Netezza Performance Server Health Check Report

Document History

Revision History

Revision	Date	Summary of Changes	Authors
1.0	05/08/2024	Document Creation	Sreekanth Rajan

Approvals

Name	Title

Contents

1. Introduction

This document's main purpose is to provide detailed steps and commands to conduct a health check on a Netezza Performance Server for the customer. Regular health checks help to ensure the system's optimal performance and proactively identify potential issues.

2. System Overview

In this section of System Overview, we will gather details about hardware configuration and the current Netezza software version and ensure that the system is UpToDate with patches and updates.

2.1 Hardware Configuration

We need to extract the details about hardware specifications in the Netezza environment and collect all the hardware-related errors and alerts from pg.log to identify any potential issues in the system.

```
[nz@c54147v1 ~]$ nzhw
Description HW ID Location   Role   State  Security
----------- ----- ---------- ------ ------ --------
Rack        1001  rack1      Active Ok     N/A
SPA         1002  spa1       Active Ok     N/A
SPU         1003  spa1.spu1  Active  Online  N/A
Disk        1004  spa1.disk1 Active Ok     N/A
```

```
[nz@c54147v1 ~]$ nzhw -detail
Description HW ID Location   Role   State  Security Serial number Product
Serial Version Detail
----------- ----- ---------- ------ ------ -------- ------------ --------
------ ------- ------------------------------------------------------------
----
Rack        1001  rack1      Active Ok     N/A
SPA         1002  spa1       Active Ok     N/A
SPU         1003  spa1.spu1  Active Online N/A      vspu-1-1
10.0    8 CPU Cores; 15.65GB Memory; Ip Addr: 127.0.0.1; Designated Spu
Disk        1004  spa1.disk1 Active Ok     N/A      disk-1
BC17    931.51 GiB; Model ST31000640SS ; Non SED;
```

We can use the above command to display information about virtual hardware used by NPS including SPUs virtual machines and virtual disks.

```
[nz@c54147v1 ~]$ nzstats

Field Name           Value
-------------------- -----------------------
Name                 c54147v1.fyre.ibm.com
Description          <sys description>
Contact              <contact name>
Location             <sys location>
IP Addr              10.21.33.132
Up Time              5572 secs
Up Time Text         1 hr, 32 mins, 52 secs
Date                 05-Aug-24, 01:48:06 PDT
State                8
State Text           Online
Model
Serial Num           <serial #>
Num SFIs             0
Num SPAs             1
Num SPUs             1
Num Data Slices      1
Num Hardware Issues  0
Num Dataslice Issues 1
```

The PostgreSQL pg.log file is a log file that contains various messages about the server's operations. These messages include information about connections, queries, errors, and other important events. We can identify potential issues using the pg.log

```
[nz@c54147v1 postgres]$ cat /nz/kit/log/postgres/pg.log | grep -i 'error\|warning'
2024-08-05 00:17:38.505661 PDT [96184]  ERROR:  Attribute 'ELAPSED' not found
2024-08-05 00:19:02.394689 PDT [96231]  ERROR:  relation does not exist
SYSTEM.ADMIN.PG_DATABASE
2024-08-05 00:19:18.616975 PDT [96240]  ERROR:  relation does not exist
SYSTEM.ADMIN.PG_TABLES
2024-08-05 00:19:27.410842 PDT [96246]  ERROR:  Attribute 'ELAPSED' not found
2024-08-05 00:20:57.699786 PDT [96297]  ERROR:  relation does not exist
SYSTEM.ADMIN.PG_STATS
2024-08-05 00:24:36.066440 PDT [96353]  ERROR:  'list
error    ^ found "LIST" (at char 1) expecting a keyword
2024-08-05 00:25:51.525987 PDT [96353]  ERROR:  'SELECT * FROM _v_qrystat
error                        ^ found "SELECT" (at char 26) expecting a keyword
2024-08-05 01:30:05.575591 PDT [98308]  ERROR:  relation does not exist
SYSTEM.ADMIN._V_SYS_RESOURCE
2024-08-05 01:30:34.983737 PDT [98323]  ERROR:  relation does not exist
SYSTEM.ADMIN._V_SYS_RESOURCE
2024-08-05 01:46:37.350450 PDT [98815]  ERROR:  relation does not exist
SYSTEM.ADMIN._V_USERS
```

2.2 Software version

Updating to the latest version of Netezza is crucial for maintaining a secure, efficient, and high-performing data warehouse environment. It ensures that you can take advantage of the latest technological advancements, security enhancements, and performance optimizations, while also benefiting from ongoing support and compatibility with modern systems and tools. The newer versions of Netezza often include optimizations that enhance query performance and overall system efficiency. The updates typically include patches for known vulnerabilities, reducing the risk of security breaches. The new versions address bugs and issues identified in older versions, leading to a more stable and reliable system. Before upgrading we need to ensure compatibility with the latest operating systems and other software dependencies. The latest version of Netezza Performance Server is 11.2.2.4. This version includes several new features and improvements:

- New Features:

1. Querying data from data lakes (AWS S3).
2. Using Kafka as a data source or data sink.
3. Improved backup and restore capabilities, with enablesplitdelete enabled by default.

- Resolved Issues:

1. Fixed issues with table broadcasts on all SPUs, host node failover, and frequent Postgres crashes.
2. Enhanced stability and performance with critical patches.

[nz@c54147v1 ~]$ nzrev
Release 11.3.0.0 [Build 4450]

We can use nzrev to check the installed Netezza build version.

[nz@c54147v1 ~]$ nzrev -V
11.3.0.0-P0-F0-Bld4450

3. System State

It is very important to check system uptime and identify any recent reboots, as it is crucial to ensure that Netezza is running and available.

```
 [nz@c54147v1 ~]$ uptime
 02:22:17 up  2:28,  1 user,  load average: 0.07, 0.02, 0.00

[nz@c54147v1 ~]$ nzstate
System state is 'Online'.

[nz@c54147v1 ~]$ nzsystem showIssues

Spu Partition Issues :
SPU  Partition Id Partition Type Status  Size (GiB) % Used Supporting Disks
---- ------------ -------------- ------- ---------- ------ ----------------
1003 0            Data           Unknown 16         0.00   1004
1003 100          NzLocal        Unknown 868        0.00   1004
1003 101          Swap           Unknown 16         0.00   1004
1003 110          Log            Unknown 1          0.00   1004
```

The above command shows the issues with the system.

```
[nz@c54147v1 ~]$ nzstats

Field Name           Value
-------------------- -----------------------
Name                 c54147v1.fyre.ibm.com
Description          <sys description>
Contact              <contact name>
Location             <sys location>
IP Addr              10.21.33.132
Up Time              8050 secs
Up Time Text         2 hrs, 14 mins, 10 secs
Date                 05-Aug-24, 02:29:25 PDT
State                8
State Text           Online
Model
Serial Num           <serial #>
Num SFIs             0
Num SPAs             1
Num SPUs             1
Num Data Slices      1
Num Hardware Issues  0
Num Dataslice Issues 1
```

This command provides various statistics about the system, including the uptime.

4. System Registry Setting

The nzsystem showRegistry command in Netezza is used to display the current registry settings of the system. The registry contains various configuration settings and parameters that control the behavior and operation of the Netezza system. Running this command provides detailed information about these settings.

```
[nz@c54147v1 ~]$ nzsystem showRegistry
#
# NPS configuration registry
# Date: 05-Aug-24 02:25:41 PDT
# Revision: 11.3.0.0

startup.objCacheFiles              = 50000
startup.maxConnections             = 200
startup.enableAudit                = yes
startup.spuGlobalSharedMemoryMB    = 512

[nz@c54147v1 ~]$ nzsystem showRegistry | grep -i memory
startup.spuSimMemoryMB             = 0
startup.spuGlobalSharedMemoryMB    = 512
startup.spuPMSharedMemoryMB        = 0
startup.spuSharedMemoryHeaps       = 1
sysmgr.memoryErrWarning                  = 1
sysmgr.memoryErrFailover                 = 1
sysmgr.hostMemoryAmountWarning           = 0
sysmgr.hostMemoryAmountFailover          = 0
sysmgr.hostMemoryUsageThresholdToRiseEvent      = -1
sysmgr.vseriesHostMemoryUsageThresholdToRiseEvent = 99
host.snGroupMemoryMinimumPct       = -1
host.snHostMemoryQuota             = 32768
host.snHostMemoryMaxEstimate       = 3000
host.qcMaxLoadMemory               = 2000
system.bladeMemoryMB               = 400
```

Key points:

- **System**: General system settings like file paths for temporary files and databases.
- **Performance**: Settings related to system performance, such as the maximum number of concurrent queries and memory allocation.
- **Networking**: Network-related settings, including IPv6 support and client timeout configurations.
- **Security**: Security settings, such as remote connection permissions and SSL enablement.

5. Performance Metrics

To evaluate Netezza server performance, we should examine factors like CPU and memory usage, identify processes consuming excessive resources, and monitor disk performance using I/O statistics to detect bottlenecks. Additionally, we need to assess available storage space, and usage trends, and analyze network traffic and bandwidth utilization to identify any network issues impacting system performance.

5.1 CPU & Memory Utilization

We need to check current CPU and memory utilization to identify processes which are consuming more resources

```
[nz@c54147v1 ~]$ top -n 1
top - 05:01:24 up  5:07,  2 users,  load average: 0.10, 0.05, 0.01
Tasks: 243 total,   1 running, 242 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.8 us,  0.8 sy,  0.0 ni, 98.5 id,  0.0 wa,  0.0 hi,  0.0 si,
0.0 st
MiB Mem :  15748.2 total,   6856.1 free,    860.2 used,   8031.9
buff/cache
MiB Swap:  16380.0 total,  16380.0 free,      0.0 used.  14293.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
COMMAND
 104656 nz        20   0   54540   4652   3720 R  12.5   0.0   0:00.02 top
      1 root      20   0  238564  11400   8416 S   0.0   0.1   0:03.23
systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.01
kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
rcu_par_gp
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
slub_flushwq
      7 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
kworker/0:0H-events_highpri
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00
mm_percpu_wq
     11 root      20   0       0      0      0 S   0.0   0.0   0:00.00
rcu_tasks_rude_

[nz@c54147v1 ~]$ free -m
               total        used        free      shared  buff/cache
available
Mem:           15748         872        6843         264        8032
14280
Swap:          16379           0       16379
```

5.2 Disk Performance Analysis

In this section, we will conduct a comprehensive analysis of disk performance. The process will include the following steps:

1. Monitor Disk I/O Statistics:
    - Track key metrics such as read/write speeds, latency, and queue length.
    - Use tools like iostat, vmstat, or dstat to gather detailed statistics.
    - Identify patterns or anomalies that could indicate performance issues or inefficiencies.
2. Identify Bottlenecks:
    - Analyze the gathered data to pinpoint areas where disk performance is suboptimal.
    - Look for signs of high latency or consistently high queue lengths, which may suggest bottlenecks.
3. Evaluate Storage Space and Usage Trends:
    - Assess the current storage capacity and usage patterns.
    - Use tools like df and du to evaluate available storage and identify trends over time.
4. Plan for Capacity Expansion:
    - Based on the analysis of usage trends, determine if there is a need for additional disk space.
    - Develop a strategy to increase disk capacity as needed to ensure optimal performance and prevent future bottlenecks.

This analysis will help optimize disk performance, ensuring the system runs efficiently and is prepared for future storage demands.

```
[nz@e1n1-npshost ~]$ iostat -x 1 5
Linux 3.10.0-1160.53.1.el7.x86_64 (e1n1)        08/05/2024      _x86_64_
(32 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           6.35    0.06    3.62    0.31    0.00   89.66

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await r_await w_await  svctm  %util
nvme0n1          93.58     0.00    4.68    5.93   566.54   706.38   239.78
0.00    0.17    0.22    0.13   0.50   0.53
nvme1n1          93.58     0.00    4.65    5.83   566.14   718.82   245.33
0.00    0.17    0.22    0.13   0.50   0.53
nvme3n1          93.58     0.00    4.65    7.34   566.40   757.02   220.68
0.00    0.19    0.30    0.12   0.55   0.66
nvme2n1          93.58     0.00    4.65    6.00   566.29   721.81   241.83
0.00    0.17    0.23    0.13   0.52   0.56
sda               0.42    17.67    1.08  123.92    39.44   695.29    11.76
0.10    0.84    0.77    0.84   0.03   0.39
```

```
md0               0.00     0.00    0.00    0.00     0.00     0.00    31.51
0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00    0.00     0.00     0.00    35.74
0.00    0.15    0.14    2.05   0.07   0.00

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.94    0.00    1.63    0.09    0.00   95.34

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await r_await w_await  svctm  %util
nvme0n1           0.00     0.00   20.00    0.00   980.00     0.00    98.00
0.00    0.15    0.15    0.00   0.25   0.50
nvme1n1           0.00     0.00   20.00    1.00   980.00   252.00   117.33
0.00    0.14    0.15    0.00   0.24   0.50
nvme3n1           0.00     0.00   20.00    3.00   980.00    12.00    86.26
0.00    0.09    0.10    0.00   0.35   0.80
nvme2n1           0.00     0.00   20.00    2.00   980.00   256.00   112.36
0.00    0.18    0.20    0.00   0.32   0.70
sda               0.00     0.00   38.00    6.00  1773.00    28.00    81.86
0.01    0.32    0.37    0.00   0.32   1.40
md0               0.00     0.00    0.00    0.00     0.00     0.00     0.00
0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00
0.00    0.00    0.00    0.00   0.00   0.00



avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           5.04    0.00    2.07    0.00    0.00   92.90

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz
avgqu-sz   await r_await w_await  svctm  %util
nvme0n1           0.00     0.00    0.00    5.00     0.00    92.00    36.80
0.00    0.00    0.00    0.00   0.80   0.40
nvme1n1           0.00     0.00    0.00    6.00     0.00   272.00    90.67
0.00    0.17    0.00    0.17   0.50   0.30
nvme3n1           0.00     0.00    0.00   13.00     0.00   392.00    60.31
0.00    0.00    0.00    0.00   0.38   0.50
nvme2n1           0.00     0.00    0.00    1.00     0.00     4.00     8.00
0.00    0.00    0.00    0.00   1.00   0.10
sda               0.00    76.00    0.00  750.00     0.00  3536.00     9.43
1.77    2.35    0.00    2.35   0.02   1.80
md0               0.00     0.00    0.00    0.00     0.00     0.00     0.00
0.00    0.00    0.00    0.00   0.00   0.00
dm-0              0.00     0.00    0.00    0.00     0.00     0.00     0.00
0.00    0.00    0.00    0.00   0.00   0.00
```

```
[nz@e1n1-npshost ~]$ df -h
Filesystem                                    Size  Used Avail Use% Mounted
on
overlay                                        98G  6.7G   91G   7% /
tmpfs                                          95G     0   95G   0% /dev
ips                                           5.9T  3.1T  2.9T  53% /nz
/dev/sda2                                      98G  6.7G   91G   7%
/etc/hosts
/dev/sda9                                     195G  108G   77G  59%
/etc/resolv.conf.upstream
tmpfs                                          95G  3.7M   95G   1%
/host/run
tmpfs                                          19G     0   19G   0%
/host/run/user/0
shm                                           152G  551M  152G   1%
/dev/shm
/dev/sda5                                      38G   20G   17G  55%
/var/lib/sedsupport
tmpfs                                          64M   13M   52M  19% /run
tmpfs                                          64M     0   64M   0%
/run/lock
tmpfs                                          64M     0   64M   0%
/var/log/journal
tmpfs                                          71G  2.9G   68G   5% /tmp
10.231.121.246:/nfs_kbbnzbackup06_backup       79T   38T   42T  48%
/mnt/ipsbackups
10.231.121.246:/nas_kb_mx_sd_bo_edw_archive    80T   68T   13T  85%
/mnt/edw_archive
```

5.3 Network Usage Analysis

In this section, we will perform a detailed analysis of network usage to ensure optimal performance. The process will include the following steps:

1. **Analyze Network Traffic**:
    - Monitor network traffic patterns using tools such as ifstat, tcpdump, or ntop.
    - Collect data on packet flow, connection types, and traffic sources/destinations.
    - Identify any unusual or excessive traffic that may indicate network congestion or security issues.
2. **Evaluate Bandwidth Utilization**:
    - Measure current bandwidth usage to determine if the network is operating within its capacity.
    - Use tools like NetFlow, SNMP, or bmon to gain insights into bandwidth allocation and consumption.
    - Identify peak usage times and evaluate if additional bandwidth is required to accommodate demand.

3. **Identify Network-Related Issues**:
    - Analyze data for signs of latency, packet loss, or jitter, which can affect network performance.
    - Investigate any recurring issues, such as slow connection speeds or frequent disconnects, to determine root causes.

By thoroughly analyzing network traffic and bandwidth utilization, we can identify potential issues affecting performance and implement solutions to enhance network efficiency and reliability.

```
[nz@e1n1-npshost ~]$ ifstat -a
#kernel
Interface        RX Pkts/Rate    TX Pkts/Rate    RX Data/Rate    TX
Data/Rate
                 RX Errs/Drop    TX Errs/Drop    RX Over/Rate    TX
Coll/Rate
lo                 2855M 0         2855M 0         4242M 0         4242M 0
                       0 0             0 0             0 0             0 0
mgt0               2468M 0         2212M 0       494728K 0         2506M 0
                       0 0             0 0             0 0             0 0
mgt1              47466K 0         3816K 0         1960M 0         3960M 0
                       0 20889         0 0             0 0             0 0
fab0               1024M 0         3794M 0         1249M 0         2346M 0
                       0 2            31 0             0 0             0 0
fab1               2521M 0       703378K 0         3036M 0       923449K 0
                       0 2             0 0             0 0             0 0
mgt-br0            2278M 0         2155M 0         3069M 0         2950M 0
                       0 0             0 0             0 0             0 0
fbond              3546M 0       203133K 0       795332K 0         3269M 0
                       0 8            31 0             0 0             0 0



fbond.4079       284174K 0         2620M 0         2397M 0         1475M 0
                       0 0             0 0             0 0             0 0
fab-br0            2492M 0       202769K 0         4034M 0         1153M 0
                       0 1141          0 0             0 0             0 0
docker0           39387K 0        40658K 0         3299M 0         1269M 0
                       0 0             0 0             0 0             0 0
fbond.4080       344017K 0         3107M 0         2400M 0         1925M 0
                       0 9593K         0 0             0 0             0 0
fbond.3121         4089M 0         2739M 0         2669M 0         1471M 0
                       0 22696K        0 0             0 0             0 0
veth9aba08d       10765K 0        15606K 0         1618M 0         1820M 0
                       0 0             0 0             0 0             0 0
usb0             220305 0             0 0        30784K 0             0 0
                       0 0             0 0             0 0             0 0
veth13a0067        1037K 0        969879 0       105805K 0       138253K 0
                       0 0             0 0             0 0             0 0
```

5.4 Database Health Check for Netezza Systems

Database health checks are crucial for maintaining the performance and efficiency of Netezza systems. A thorough health check involves analysing various aspects of the database to ensure optimal performance and prevent potential issues. Below are key considerations and steps to perform a comprehensive database health check:

1. Analyze Database Size and Growth Patterns

- **Monitor Database Size**: Regularly check the overall database size to understand its current state and anticipate future storage needs.
- **Growth Trends**: Analyze historical data to identify growth patterns. This helps in forecasting storage requirements and planning for capacity upgrades.
- **Capacity Planning**: Ensure the system has sufficient resources to accommodate anticipated growth without performance degradation.

2. Identify Large Tables and Indexes

- **Large Tables**: Identify tables that are disproportionately large and could impact performance. These tables might require special attention or optimization.
- **Index Management**: Locate large or unused indexes that might be consuming unnecessary resources. Review and optimize index usage to improve query performance.

3. Analyze Query Execution Time

- **Monitor Query Performance**: Identify slow-running queries that could be impacting overall system performance. Regular monitoring helps in pinpointing queries that need optimization.
- **Explain Plans**: Use explain plans to gain insights into query execution paths and identify bottlenecks. This tool is essential for diagnosing performance-related issues and making informed optimization decisions.

4. Check Table and Index Statistics

- **Statistics Collection**: Ensure that statistics for tables and indexes are up-to-date. Accurate statistics are vital for the query optimizer to make efficient execution plans.
- **Regular Updates**: Implement regular schedules for updating statistics, especially after large data changes, to maintain query performance.

5. Perform Regular Maintenance Tasks

- **Grooming Tasks**: Regularly perform grooming tasks to reclaim space from deleted or outdated data. This includes tasks like table reorganization and index rebuilding.
- **Maintenance Schedule**: Establish a routine maintenance schedule to ensure all tasks are performed consistently, preventing performance degradation over time.

6. Netezza Backup Strategy

A comprehensive Netezza backup strategy is crucial for ensuring data integrity, availability, and disaster recovery. As part of a health check report, the backup strategy should include regular backups, monitoring, and verification processes.

The **nzbackup** command in Netezza is used to back up the database. The **-history** option is particularly useful for viewing the history of backup operations. This includes details such as the start and end times of each backup, the status of the backup, and the size of the backup files.

By using the **nzbackup -history** command, you can keep track of your backup operations and ensure that your data is consistently protected

[nz@e1n1-npshost ~]$ nzbackup -history

DBNAME 20240724172935 6 DIFF COMPLETED 2024-07-29 19:49:50 backupsvr.149652.2024-07-29.log
DBNAME 20240731173148 1 FULL COMPLETED 2024-07-31 10:31:48 backupsvr.156836.2024-07-31.log
DBNAME 20240731173148 2 DIFF COMPLETED 2024-07-31 19:36:34 backupsvr.150034.2024-07-31.log
DBNAME 20240731173148 3 DIFF COMPLETED 2024-08-01 19:33:29 backupsvr.8330.2024-08-01.log
DBNAME 20240731173148 4 DIFF COMPLETED 2024-08-02 19:46:23 backupsvr.60052.2024-08-02.log
DBNAME 20240731173148 5 DIFF COMPLETED 2024-08-04 19:42:11 backupsvr.101244.2024-08-04.log
DBNAME 20240731173148 6 DIFF COMPLETED 2024-08-05 19:50:11 backupsvr.140180.2024-08-05.log

- **Completed**: Indicates that the backup was successful.
- **Failed**: Indicates that the backup was not completed successfully. Further investigation into logs or error messages would be needed to determine the cause of failure.

1. Backup Types

Netezza supports several types of backups:

- **Full Backups**: Capture the entire database.
- **Incremental Backups**: Capture only the changes since the last backup.
- **Differential Backups**: Capture the changes since the last full backup.

2. Backup Schedule

Establish a regular backup schedule:

- **Daily Backups**: Perform incremental or differential backups daily to ensure that recent changes are captured.
- **Weekly Full Backups**: Perform full backups weekly to ensure you have a complete snapshot of your database.

3. Backup Retention Policy

Define how long you will keep each type of backup:

- **Daily Incremental/Differential Backups**: Retain for one week.
- **Weekly Full Backups**: Retain for one month.
- **Monthly Backups**: Retain for one year or as required by business needs.
- The following backups are recommended every week:
    - **nzhostbackup** (requires a system Pause)
    - **nzbackup -globals -dir /path**

7. Optimize Query Performance

This section discusses optimizing Netezza query performance, which involves steps and best practices to ensure that queries run efficiently and use system resources best. Here's a comprehensive approach to analyze and optimize Netezza query performance:

- Check active queries using _v_qrystat to identify long-running or resource-intensive queries.

```
[nz@c54147v1 ~]$ nzsql -c " select * from _v_qrystat "
 QS_SESSIONID | QS_PLANID | QS_CLIENTID | QS_CLIIPADDR | QS_SQL |
QS_STATE | QS_TSUBMIT | QS_TSTART | QS_PRIORITY | QS_PRITXT |
QS_ESTCOST | QS_ESTDISK | QS_
ESTMEM | QS_SNIPPETS | QS_CURSNIPT | QS_RESROWS | QS_RESBYTES
-------------+-----------+------------+-------------+--------+----------+------------+-----------+------
-------+-----------+------------+------------+----
-------+------------+------------+------------+------------
(0 rows)
```

- Use _v_qryhist to review past query performance and identify recurring issues.

```
[nz@c54147v1 ~]$ nzsql -c " select * from _v_qryhist "
 QH_SESSIONID | QH_PLANID | QH_CLIENTID | QH_CLIIPADDR | QH_DATABASE |
QH_USER |                                        QH_SQL
                          |    QH_TSUBMIT     |     QH_TSTART     |      QH_TEND      |
QH_PRIORITY | QH_PRITXT | QH_ESTCOST | QH_ESTDISK | QH_ES
TMEM | QH_SNIPPETS | QH_SNPTSDONE | QH_RESROWS | QH_RESBYTES |
QH_CLIENT_USER_ID | QH_CLIENT_APPLICATION_NAME |
QH_CLIENT_WORKSTATION_NAME | QH_CLIENT_ACCOUN
TING_STRING
-------------+-----------+------------+-------------+------------+---------+--------------------------


       62331 |         1 |          6 | 10.21.33.132 | SYSTEM      | ADMIN   | SELECT datasliceid,
COUNT(*) as rows_per_slice FROM sales GROUP BY dataslice
id ORDER BY rows_per_slice DESC | 2024-08-06 05:35:37 | 2024-08-06 05:35:37 | 2024-
08-06 05:35:45 |          3 | normal    |         0 |         1 |
   1 |         1 |          1 |         0 |         0 |               |                       |                       |
       62331 |         2 |          6 | 10.21.33.132 | SYSTEM      | ADMIN   | INSERT INTO sales
(sale_id, customer_id, sale_date, amount) VALUES (1, 101,
'2024-08-01', 150.75)          | 2024-08-06 05:36:51 | 2024-08-06 05:36:51 | 2024-08-06
05:36:52 |          3 | normal    |         0 |         0 |
```

- Use EXPLAIN to review the query execution plan.

```
SYSTEM.ADMIN(ADMIN)=> EXPLAIN SELECT * FROM sales WHERE sale_date >
'2023-01-01' ;
NOTICE:  QUERY PLAN:

QUERY PLANTEXT:

Sequential Scan table "SALES" (cost=0.0..0.0 rows=5 width=20 conf=80)
```

- Ensure tables are evenly distributed across SPUs to avoid data skew.

```
SYSTEM.ADMIN(ADMIN)=> SELECT datasliceid, COUNT(*) as rows_per_slice FROM
sales GROUP BY datasliceid ORDER BY rows_per_slice DESC ;
 DATASLICEID | ROWS_PER_SLICE
-------------+----------------
           1 |              5
```

7.1 Zone Maps for Query Performance

Zone maps in Netezza are an essential feature for query performance optimization, particularly in large-scale databases. They help to minimize disk I/O by allowing the database engine to skip over large sections of data that are irrelevant to the query, thus speeding up query execution times. Zone maps are metadata structures that store the minimum and maximum values for data stored in each extent of a table. When a query is executed, Netezza uses these zone maps to determine which extents need to be scanned and which can be skipped, based on the query predicates. Zone maps are automatically created and maintained by Netezza when data is loaded into a table

Best practices for effective utilization of Zone map

- We need to choose the Right Distribution Key to distribute data evenly across all data slices. A poorly chosen distribution key can lead to data skew, reducing the effectiveness of zone maps.
- For very large tables, consider partitioning them by commonly filtered columns.
- If possible, avoid applying functions to columns in the WHERE clause that benefit from zone maps, as this can prevent the zone map from being used.
- Periodically groom your tables to reorganize data and optimize the effectiveness of zone maps.
- To verify all the columns on which Netezza can create zone maps, you can use the nz_zonemap utility from the SQL toolkit. This toolkit includes various useful scripts and commands to assist in optimizing Netezza performance, including identifying columns that are candidates for zone maps.
- Once the SQL toolkit is installed, you can use the nz_zonemap script to list the columns that can have zone maps created on them

nz_zonemap DATABASENAME TABLENAME Distributed Column name

Key Points:

- Prefer hash joins for large datasets and nested loop joins for smaller ones.
- Ensure that join keys are properly indexed and distributed.
- Regularly groom tables to reclaim space from deleted rows and reorganize data.
- Choose a column that evenly distributes data across all SPUs
- Choose columns that are frequently used in large table joins to get collocated joins
- Use the same data type for the join columns
- Choose a single column – Avoid multi-column distributions if possible
- Choose RANDOM distribution for small dimensions and lookup tables only
- When unloading large data sets use EXTERNAL TABLES instead of nzsql
- Regularly check for data skew and redistribute tables as necessary.
- Running statistics in Netezza helps optimize query performance by updating the metadata about the tables and columns, such as the number of rows, distinct values, and data distribution

  [nz@c54147v1 ~]$ nzsql -c " GENERATE STATISTICS ON sales; "
  GENERATE STATISTICS

- Regularly update statistics, especially after significant data changes (e.g., bulk inserts, updates, or deletes).