



Technical Report

## Building a Scalable Data Warehouse Solution

Stephen Daniel, Saad Jafri, Michael Doherty, NetApp  
April 2009 | TR-3760

### **A MODULAR, SCALABLE STORAGE SOLUTION FOR DATA WAREHOUSES**

This paper presents a storage architecture for an Oracle® data warehouse solution. Key architectural features include scalable capacity from 30TB to over 2PB and performance up to 32GB/sec. In addition we present performance results demonstrating an 8GB/sec implementation with near-linear scaling.

## TABLE OF CONTENTS

|          |  |           |
|----------|--|-----------|
| <b>1</b> | <b>THE NEED FOR A MODULAR ARCHITECTURE .....</b> | <b>3</b>  |
| <b>2</b> | <b>SCALABLE ARCHITECTURE .....</b>               | <b>4</b>  |
| 2.1      | <b>MAJOR COMPONENTS .....</b>                    | <b>4</b>  |
| 2.2      | <b>MINIMUM BUILDING BLOCKS .....</b>             | <b>4</b>  |
| 2.3      | <b>PERFORMANCE SCALE-UP .....</b>                | <b>5</b>  |
| 2.4      | <b>CAPACITY SCALE-UP .....</b>                   | <b>5</b>  |
| <b>3</b> | <b>PERFORMANCE RESULTS .....</b>                 | <b>5</b>  |
| 3.1      | <b>SYNTHETIC WORKLOAD .....</b>                  | <b>5</b>  |
| 3.2      | <b>CUSTOMER WORKLOADS .....</b>                  | <b>7</b>  |
| 3.3      | <b>PERFORMANCE ANALYSIS.....</b>                 | <b>8</b>  |
| <b>4</b> | <b>MANAGEABILITY.....</b>                        | <b>8</b>  |
| <b>5</b> | <b>CONCLUSION .....</b>                          | <b>9</b>  |
|          | <b>APPENDIX A: IMPLEMENTATION DETAILS.....</b>   | <b>10</b> |

## 1 THE NEED FOR A MODULAR ARCHITECTURE

During the development of Data ONTAP® 7.3, NetApp engineering made a substantial investment in improving the performance and manageability of our primary storage solutions when used in data warehouse environments. As part of this effort we discussed requirements with many current and prospective data warehouse customers. During these meetings we heard requirements for simplicity, manageability, and a wide range of capacity and performance.

We started the process convinced we needed a modular architecture capable of scaling. We concluded that the architecture required not only scalability but independent scaling of capacity and performance. We needed a simple architecture that could be tuned for the various performance and capacity requirements of the data warehouse market. The final challenge was to develop this architecture in a way that preserves the simplicity, reliability, and manageability of all other solutions delivered by NetApp.

Recently, one of our customers presented us with a challenge—to help his company rearchitect its existing 30TB Oracle/NetApp® data warehouse. The company's goal was to achieve 6GB/sec performance on a database 3 times larger than its current implementation.

This challenge presented us with the need, opportunity, and resources to take our nascent ideas for a scalable data warehouse architecture and put them to the test in a customer-focused proof of concept.

The balance of this paper contains these sections: First, we discuss the architecture and its scaling limits, both upper and lower bounds. Second, we discuss results from our proof-of-concept implementation. Then we provide a brief overview of the manageability features of the architecture. Finally, in an appendix, we present the implementation details so that others may easily and fully reproduce this architecture.

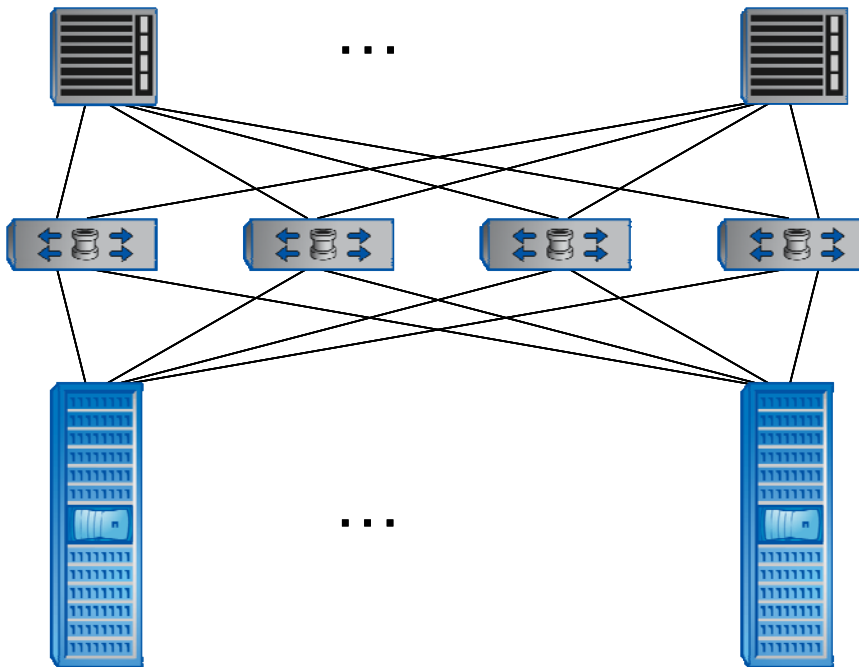


Figure 1) Schematic of a scalable architecture.

## 2 SCALABLE ARCHITECTURE

### 2.1 MAJOR COMPONENTS

To meet our goals for simplicity, scalability, and modularity we chose these building blocks:

#### ORACLE SOFTWARE

We chose to use Oracle11g™ RAC with ASM for our scale-out Oracle data warehouse. Using RAC allows us to scale the compute power of the system in increments. ASM allows straightforward management of data that must be striped across storage pools. Oracle 11g provides features that allow us to optimize the data layout. Oracle10g will also work well, albeit with a slight reduction in performance (see section A.2).

#### SERVERS

We are using midrange commodity servers to run the Oracle Database. In our proof-of-concept lab we choose IBM x3650 servers, each with four sockets, eight total CPU cores, and 32GB of main memory. IBM x-series servers provide excellent price/performance, as demonstrated by numerous TPC publications using IBM x-series equipment.<sup>1</sup>

#### STORAGE SYSTEMS

We chose to build the scalable storage using the NetApp FAS3170 storage system. At the time of this writing the FAS3170 system provides NetApp's best available price/performance for sequential workloads.

#### INTERCONNECT

Based on preliminary analysis we set a goal of sustaining an average of 1GB/sec for each storage controller and each database server. Oracle recommends no more than 200MB/sec/CPU<sup>2</sup>. With eight CPUs per server we are sizing for about 125MB/sec/CPU, which provides some headroom but not so much as to waste Oracle RAC licenses.

Sustaining a target average 1GB/sec rate on a workload that is statistically load-balanced requires an interconnect architecture that can support bursts well in excess of 1GB/sec. For our first implementation of this architecture we chose to use 4Gbps Fibre Channel as the interconnect. We intend to evaluate 10Gbps Ethernet (both NFS and FCoE) in the future.

We chose to use four ports at 4Gbps for a nominal bandwidth to each server of 1.6GB/sec. This provides a comfortable amount of headroom for short bursts in traffic and allows the likelihood that the interconnect will not achieve the nominal full bandwidth.

In order to provide the required data rates, we also connected four ports to each storage controller. For simplicity we structured the storage network as four distinct planes, each implemented as a single Fibre-Channel switch. Each storage controller and each server had a distinct connection to each switch. No interswitch links were required.

Oracle RAC requires that all servers have a view to all storage. Using distinct planes for the FC SAN simplified the switch setup. No zoning or masking was required, since all initiators need to see all targets. Because each plane is isolated there were no extraneous paths to manage. Each server sees four active paths to each LUN, one on each plane. Each server also sees four inactive paths. These paths are not used except in the case of storage controller failure.

### 2.2 MINIMUM BUILDING BLOCKS

The minimum configuration is two servers and one FAS3170A (including two storage controllers). In order to sustain the required average throughput we configured a minimum of three DS14mk4 disk shelves on each storage controller. This minimum configuration is designed to achieve approximately 2GB/sec on sequential reads (table scans).

---

<sup>1</sup> See, for example, [http://www.tpc.org/tpch/results/tpch\\_result\\_detail.asp?id=106100602](http://www.tpc.org/tpch/results/tpch_result_detail.asp?id=106100602).

<sup>2</sup> [http://www.oracle.com/technology/products/bi/db/10g/pdf/twp\\_bi\\_dw\\_build\\_multi\\_tb\\_dw\\_using\\_rac\\_linux\\_0406.pdf](http://www.oracle.com/technology/products/bi/db/10g/pdf/twp_bi_dw_build_multi_tb_dw_using_rac_linux_0406.pdf)

The minimum configuration uses 300GB 15,000 RPM FC disks, providing about 15TB of usable space.

From this minimum 2GB/sec 15TB usable configuration, customers can independently scale up performance and capacity to meet their needs.

### **2.3 PERFORMANCE SCALE-UP**

To scale up performance we added additional servers and FAS3170A systems. Balance is maintained by keeping the number of servers equal to the number of storage controllers. The number of FC switches stays constant at four, but, as the system grows, each switch needs more ports. For example, to build a system with 8 servers and 8 storage controllers, 16-port switches are required.

There are practical upper-level bounds on performance. Beyond 32 servers and 32 controllers, the number of Oracle RAC nodes and the number of FC switch ports both become difficult to manage. Other architectures are possible but they are beyond the scope of this paper. Therefore the current upper limit on performance for this architecture is a nominal 32GB/sec.

### **2.4 CAPACITY SCALE-UP**

The required minimum capacity scales with the number of servers and storage controllers. Each 3170A system provides at least 15TB of usable space.

The first scale-up increment comes from replacing the 300GB drives with 450GB drives. This changes the usable capacity per 3170A system from 15TB to about 23TB.

For systems with a requirement for very high capacity, additional shelves can be added. The number of disk shelves per storage controller should be kept balanced and should be limited to 18 shelves per storage controller. This places an upper limit of about 138TB of usable space per FAS3170A system. However, once the desired number of shelves is above six per storage controller it is possible to consider SATA drives, because the lower performance of these drives is offset by the increasing number. Using large SATA drives will more than double the capacity of each 3170 system. For this reason the maximum capacity of each 3170A system is at least ¼ PB. Scaling up to 16 FAS3170A systems (32 controllers) will provide in excess of 4PB of storage.

## **3 PERFORMANCE RESULTS**

### **3.1 SYNTHETIC WORKLOAD**

We tested this architecture in our proof-of-concept lab using a system built with 4 FAS3170A systems (8 storage controllers), 8 database servers, and 24 shelves of 300GB Fibre-Channel disks. This resulted in about 60TB of usable space and a nominal 8GB/sec read bandwidth.

Our first goal was to demonstrate that we could read and process data with an Oracle query at approximately the nominal system bandwidth limits. To perform this test we built a database with a large table and scanned that table. We chose to use approximately 3TB of data for this table. This value represents a compromise because we'd like to use a table that fills the system, but generating, importing, and managing very large amounts of data represent a logistical challenge.

Both the Oracle and NetApp components require a few nondefault parameter settings to get the system running at full speed. We achieved an average throughput of 7.9GB/sec while scanning our 3TB table. This was extremely close to the system's nominal 8.4GB/sec limit, and analysis of the data confirmed that the statistical fluctuations inherent in parallelizing large queries make it very unlikely that we can improve upon this 7.9GB/sec number.

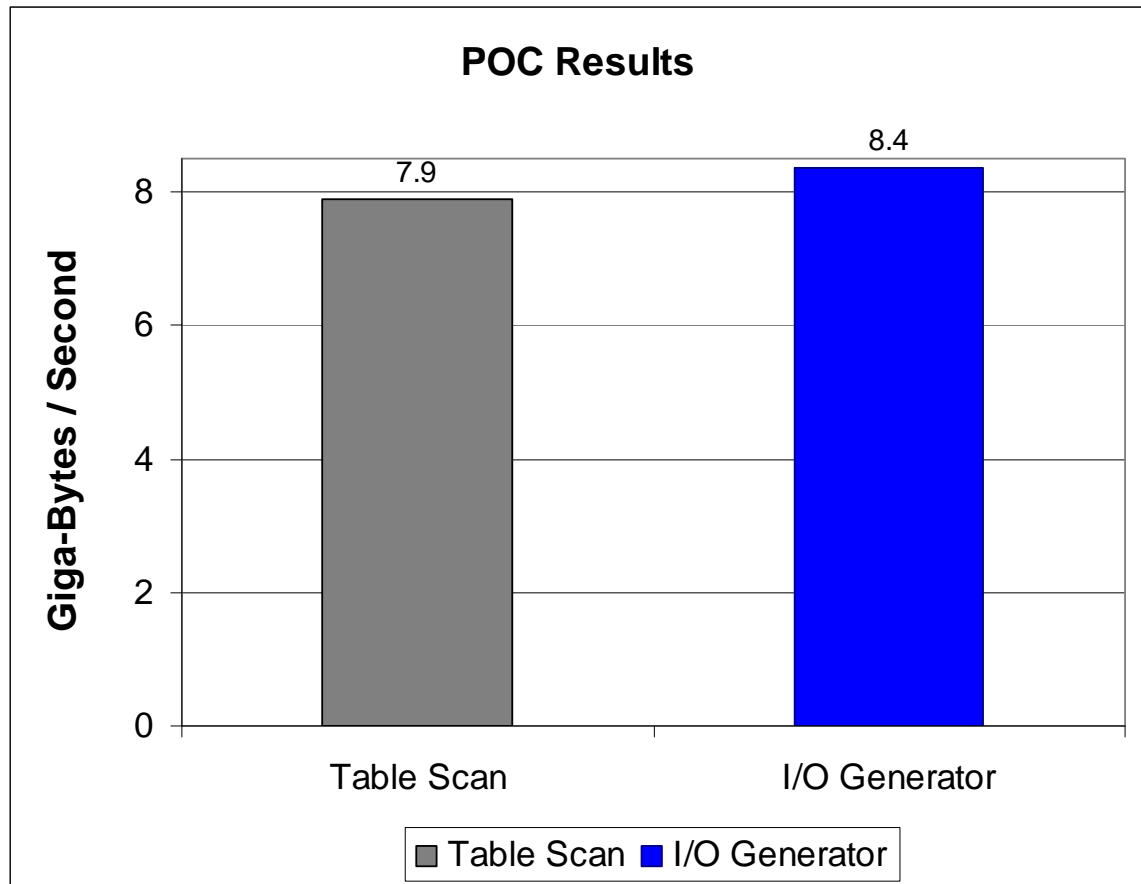


Figure 2) Four-node read performance results.

Our second test attempted to emulate some of the workloads seen in major DSS reporting tools, such as Cognos, Crystal, and Business Objects. The test used multiple input tables, multiple output tables, and multiple concurrent queries. The test database contained about 18TB, and the overall workload was approximately 50% read, 50% write.

Read/write tests are significantly more challenging to the storage system than pure reads. Writing data to disk requires calculating both per-block checksums and per-raid-stripe double parity. Write workloads also require the storage controllers to replicate data to their failover partners. All of these availability features require storage controller bandwidth. For these reasons we expected significantly lower system performance on this test than the pure read test. We were quite pleased to measure an aggregate bandwidth of 5.3GB/sec (reads plus writes), about two-thirds the pure read bandwidth.

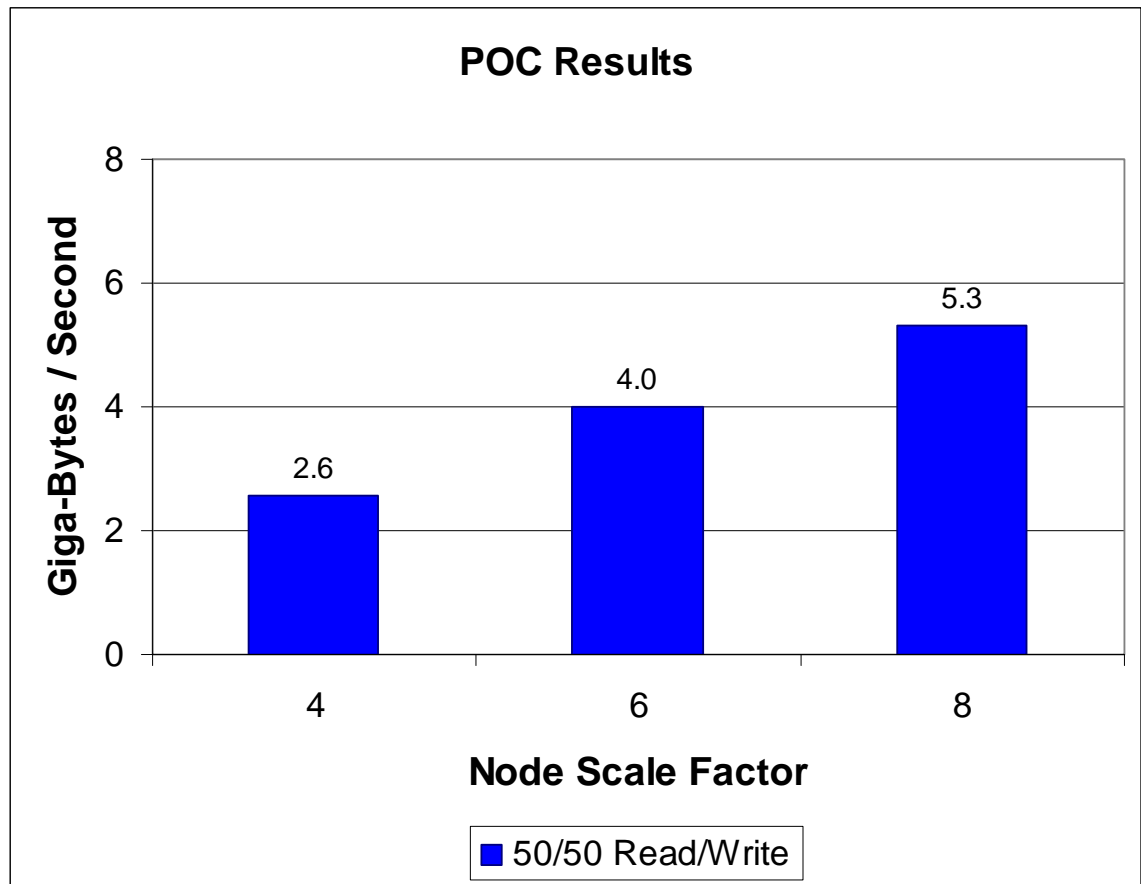


Figure 3) Proof-of-concept performance results.

For our read/write tests we were also concerned about enabling the system to scale appropriately. For these systems we scaled back from eight servers/eight controllers to six/six and four/four. As shown in Figure 3 we observed nearly linear scaling on those tests. Linear scaling shows that the system's performance is limited only by the amount of hardware in use, rather than being limited by the cluster interconnect between the RAC nodes.

### 3.2 CUSTOMER WORKLOADS

In order to validate our architecture we went back to the customer mentioned in Section 1 and invited him into our lab. Our goal was to try some of the company's production workload on the lab system described in section 3.1 and in Appendix A.

A number of practical considerations prevented the customer from installing his entire 30TB data warehouse on our lab system. However, he brought five production queries and a 750GB slice of that warehouse to our lab. In order to avoid testing on a nearly empty system the customer's data was loaded alongside our existing 18TB test database.

Our lab system has approximately four times the raw throughput of the customer's existing implementation. Our goal was to see whether the architecture was scalable enough to provide the expected four-times improvement in performance, even though the customer's production queries are far more complex than the simple pure-read and read-write tests we used initially.

As shown in Figure 4, performance improved on each query by factors ranging from 3.0 to 5.2.

These results provide an excellent proof point showing how scaling can improve performance and cope with growth in large data warehouse environments.

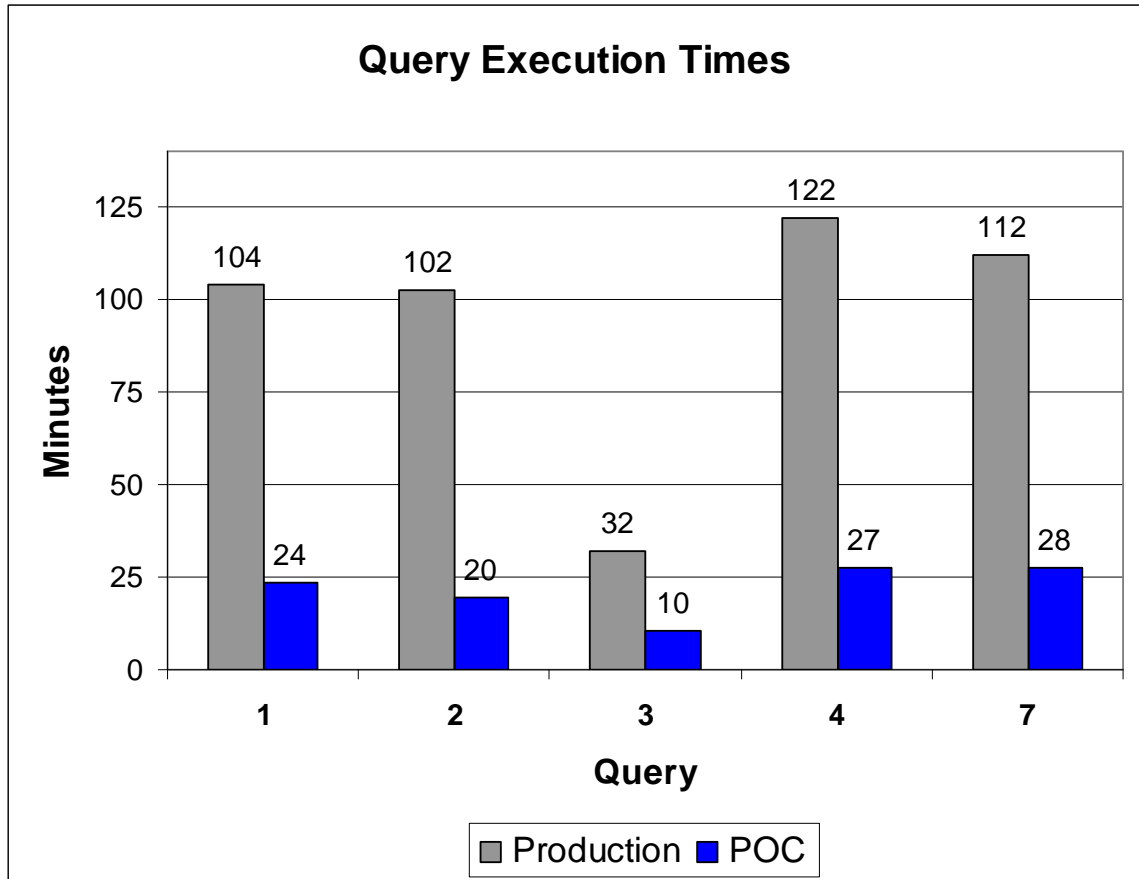


Figure 4) Production query performance results.

### 3.3 PERFORMANCE ANALYSIS

Analysis of the performance data from these systems shows two major findings. First, the key to achieving scalable high performance is to have sufficient parallelism. We found some tuning was required to enable enough parallelism to achieve high performance in our lab queries. (Please see Appendix A for more detail.)

Second, our analysis of the lab queries shows they are I/O bound. We believe this is an appropriate balance, because the complexity of many real-world queries will require a higher ratio of compute to I/O within the database.

## 4 MANAGEABILITY

This paper focuses on the architecture and performance of the scalable data warehouse solution. However, performance was not the only goal of this design. We architected the solution to provide customers with access to the full range of storage and data management products and features offered by NetApp and Oracle. This section provides a few specific examples. Interested customers should refer to the NetApp technical library for more examples of Oracle and NetApp integration.

## **LOAD BALANCING**

The architecture used Oracle ASM to provide load balancing across multiple storage pools. NetApp FlexVol® technology enables the workload to be balanced across each disk within the ASM storage pools.

## **BACKUPS**

Most large-scale data warehouse customers face significant challenges in backup and restore. NetApp's SnapManager® for Oracle creates logical backups of the data warehouse quickly and easily. These backup copies are based on NetApp Snapshot® copies. Creating and maintaining Snapshot copies does not impact the system's performance. Snapshot copies are point-in-time consistent across the storage arrays and consume storage proportional to the block-level difference between the backup copies and the active copy. These logical backups provide a foundation for a number of physical backup systems. They also provide convenient restore points that can protect against application processes that fail or accidentally load invalid data.

## **DISASTER RECOVERY**

Increasingly, customers face a business continuance requirement, even for the largest of data warehouses. Maintaining a consistent, replicated copy of a petabyte-scale data warehouse can be a daunting challenge, even with reasonable recovery point objectives. NetApp SnapMirror® can replicate the consistent backups created by SnapManager for Oracle to remote sites using very economical, continuous block-level differential technology.

## **FLEXCLONE**

NetApp SnapManager for Oracle also provides a convenient and straightforward interface to NetApp FlexClone® technology. FlexClone volumes are writable, logical copies of the database. They consume space only for block differences between the clone and the Snapshot copy of the database. This cost-effective method for managing development and test copies makes possible development and testing scenarios that are not economically feasible for storage systems that require database clones based on full physical copies.

## **5 CONCLUSION**

This paper presents an overview of a modular, scalable architecture for solving a range of data warehouse and decision support problems. A proof-of-concept implementation showed outstanding performance results for this architecture, both on synthetic and customer queries.

## APPENDIX A: IMPLEMENTATION DETAILS

### A.1 PHYSICAL CONSIDERATIONS

The requirement for high performance places significant constraints on how the system components are connected. Each 3170 storage controller has four built-in 4Gbps FC ports. We used those for connectivity to the switches that connected to the database servers. For connectivity to the disk shelves we used four quad-port FCAL cards. The shelves are wired for multipath HA. In this configuration each loop uses two ports and each shelf is on two loops, an active loop and a standby loop. The 12 ports available on these 3 cards are used to create a total of 6 loops, 3 active and 3 standby. The cabling is done in a way that allows the system to continue normal operation in the event of a failure of any quad-port card.

This diagram shows the exact wiring used in the proof-of-concept implementation. This implementation has only one disk shelf per loop. Configurations with higher capacity requirements may have up to six shelves per loop.



## A.2 LOGICAL STORAGE CONFIGURATION

### AGGREGATES, RAID GROUPS, VOLUMES, AND LUNS

For configurations with only three disk shelves per storage controller we recommend:

- One spare disk
- One raid group of 20 disks
- One raid group of 21 disks

For systems built with 300GB disks, both of these raid groups will be placed into a single aggregate. This aggregate will host both /vol/vol0 and a data volume. For systems built with larger disks the raid groups will be in separate aggregates. Very large disks may require more than two aggregates (and more than two raid groups) to work around the current limit of 16TB of data disks in an aggregate.

Higher-capacity systems with more shelves will require more raid groups and aggregates. Our general guidelines for designing the raid groups include:

- Keep one spare disk for systems with three shelves, two spare disks for larger configurations.
- Keep raid groups to a minimum of 16 disks, except when the 16TB limit forces smaller raid groups.
- For systems with three shelves we recommend locating /vol/vol0 in an aggregate with data. For larger systems we recommend a separate two-disk aggregate for /vol/vol0.
- For most implementations we recommend the simplicity of one database volume in each aggregate. More complex environments may use multiple volumes so that data may be separated into performance classes.

Additionally:

- We recommend a minimum of one LUN per volume and a maximum of four LUNs per volume. We prefer 4 LUNs where possible, but we do not recommend making more than 1,024 LUNs visible to the Linux® systems.

## A.3 DATA ONTAP CONSIDERATIONS

### DATA ONTAP VERSION

This architecture requires Data ONTAP version 7.3.1 or later. Customers should consult with NetApp for information on the latest Data ONTAP versions.

### DATA ONTAP CONFIGURATION

We set one nonstandard flag in each storage system's /etc/rc file:

```
setflag wafl_max_write_alloc_blocks 256
```

This flag optimized the WAFL® on-disk data layout. **Note:** Data ONTAP 7.3.2 and later will replace this setflag with an option.

Additionally, the initiator groups (igroups) defined for the LUNs on the storage controllers were set to use Asymmetric Logical Unit Access (ALUA). ALUA is an extension of the Fibre-Channel protocol that defines how multipath I/O should be managed between hosts and storage devices. The standard enables the host to understand the performance difference between the paths when there are multiple paths to a LUN.

## A.4 ORACLE CONSIDERATIONS

### ORACLE VERSION

We recommend Oracle11g or later. Oracle's ASM for 11g allows users to specify the ASM Allocation Unit (AU SIZE). Our testing shows that using an AU SIZE of 64MB provides higher throughput than the default size of 1MB. Oracle10g restricts the ASM AU SIZE to 1MB.

## ORACLE CONFIGURATION

The following was done to set up an ASM disk group with an AU SIZE of 64MB.

1. Created 4 LUNs per NetApp controller (with a total of 32 LUNs for 8 controllers configuration)
2. Assigned an identifier in ASM to these LUNs such as DATA1, DATA2....DATA32
3. Created a single ASM disk group called DATA with an extent size of 64MB (AU\_SIZE) containing all LUNs; the following syntax was used to create the disk group:

```
create diskgroup DATA external redundancy
disk
  'ORCL:DATA1' ,
  'ORCL:DATA2' ,
  'ORCL:DATA3' ,
  'ORCL:DATA4' ,
  'ORCL:DATA5' ,
  'ORCL:DATA6' ,
  'ORCL:DATA7' ,
  'ORCL:DATA8' ,
  'ORCL:DATA9' ,
  'ORCL:DATA10' ,
  'ORCL:DATA11' ,
  'ORCL:DATA12' ,
  'ORCL:DATA13' ,
  'ORCL:DATA14' ,
  'ORCL:DATA15' ,
  'ORCL:DATA16' ,
  'ORCL:DATA17' ,
  'ORCL:DATA18' ,
  'ORCL:DATA19' ,
  'ORCL:DATA20' ,
  'ORCL:DATA21' ,
  'ORCL:DATA22' ,
  'ORCL:DATA23' ,
  'ORCL:DATA24' ,
  'ORCL:DATA25' ,
  'ORCL:DATA26' ,
  'ORCL:DATA27' ,
  'ORCL:DATA28' ,
  'ORCL:DATA29' ,
  'ORCL:DATA30' ,
  'ORCL:DATA31' ,
  'ORCL:DATA32'
ATTRIBUTE 'AU_SIZE'='64M', 'compatible.asm' = '11.1',
'compatible.rdbms' = '11.1';
```

For 50/50 read/write tests, we started with 8 controllers/8 servers, then scaled back to 6/6 and then to 4/4. We used the asm rebalance feature of Oracle ASM when moving from the 8/8 to the 6/6 configuration to relay the data within the disk group. For the 6/6 configuration, 24 LUNs were used. We used asm rebalance again when moving from the 6/6 to the 4/4 configuration. For the 4/4 configuration, 16 LUNs were used.

The following init.ora file parameters were used:

Table scan test:

| Parameter Name                  | Value  |
|---------------------------------|--|
| audit_file_dest                 | /oracle/app/admin/tpch/adump                 |
| audit_trail                     | DB   |
| cluster_database                | TRUE   |
| cluster_database_instances      | 8  |
| Compatible                      | 11.1.0.0.0                                   |
| control_files                   | +DATA/tpch/controlfile/current.256.670179543 |
| db_block_size                   | 16384  |
| db_create_file_dest             | +DATA  |
| db_domain                       |  |
| db_file_multiblock_read_count   | 256  |
| db_name                         | tpch   |
| db_writer_processes             | 4  |
| diagnostic_dest                 | /oracle/app                                  |
| Dispatchers                     | (PROTOCOL=TCP) (SERVICE=tpchXDB)             |
| dml_locks                       | 5000   |
| Filesystemio_options            | SETALL                                       |
| instance_number                 | 1  |
| local_listener                  | LISTENER_TPCH1                               |
| log_buffer                      | 4194304                                      |
| log_checkpoint_timeout          | 1200   |
| log_checkpoints_to_alert        | TRUE   |
| nls_date_format                 | YYYY-MM-DD                                   |
| open_cursors                    | 600  |
| optimizer_index_cost_adj        | 1400   |
| optimizer_mode                  | CHOOSE                                       |
| parallel_execution_message_size | 16384  |
| parallel_max_servers            | 256  |
| parallel_min_percent            | 64   |
| pga_aggregate_target            | 6442450944                                   |
| processes                       | 1000   |
| recovery_parallelism            | 8  |
| remote_listener                 | LISTENERS_TPCH                               |
| remote_login_passwordfile       | EXCLUSIVE                                    |
| replication_dependency_tracking | FALSE  |
| sessions                        | 1105   |
| sga_max_size                    | 3221225472                                   |
| sga_target                      | 3221225472                                   |
| shared_pool_reserved_size       | 67108864                                     |
| shared_pool_size                | 2147483648                                   |
| sort_area_size                  | 349175808                                    |
| spfile                          | +DATA/tpch/spfiletpch.ora                    |
| thread                          | 1  |

|                                   |          |
|-----------------------------------|----------|
| transactions                      | 512      |
| transactions_per_rollback_segment | 20       |
| undo_retention                    | 400000   |
| undo_tablespace                   | UNDOTBS1 |

The 50/50 read/write and customer workload query execution tests were:

| Parameter Name                | Value  |
|-------------------------------|--|
| _pga_max_size                 | 10737418240  |
| _smm_max_size                 | 6250000  |
| aq_tm_processes               | 1  |
| cluster_database              | TRUE   |
| cluster_database_instances    | 8  |
| compatible                    | 11.1.0.7   |
| control_file_record_keep_time | 15   |
| control_files                 | +DATA/controlatdemo01.ctl, +DATA/controlatdemo02.ctl |
| cursor_sharing                | exact  |
| db_32k_cache_size             | 1073741824   |
| db_block_size                 | 8192   |
| db_cache_size                 | 1073741824   |
| db_file_multiblock_read_count | 32   |
| db_files                      | 2000   |
| db_name                       | ATDEMO   |
| db_writer_processes           | 8  |
| diagnostic_dest               | /admin/ATDEMO  |
| dml_locks                     | 25000  |
| fast_start_parallel_rollback  | HIGH   |
| global_names                  | TRUE   |
| instance_name                 | ATDEMO1  |
| instance_number               | 1  |
| job_queue_processes           | 8  |
| log_buffer                    | 102400000  |
| log_checkpoint_interval       | 999999999  |
| log_checkpoints_to_alert      | TRUE   |
| nls_date_format               | dd-mon-rr  |
| open_cursors                  | 3000   |
| open_links                    | 10   |
| optimizer_index_cost_adj      | 70   |
| optimizer_mode                | choose   |
| parallel_max_servers          | 160  |
| parallel_min_servers          | 25   |
| pga_aggregate_target          | 10737418240  |
| pre_page_sga                  | TRUE   |
| processes                     | 1000   |
| query_rewrite_enabled         | true   |

|                                   |             |
|-----------------------------------|-------------|
| query_rewrite_integrity           | trusted     |
| recyclebin                        | OFF         |
| remote_login_passwordfile         | EXCLUSIVE   |
| remote_os_authent                 | TRUE        |
| resource_limit                    | TRUE        |
| resource_manager_plan             | SYSTEM_PLAN |
| session_cached_cursors            | 500         |
| session_max_open_files            | 20          |
| shared_pool_size                  | 1073741824  |
| star_transformation_enabled       | false       |
| statistics_level                  | TYPICAL     |
| streams_pool_size                 | 268435456   |
| thread                            | 1           |
| transactions                      | 400         |
| transactions_per_rollback_segment | 100         |
| undo_management                   | AUTO        |
| undo_retention                    | 2700        |
| undo_tablespace                   | UNDOTBS1    |

#### QUERY PARALLELISM

The following was done to provide enough parallelism for different tests:

Table scan:

1. max\_parallel\_server parameter was set to 256 in the init.ora file.
2. Parallel hint was specified in the select query with a degree of parallelism (DOP) of 256.

50/50 read/write tests:

1. max\_parallel\_server parameter was set to 160 in the init.ora file.
2. Ninety-six queries/sessions per RAC node were launched simultaneously. Therefore, for 8 servers/8 controllers, there were  $96 * 8 = 768$  queries/sessions in progress. For the 6/6 configuration, there were 576 queries/sessions in progress and, for the 4/4 configuration, there were 384 queries/sessions in progress.

The customer workload query execution was:

1. max\_parallel\_server parameter was set to 160 in the init.ora file (same as 50/50 read/write tests).
2. A parallel clause with a degree of 8 was specified in each query.