries that also need to read from the same physical drives will not suffer as much of a read delay due to the write I/O traffic.

SP write cache can also be quite useful for improving queries that include some amount of data SORTING. This includes queries that try to group, rank, and summarize data in the dataset such as queries with GROUP BY, ORDER BY, and others. Some queries may still be optimized by the DBMS query optimizer to do merge joins using a sorted list of the join keys.

DBMS generally tries to perform the data sorting in memory if possible. However, if the set of relevant data involved is too large to be held in memory, the sorting is achieved by writing a partially sorted list of memory data to temporary sort files, then reading those files back and merging them into a properly sorted file.

So, whenever an intermediate file is written as a result of a sorting operation, it is given that shortly after the data is written into a temporary sort file, it will be read back as part of the query processing. Hence, this is the one area where the data written from the host has a very high likelihood of being immediately read back and used. Also important to note is that in any parallel subtask processing model, any processing that requires some level of sorting becomes a serialization point. If we need to create an ordered list of data values, we may be able to scan all values as parallel subtasks, and order each sub-list within each subtask. But in order to create the entire sorted list, every subtask has to wait on the slowest subtask to complete its portion, before merging of the sub-list can begin. So, the faster we can sort each sub-list in each subtask, and to merge the sorted sub-lists, the faster the query will be completed.

FAST Cache leverages Flash drives

If Flash drives are present in a VNX system, and the FAST Cache feature is enabled for that system, then some of these very high-performances, low access latency drives can be used to support a second, and extended, level of SP data caching. Unlike SP memory cache, which is physically limited by the amount of physical memory we can put into a particular VNX system's storage processor, the amount of Flash drive-enabled extend cache is mostly limited by the number of drives we choose to deploy. For every VNX system, there is a maximum number of Flash drives that can be used for the purpose of configuring the extend cache. This is a function of the amount of memory that can be taken from the SP total allotment without creating memory contention that would otherwise affect the other normal processing need of the SPs.

Whereas with the SP memory cache, we are typically limited to a number of gigabytes of caching memory, and that often becomes not particularly effective when we are trying to cache the most often used portion of many terabytes of data, with FAST Cache, it is possible to configure up to a few terabytes of cache.

When the application implements extra data structures such as cubes, materialized views, indices, and others against the dataset with terabytes' worth of raw data, these data structures tend to become proportionally large. Dimension cubes in tens to hundreds of gigabytes are not uncommon, and there are often multiples of these cubes generated to optimize for different business purposes. Unstructured data searches common in many web applications such as search engines often create hot raw datasets that are quite substantial in size, and many such larger data objects may peak in their usage concurrently.

FAST Cache can be configured to support both read-only data caching and read-write data caching. For read-only data caching, parity RAID can be used, offering more usable caching space. For read-write caching, mirrored RAID (R10) must be used, effectively offering only half of the cache space from the total number of Flash drives deployed.

FAST Cache works on a LUN data “slice” reuse frequency principle. LUN data is tracked in the current design on a 64 KB “track” basis. A usage tracking map is maintained for each LUN that can leverage FAST caching. When the same 64 KB track is referenced repeatedly at a system heuristically determined frequency within a certain time window, it is deemed to be “hot enough” to be qualified for being cached in the extended cache. When a new data LUN track is moved into FAST Cache, it will possibly displace another track that is FAST Cache-based on a general LRU algorithm.

Choosing between deploying for Flash drives for FAST caching versus FAST auto-tiering use

In as much as data in the FAST Cache is kept on Flash drives with RAID protection, and the data is theoretically persistent even in the event of a power failure of the system, FAST Cache is always intended to hold copies of data from LUNs that are stored on other drives in the same VNX system. Data track “copies” from many different LUNs may be present in the drives making up the FAST Cache at any one time, and that information is maintained by the VNX system in a FAST Cache mapping memory table in the SP memory. For read-write FAST Cache, that mapping table is persistent even across a system failure, ensuring that no committed write data left in the FAST write cache will ever be lost.

While the objectives of FAST tiering and FAST Cache are similar in trying to ensure that the data with the highest usage frequency is kept in the Flash drive to improve application data I/O performance, FAST Cache is generally more useful when working with datasets that tend to have a more dynamic and fluctuating pattern that is more difficult to predict or anticipate over time.

If we specifically organize data schemas in the warehouse to implement cubes, for example, we know we intend to rely on, and use, the data in the cubes, and ideally, as often as possible, then FAST tiering would be a logical approach. The storage system senses the usage frequency of the cube data as stored in the different slices within the pool, and correctly migrates the slices to be stored in the Flash drives within the pool.

FAST Cache extends the data caching characteristics of the SP memory cache, and is better suited for contending with multiple datasets that may be needed by different applications over time, driven by changing business needs. For example, tables that contain data from operational branches throughout the world likely will be hit with different degrees of frequency around the clock. Certain indices may be used very heavily by certain sets of business reports that usually run at a particular business operational window. To accommodate a usage pattern that is likely to change much more readily over time, and openly triggered by changes in the business operations, FAST Cache would deliver the hoped-for speed improvement from the I/O subsystem more rapidly in reaction to those changes.

Note that the capacity of the Flash drives is part of the total capacity that can be used to store data with FAST tiering. With FAST Cache, the capacity of the drives contributes to higher operational I/O performance, but “data of record” is still stored in other drives, not in Flash drives, as the permanent copy.

Temporary data such as TEMP table spaces used to support out of memory sorts should not be stored in the highest-performance FAST tier, since TEMP files are really nothing more than “scratch pads.” There may be frequent activities against the sort file area during certain queries, where the performance of Flash drives can help

tremendously. With FAST Cache, the usage pattern on TEMP to support these sorts will end up pushing the TEMP space in FAST Cache.

A combination of FAST tiering and FAST Cache can be very practical for many DW-oriented applications. As usual, understanding how the stored data will generally be used by the applications is key in helping to make the right choice.

EMC engineering case studies deploying VNX5300 with an Oracle DW test workload

Overview

An Oracle DW test workload provided by the Oracle DW Engineering team to their partners to facilitate evaluation of how particular components in a total DW solution system could affect the effectiveness of the entire system was used. This enabled EMC Engineering to study the effect of pairing the right EMC storage platform and configuration with particular servers to most effectively support running typical Oracle DW workloads.

This workload is built on point-of-sales data primarily partitioned by date and sub-partitioned by hash. A series of “typical queries” is run against the data by a user stream. The elapsed time taken to finish all queries in this stream is taken as a reference point. An “acceptable” user work stream service time is then established based on this reference point.

The test mechanism then allows multiple concurrent user streams, each executing that same series of queries, though possibly in different query request orders, to be simultaneously initiated and run against the system under test. The key metrics are the number of concurrent user streams that can be run on the system under test simultaneously before the prescribed user service stream response time is exceeded.

For any system under test, the idea is that the server(s) and supporting storage(s) should be calibrated and balanced as closely as possible to sustain the processing of as many concurrent user query streams as possible.

In characterizing the performance of the new VNX platforms, EMC Engineering assembled a new system under a test configuration (with more current and powerful servers) to match up with a new VNX storage platform, and ran this test workload to compare the ability to scale for concurrent users.

Commodity servers are getting more compact and more powerful. To be able to provide enough I/O balance, the VNX storage platform used to balance one of the newer, more compact servers would have to also be capable of delivering more sustained bandwidth in general. Furthermore, because it is possible to pack more processing power in the same server footprint, it follows that not only would the VNX platform be able to push to a higher bandwidth envelope on each unit, but it must also be able to do so in a smaller footprint, or else the advantage of the compactness from the new server investment would be greatly diluted.

EMC Engineering worked with Oracle and different server partners in the past to create reference configurations of past servers and EMC storage that result in the most optimal balance. With those reference system configurations, a concurrent user scaling limit based on the concurrent user query test workload had been established and published in the past. At the time of those publications, those were the “most optimal” configurations based on the prevailing server and storage product options available.

With newer, more powerful, and more compact servers now matched to the new VNX platforms, newer “optimal” system configurations have been defined and tested.

The case studies compared the configurations from the past to the new configurations defined that include the newer servers supported by the properly balanced VNX storage system configuration. The comparison reflects the advantage expected should a site move from the previous “balanced” reference configuration to the new configuration. The advantages include a system solution that yields more power for user support, in a reduced hardware footprint, and at a significantly reduced cost.

Case Study 1: Dell rack servers with unified storage

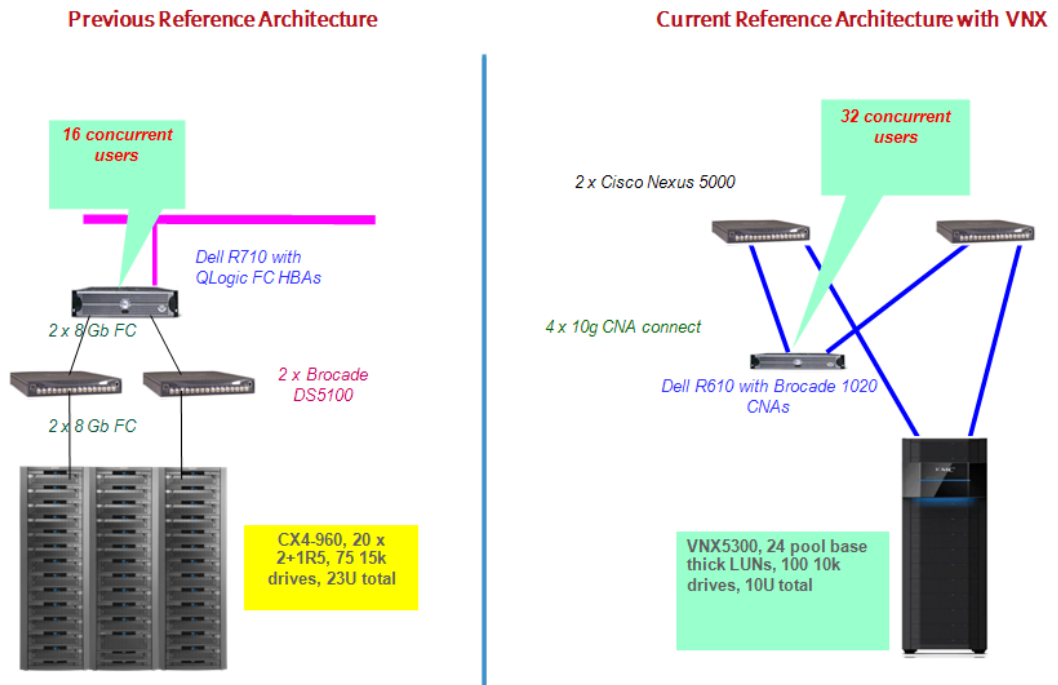


Figure 7. Dell server with unified storage reference configuration comparisons

Testing was once conducted with the following configuration:

- Dell R710 with two-socket, quad-core Nehalem processors at 2.33 GHz and 32 GB RAM
- EMC CLARiiON® CX4-960 with 75 FC drives of 300 GB @ 15k rpm, 3.5-inch drives
- Five disk array enclosures with 3U per enclosure
- 19U SAN storage footprint
- Traditional 2+1R5 fixed LUNs (20 LUNs)
- Oracle Real Application Cluster Database 11gR1
- Oracle Enterprise Linux 5.4
- 10 TB of a test database with 50 percent Oracle data compression

Under that test configuration with the Oracle test workload, 16 concurrent user streams were executed while staying within the prescribed service-level requirements. The new tested Reference Architecture includes:

- Dell R610 server with two-socket, hex-core Westmere processors and 64 GB RAM
- VNX5300 with 100 SAS drives of 600 GB @ 10k rpm, 2.5-inch drives
- 4 x 10 Gb FCoE connections
- Four DAEs with 2U per enclosure
- 10U SAN storage footprint
- 24 pool-based thick LUNs
- Oracle Real Application Cluster Database 11gR2
- Oracle Enterprise Linux 5.5
- 10 TB of a test database with 50 percent Oracle data compression

Compared to the previously tested system, the new system configuration delivers:

- Twice the number of concurrent users supported
- Reduced server/storage footprint from 19U to 10U (48 percent reduction)
- Almost four times the total usable storage capacity
- Flexibility of a pool-based LUN setup as opposed to effort in configuring custom narrow RAID fixed LUNs to push sufficient bandwidth required from storage
- Lower total storage system cost

Case Study 2: Deploying with UCS blade servers

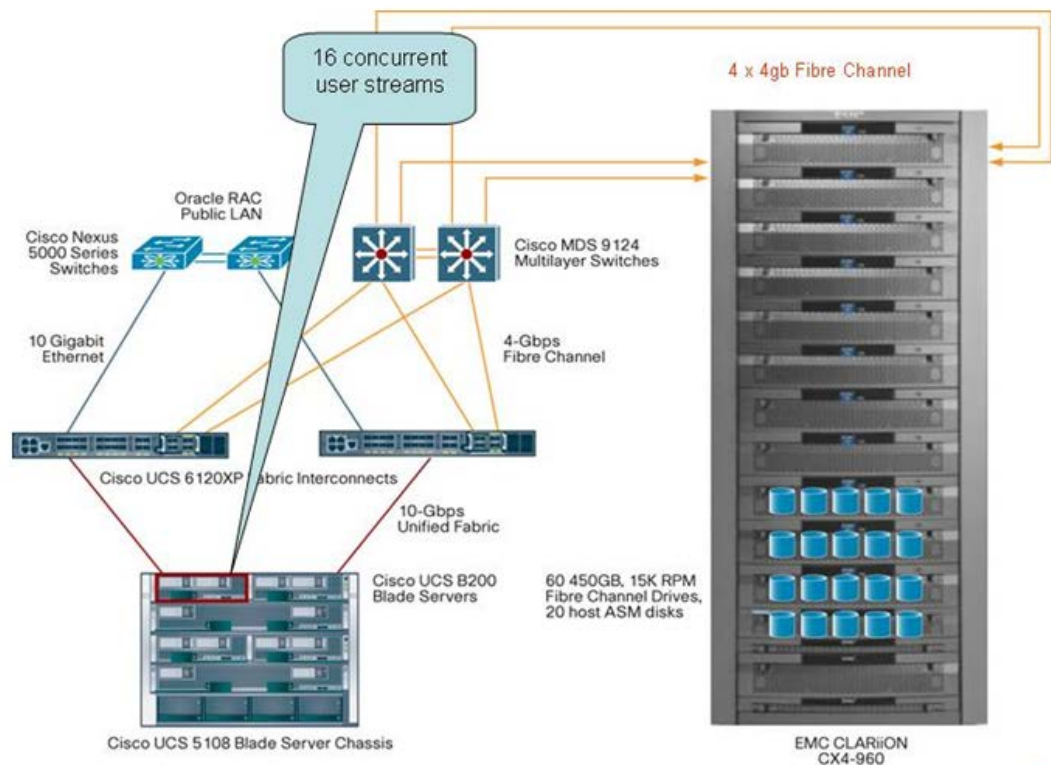


Figure 8. Previous Reference Architecture with a Cisco UCS B200 M1 blade server

The configuration tested together with the Cisco server division and Oracle over a year ago included the following:

- UCS half-width B200 M1 blade with 4 x 4 Gb FC virtual HBAs, and two-socket, quad-core Nehalem processors with 48 GB RAM
- Oracle Real Application Cluster Database 11gR1
- Oracle Enterprise Linux 5.4
- CLARiiON CX4-960 with 60 total FC drives of 450 GB @ 15k rpm, 3.5-inch form factor
- 20 fixed LUNs of 2+1R5
- Spread over four DAEs (not counting the base DAE for storage vault drives and hot spares)
- 6 TB of test data stored 50 percent compressed
- 12U SAN storage footprint for the four DAEs holding the test data

The new configuration tested with a VNX5300 included:

- UCS full-width B250 M2 Westmere blade with 8 x 8 Gb virtual HBAs, and two-socket, hex-cores with 196 GB RAM
- Oracle Real Application Cluster Database 11gR2
- Oracle Enterprise Linux 5.5
- VNX5300 with 72 SAS drives of 600 GB @ 10k rpm, 2.5-inch form factor
- 24 fixed LUNs of 2+1R5
- Spread over three DAEs (not counting the base DAE for vault drives and hot spares)
- 6 TB of test data stored 50 percent compressed
- 6U SAN storage footprint for the three DAEs holding the test data

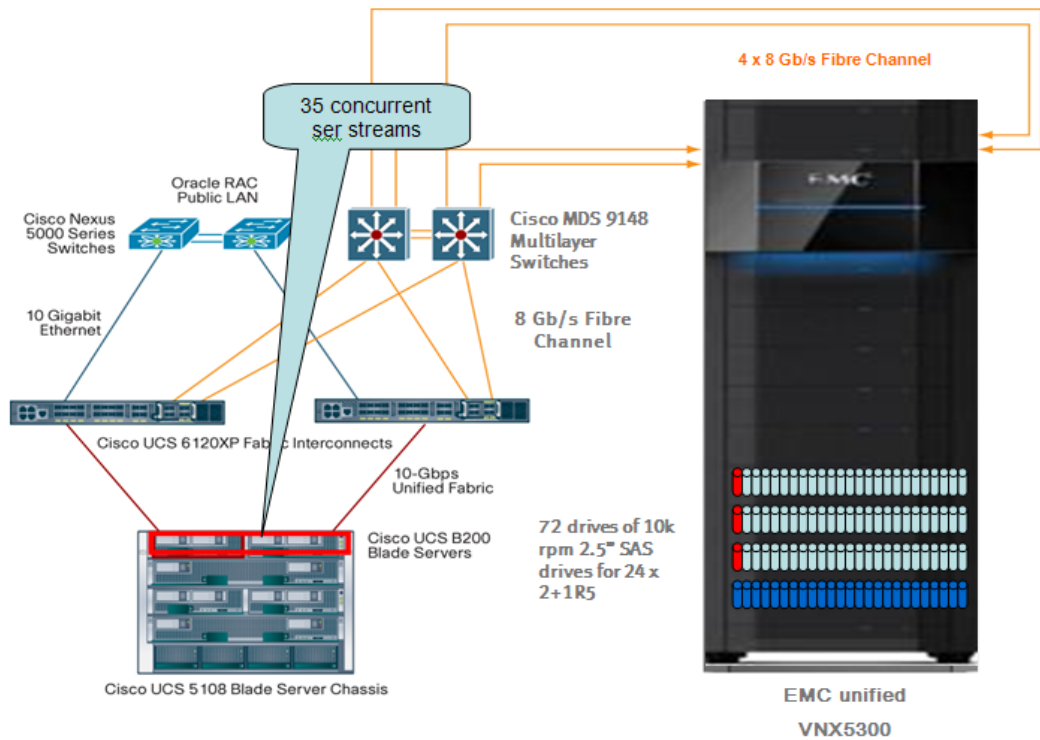


Figure 9. New Reference Architecture: UCS B250 M2 with EMC VNX5300

Comparing the new full-width blade deployment against the previous tested configuration, the new system delivers:

- More than twice the concurrent supported user streams (35 versus 16)
- The ability to leverage fully the added power of the full-width Westmere blade, plus the big memory supported by the blade
- The ability to leverage the new UCS support for 8 Gb FC uplink connections to drive the higher sustainable read bandwidth from the VNX5300
- Four DAEs and 10U of storage in the VNX5300 compared to 19U of storage space for the five DAEs and the CX4-960 SPs and battery
- Lower total system hardware cost

Leveraging the R31.5 FC enhanced read bandwidth feature for VNX5500

The new configuration tested with VNX5500 includes:

- Two UCS full-width B250 M2 Westmere blades with 8 x 8 Gb virtual HBAs, and two-socket, hex-cores with 196 GB RAM
- Oracle Real Application Cluster Database 11gR2
- Oracle Enterprise Linux 5.5
- VNX5500 with 144 SAS drives (600 GB, 10k rpm, 2.5 inch form factor)
- 48 fixed LUNs of 2+1 RAID5

- Spread over six DAEs (not counting the base DAE for vault drives and hot spares)
- 6 TB of test data stored 50 percent compressed
- 12U SAN storage footprint for the three DAEs holding the test data
- VNX5500 configured with four backend SAS buses and dedicating. One of the two expansion I/O modules is dedicated for supporting the additional two backend buses

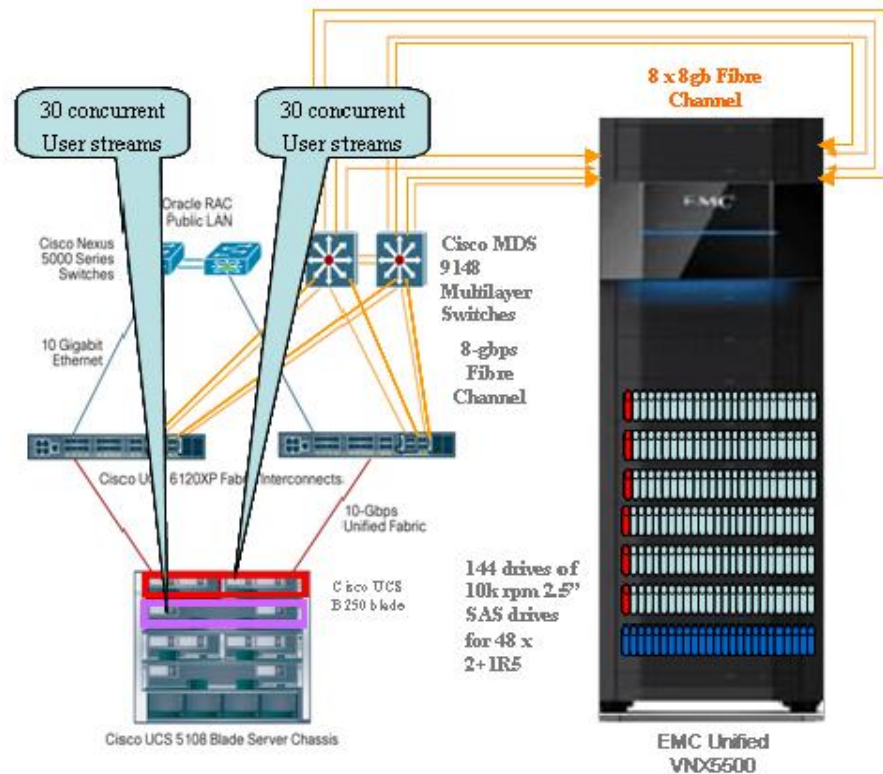


Figure 10. EMC VNX5500 bandwidth enhanced option with two UCS B250 M2

The following are some features comparing this testing to the testing with the VNX5300 configuration:

- Exploit the power of two blades in the UCS chassis to scale up by 71% the concurrent user support level, while scaling up the underlying storage support without adding complexity to manage multiple storage systems.
- Modularly expand the storage from three enclosures holding the key data to six enclosures, to ensure that the expanded data scanning bandwidth achievable through expanded backend bus configuration is fully realized and leveraged.
- A balanced total system configuration that is just as simple to deploy, but targeting a deployment requirement for supporting *more* concurrent users with equal or better user responsiveness.

- Lower total system cost compared to the deployment with multiple UCS blades against multiple VNX5300 storage units so that it is able to contend with growth in concurrent user service support required (frequently the consequence of initial success with many new DW/DSS/BI projects).

Conclusion

Overview

Prominent DBMS vendors in the DW engine marketplace have been engaging with EMC storage platform engineering to make sure we are well informed of the key I/O performance characteristics our new storage products need to be able to support well in order for their software to run optimally on different server/storage hardware combinations. Key themes from the vendors have been:

- More sustained MB/s read/write bandwidth
- More MB/s/drive (to reduce the number of drives needed)
- More density (MB/s bandwidth per storage U form factor achievable)
- Higher usable storage capacity
- Ease of storage configuration, deployment, and calibration for proper server/storage balance
- Lower overall storage system cost

The VNX series, in particular some lower-end products, has been designed, implemented, and tested rigorously focusing on these objectives. The combination of the new VNX storage, matched to some of the newer-generation high-power commodity servers, offers a major deployment platform boost that is urgently needed to contend with the rapid increase in data volume and processing demand for most upcoming, new, strategic DW projects. Few enterprises can afford to delay these projects any more in this fast changing, highly competitive global business environment.

Key findings from the use cases

- For the purpose of driving sustained bandwidth, the lower-end VNX storage systems, especially the VNX5300, offer a very price-performance attractive match against the newer, higher-power multi-core servers offered by the different server vendors. The power and footprint requirements of the new servers, coupled with the VNX5300's focus on leveraging the small form factor, dense drives, enable almost a doubling in user support in half the previous system hardware footprint. This translates to an effective ability of a 4x user workload growth without needing to worry about a data center space upgrade.
- The VNX5300 is at the low end of the VNX family, and would therefore be far more attractive as a new system acquisition to fill the storage needs that once had to be filled with the top-end CX4 family product, without concern about being able to meet the DW business processing I/O bandwidth requirement.
- The VNX5500 with the new FC sustained read bandwidth enhancement in Release 31.5 offers an additional cost-effective option, which balances the increasing data feed rate requirements against the increased server processing capabilities and leads to fewer storage VNX storage units. This offers higher consolidation and reduces space and power requirements without compromising service delivery.

References

EMC documents

For additional information, refer to the following EMC documents:

- [*Introducing EMC Unisphere: A Common Midrange Element Manager*](#)
- [*VNX Family Data Sheet*](#)
- [*VNX Series Specification Sheet*](#)
- [*EMC CLARiiON Best Practices for Performance and Availability: Release 30 Firmware Upgrade*](#)