Multipath Load Balancing Recommendations

Technical Brief

Abstract

This technical brief provides recommendations for managing latency issues in a fabric using the path selection parameter available in most operating system environments. The path selection function determines the mechanism used by the OS to optimize the I/O path selection for multipath I/O drivers.

Table of Contents

Chapter 1: Introduction

In this technical brief, we discuss how to determine the best “path selection” algorithm for your multipath driver based on your particular operating system environment. Our goal is to provide the information that you need to configure your multipath driver to get the most operational efficiency while maintaining a high level of resiliency.

The reason you are investing in a multipath driver for your environment is to obtain a high level of I/O resiliency for your Fibre Channel SAN environments while optimizing the I/O flow of your applications. The multipath driver does this by utilizing the available paths to your I/O devices using the path selection algorithm configured for the driver. This configuration selection defines how the driver distributes the I/O flow over the multiple physical paths that you may have to your I/O devices. It also defines how the driver selects the most effective path to maximize I/O throughput and resiliency.

We start with our recommendations, and then we examine the details of our investigation supporting these recommendations, followed by our references for each of the operating system environments. Note, these recommendations address latency issues due to congestion in your environment, and Broadcom is working with the Fibre Channel community to improve the response mechanisms for the multipath driver when encountering marginal link issues.

Chapter 2: Recommendations

Broadcom recommends the following configuration options for each operating system environment to improve both resiliency and performance.

Operating System[a]	Options	Default	Broadcom Recommendation
AIX	Round Robin Shortest Queue Failover	Round Robin	Shortest Queue
Linux (Red Hat, SUSE)	Round Robin Queue Length Service Time	Round Robin	Queue Length
Windows	Round Robin Least Queue Depth Weighted Paths	Round Robin	Least Queue Depth
VMware	Round Robin Most Recently Used Fixed	Fixed (active-active) Most Recently Used (active-passive)	Round Robin

a. The minimum OS levels supporting the improved path selection algorithms are AIX 7.0, Linux Red Hat 6.0, Linux SUSE 11.0, Windows 2012, and VMware 6.7.

These recommendations specifically address issues associated with congestion that introduce higher latencies on one path compared to other available paths. The above recommendations indicate the specific algorithm that responded most efficiently to congestion conditions for each operating system environment.

Although this technical brief is focused on recommendations for dealing with latency due to congestion, Broadcom is committed to constantly improving the user experience of Fibre Channel deployments. Toward that end, we are working with the Fibre Channel ecosystem to address persistent, intermittent errors due to marginal cables or SFPs (for example, CRC and other link errors, also known as “sick but not dead” issues). You can help us pursue these improvements by contacting your multipath vendor any time the system fails to recover from single path failures when multipath solutions are deployed.

2.1 Observations

Our investigation determined that multipath driver behavior varies considerably between implementations due to the varying degrees of configuration capabilities. The effect on resiliency and performance differs vastly based on the configuration of the “path selection” algorithm. Although the common default of “round-robin” sufficiently addressed performance in an optimal environment, the resiliency characteristics suffer when latency due to congestion is present.

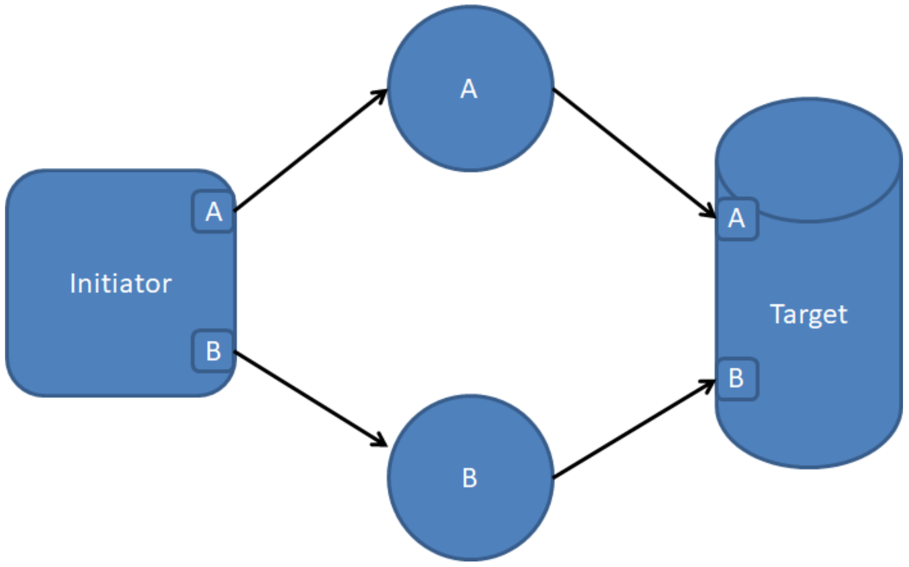
We also observed that the documentation provided by storage vendors often recommends the default setting as the best practice path selection algorithm; however, we found that the default is often the round-robin option, which was found to be insufficiently resilient. It appears that this recommendation generally predated the addition of the more advanced and resilient path selection algorithms such as queue-depth or service-time. Therefore, it is recommended that users consult with their storage vendors regarding these more advanced algorithms and the associated improvements in resiliency provided by the newer technology.

Note that each environment is unique and users should perform basic performance and resiliency tests particular to their environment to determine the most optimal setting. Please contact a local Broadcom representative for further assistance or more information.

Chapter 3: Detailed Description

We now look at the details of our investigation, which support the recommendations given above.

The effectiveness of the multipath driver is dependent upon the stability and resiliency of the physical environment, and, therefore, the selection of the most appropriate path selection algorithm is based on these characteristics as well. In the following diagram, the path between the initiator (host) and the target (storage) consists of two paths: path A and path B.



The multipath driver utilizes the path selection algorithm to optimize the use of both the A and B paths based on the resiliency and load-balancing characteristics of the algorithm. Each operating system environment has an environmental setting associated with the storage device that determines which of the path selection algorithms is used.

3.1 General Algorithms

The general path selection algorithms are common across operating system environments and are designed to address resiliency or performance.

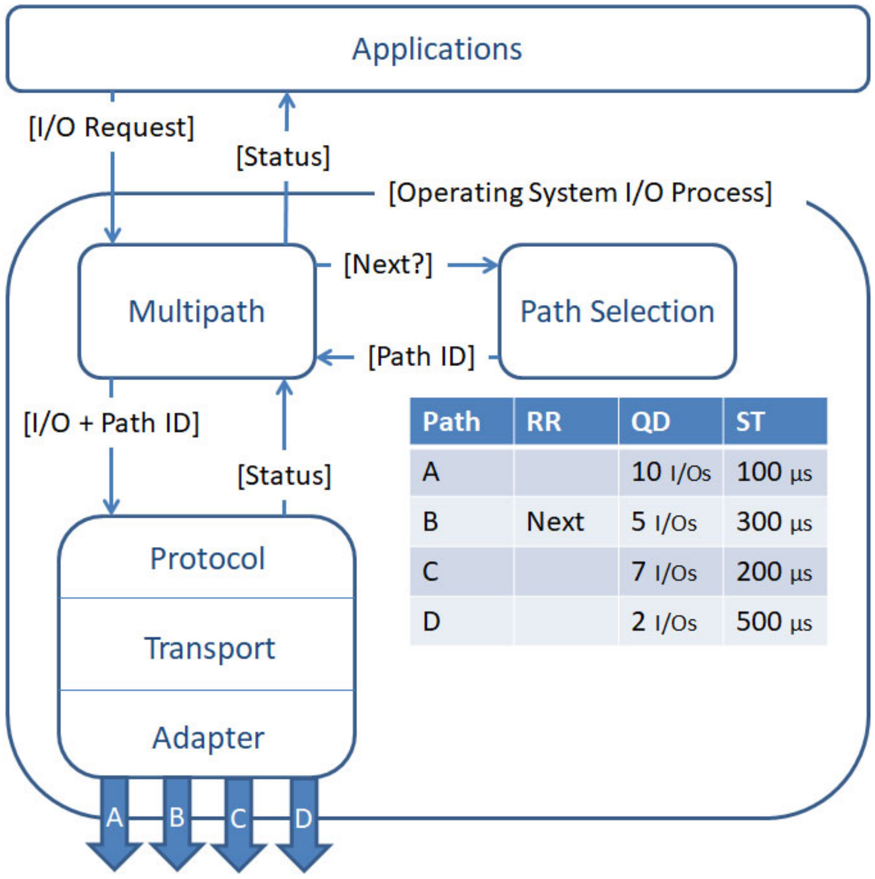
The most basic algorithm addresses resiliency and uses the available paths for failure recovery. In this case, a primary path is selected, and all I/O is sent along that path until a failure is detected and one of the secondary paths becomes the primary. For example, in the diagram above, path A is used as the primary path until an error is detected. When this occurs, path A is marked as failed, and path B becomes the new primary path. This approach is referred to as “failover” and does not address performance characteristics.

The more advanced algorithms not only address the resiliency aspect of multipath solutions, but also attempt to utilize all available paths to obtain high levels of performance. These algorithms differ largely in the techniques for maintaining performance. The most rudimentary version of the algorithms (and the most common) is load balancing by iterating through the available paths. This technique simply sends each new I/O request on the next path in the list of available paths (often referred to as “round-robin”). The more advanced versions of the algorithms perform load balancing based on the operational characteristics of the paths. New I/O requests are assigned to the path with the most optimal operational characteristics based on queue depth, quantity of pending I/O, response time, and so on.

In most implementations today, multipath driver solutions default to the rudimentary performance-based load-balancing algorithm (round-robin) rather than the basic resiliency solution (failover). However, Broadcom’s research suggests that considerable improvements in both resiliency and performance can be achieved by selecting one of the more advanced path selection algorithms.

3.2 Algorithm Comparison

In order to understand the differences between the available path selection algorithms, the following abstraction of a generic multipath I/O process is used:[1]



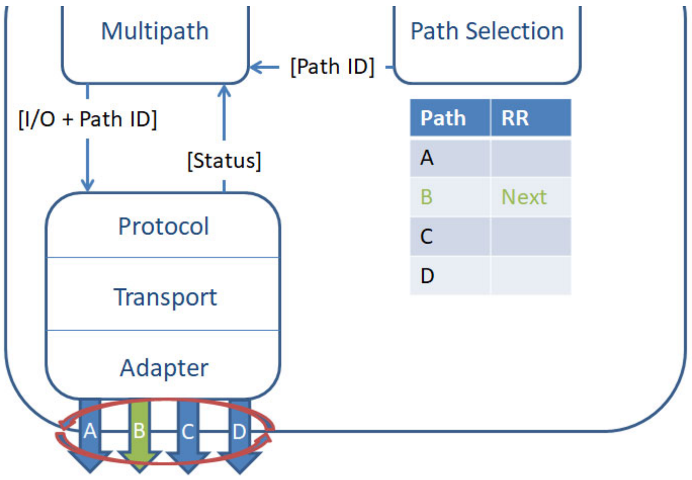
Path	RR	QD	ST
A		10 I/Os	100 µs
B	Next	5 I/Os	300 µs
C		7 I/Os	200 µs
D		2 I/Os	500 µs

In this diagram, the primary functions are highlighted beginning with the application driving the I/O requests, which are handled by the operating system I/O process. The I/O process contains the multipath driver, the path selection function, and the device access layers (that is, protocol, transport, and adapter drivers). In the diagrams that follow, each of the path selection algorithms is shown using variations of this diagram to highlight the specific behavior of the algorithms.

1. This abstraction is intended to highlight the general concepts associated with the path selection process and is not intended to reflect the precise implementation of a file/block I/O system. For example, in Linux, the operating system I/O process provides a block interface to the file system/block application, and the I/O requests are queued across the various layers of the multipath driver: protocol, transport, and adapter. Consequently, the depictions of the queues in these diagrams are intended to emphasize the behavior of the path selection algorithm rather than the specific implementation location/ residency of the queues.

The round-robin (RR) algorithm simply iterates through each of the available paths by returning the next path in the list when the multipath I/O driver requests the next path (first A is returned, then B, then C, then D, then A, and so on).

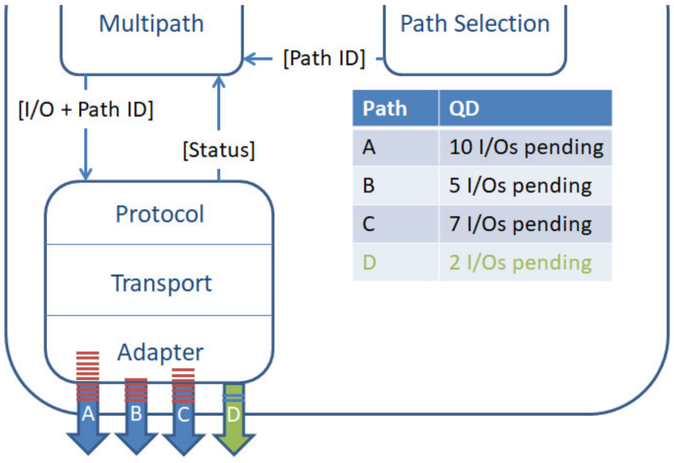
In this example, the path selection process using the round-robin algorithm returns the path identifier for path B since it is the next path choice in the round-robin cycle.



Path	RR
A	
B	Next
C	
D	

The queue-depth (QD) algorithm uses a technique based on the number of outstanding I/O requests queued for each path. The path with the least number of queued requests is the path returned as the next available path for I/O (that is, the path with the least number of pending I/Os).

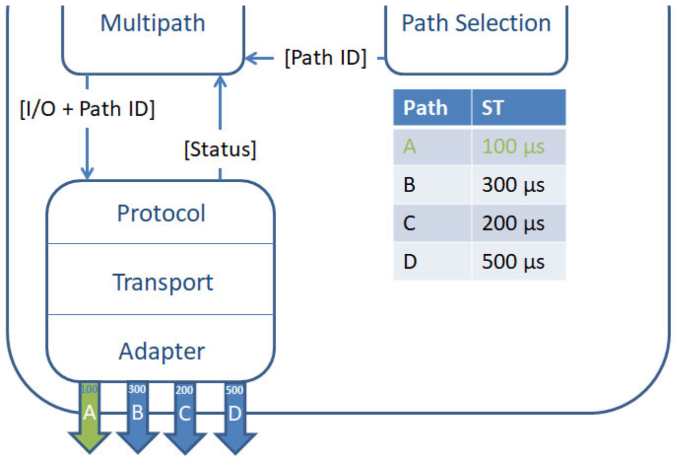
In this example, the path selection process using the queue-depth algorithm returns the path identifier for path D since it has the least number of pending I/Os with 2.



Path	QD
A	10 I/Os pending
B	5 I/Os pending
C	7 I/Os pending
D	2 I/Os pending

The service-time (ST) algorithm measures the amount of time that each I/O request takes to complete, and it maintains an average for each of the available paths. The path with the lowest average service time is the path returned as the next available path for I/O (that is, the path with the fastest service time).

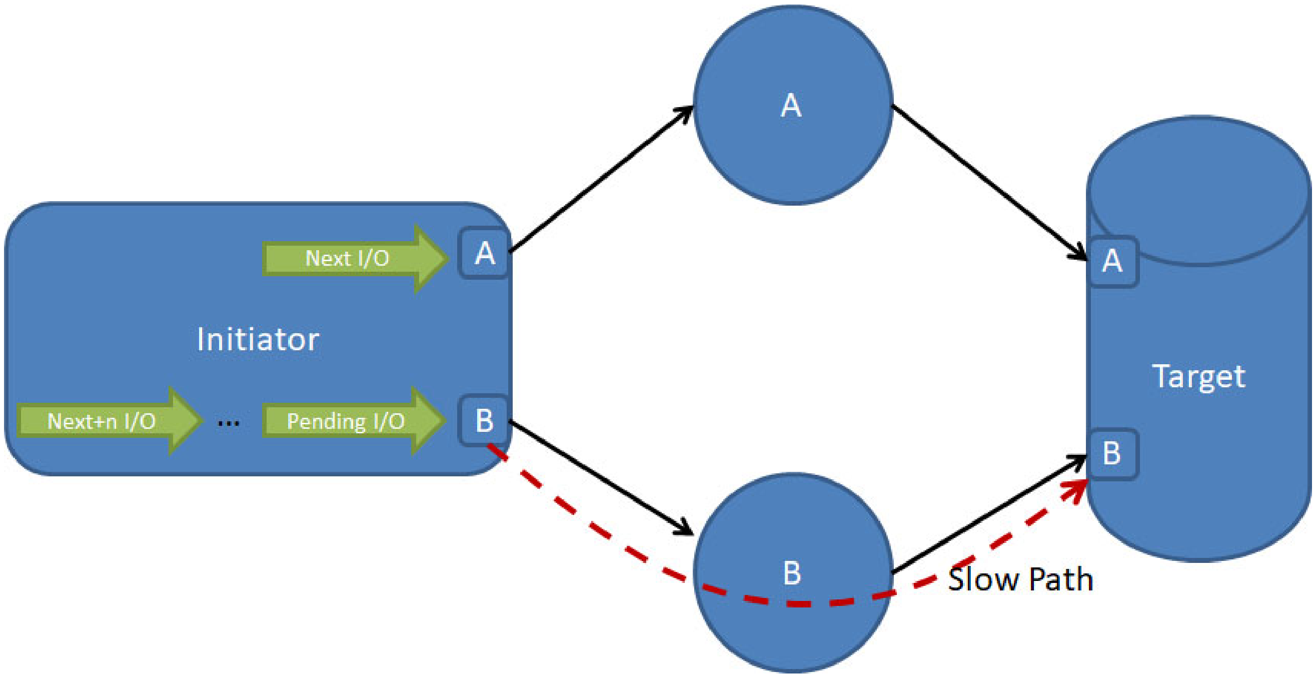
In this example, the path selection process using the service-time algorithm returns the path identifier for path A since it has the smallest/fastest service time at 100 µs.



Path	ST
A	100 µs
B	300 µs
C	200 µs
D	500 µs

3.3 Round-Robin Observations

During Broadcom’s investigation, it was observed that latency on one path had a dramatic effect on the overall performance and throughput of the multipath solution when an application depends on an I/O to complete before the next I/O is sent. The impact was isolated to solutions that used the round-robin path selection algorithm by default. In the following diagram, path B is the high-latency path with a pending I/O waiting for a response from the target to complete processing.



When the round-robin path selection algorithm is used, the next I/O is sent on path A, but the I/O after that (namely, the “Next+n I/O”) ends up waiting for the pending I/O on path B to complete before being sent. This behavior causes more and more I/O requests to be stacked up waiting on the high-latency path, which effectively reduces the overall throughput to the level of the slowest path (in this case, path B). Therefore, application performance for sequential I/O degrades to the level of the slow path regardless of the number of alternate paths available. Thus, the round-robin path selection algorithm is ineffective at addressing the impact of latency due to congestion on a single path.

In contrast, when the path selection function is configured to use one of the newer algorithms such as queue-depth or service-time, the impact to overall throughput is significantly reduced. In the scenario described above, the path selection process for queue-depth would note that the queue for path B is longer than the queue for path A and would consistently send the next I/O on path A. Likewise when the path selection process for service-time is configured, the process notes that the average service time on path A is consistently lower than path B and more I/Os are directed to path A. In both cases, using the newer algorithm allows the overall throughput of the solution to reach 50% or more depending on the latency level of path B.

Chapter 4: AIX Reference

In AIX, there is an environmental setting associated with the storage device that controls the path selection algorithm for the multipath driver. The setting is the *algorithm* variable, which can be set to one of the following values:

- round_robin

  The round_robin setting distributes I/O across all enabled paths to a storage device. The path control module attempts to distribute I/O across all enabled paths equally or according to the path_priority attribute. If a path fails or is disabled, it is no longer used for sending I/O.

  Note: To obtain optimal performance in a failed-path scenario, ensure that the ordered path list alternates paths between separate fabrics.

- shortest_queue

  The shortest_queue setting behaves similar to round_robin when the load is light, but as the load increases, this algorithm favors the path that has the fewest active I/O operations. Consequently, if one path is slow due to congestion or high latency, the other less congested paths are used for more of the I/O operations.

  Note: The path priority values are ignored by the shortest_queue algorithm.

- fail_over

  The fail_over setting causes I/O to be routed down only one path at a time. The path control module keeps track of all the enabled paths (per disk) in an ordered list. If the path being used to send I/O fails or is disabled, the next enabled path in the list is selected and I/O is routed to that path.

The percentage of I/O routed down each path can be weighted by setting the path_priority attribute on each path for each disk. Also, the path_priority attribute determines the sequence for path selection within a list of paths, which sorts the list by ascending path_priority value.

4.1 Configuration Example

The `chdev` command is used to change the characteristics of a device for AIX environments, which includes setting the *algorithm* attribute for selecting the path selection algorithm for the multipath driver. The syntax for the `chdev` command follows:

```
chdev -l Name [ -a Attribute=Value ... ] [ -f File ] [ -h ] [ -p ParentName ]
[ -P | -T ] [ -U ][ -q ] [ -w ConnectionLocation ] [ -g ]
```

Note, the `–a` option provides the ability to set an attribute value and is used to configure the *algorithm* attribute.

When changing attributes, the device logical name specified by the `–l` *`Name`* flag indicates the device parameters to be modified. When changing the device characteristics, the flags can be specified either on the command line or in the specified `–f` *`File`* flag. Use the `chdev` command to set the path selection algorithm for AIX and to set the reserve policy, if needed. The following is an example of using the `chdev` command to set the hdisk5 storage device to use the shortest_queue algorithm:

```
chdev -l hdisk5 -a algorithm=shortest_queue -a reserve_policy=no_reserve
```

4.2 Recommendation

Broadcom recommends setting the *algorithm* variable to shortest_queue for AIX environments in order to produce a significant improvement in the multipath driver load-balancing behavior.

4.3 References

The following are additional references to multipath configuration for AIX environments:

- IBM

  IBM AIX MPIO - Best practices and considerations

  IBM Support - The Recommended Multi-path Driver to use on IBM AIX and VIOS When Attached to SVC and Storwize storage

  IBM Knowledge Center - Path control module attributes

  IBM Knowledge Center - chdev Command

Chapter 5: Linux Reference

In Linux, there is an environmental setting associated with the storage device that controls the path selection algorithm for the multipath driver. The setting is the *path_selector* variable, which can be set to one of the following values:

- round-robin

  The round-robin setting causes the path selection function of the multipath driver to simply alternate I/O between the available paths in order (A, then B, then A, and so on). The end result is that the number of I/O requests is evenly distributed among the available paths.

- queue-length

  The queue-length setting causes the path selection function of the multipath driver to queue the next I/O for the path with the smallest queue length; thus, utilizing the path with the least number of pending I/O requests. The end result is that the I/O queues are kept even between the available paths.

- service-time

  The service-time setting causes the path selection function of the multipath driver to send the next I/O along the path with the shortest estimated service time. The service time is calculated by dividing the total size of the outstanding I/O for each path by the relative throughput for the path; thus, assigning the new I/O to the path with the shortest estimated service time. The end result is that the I/O requests are assigned to produce the same overall service time for all available paths.

Regarding these *path_selector* options, the *rr_min_io* variable determines the number of I/O requests that are assigned to the selected path before moving to the next path as dictated by the *path_selector* algorithm. This allows for groups or batches of I/O requests to be allocated to a path before moving to the next path in the selection process. Note, the default value for *rr_min_io* is 1, which is the recommended value.

5.1 Configuration Example

Modifying the *path_selector* variable determines which algorithm the multipath driver uses. The *path_selector* variable is located in the `multipath.conf` file, which consists of entries in the following general format:

```
<section> {
       <attribute> <value>
       ...
       <subsection> {
               <attribute> <value>
               ...
       }
}
```

The `<section>` tag indicates the element to be configured, and the `<attribute> <value>` tag pairs indicate the variables and their assignments. The following is an example of setting the *path_selector* variable to the queue-length value for a storage device:

```
device {
   vendor                  "Vendor or Manufacturer's Name"
   product                 "Storage Product Name"
   path_grouping_policy    group_by_serial
   polling_interval        30
      failback                immediate
      no_path_retry           5
      rr_min_io               1
      path_selector           "queue-length 0"
      path_checker            tur
      user_friendly_names     yes
}
```

Selecting either of the other path selection algorithms is achieved by simply replacing the *path_selector* variable with one of the following examples:

```
device {
   ...
      path_selector           "service-time 0"
      ...
}
```

Or

```
device {
   ...
      path_selector           "round-robin 0"
      ...
}
```

It should also be noted that in order to effect any change to the `multipath.conf` file, the `multipathd` daemon must be restarted. This can be accomplished with a server reboot or manual daemon restart.

5.2 Recommendation

Broadcom recommends setting the *path_selector* variable to `queue-length 0` in the `multipath.conf` file for Linux environments in order to significantly improve the multipath driver load-balancing behavior.

5.3 References

The following are additional references to multipath configuration for Linux environments:

- Red Hat

  Red Hat Customer Portal - RHEL8 Configuring device mapper multipath - Chapter 4. Modifying the DM-Multipath configuration

- Ubuntu

  Ubuntu Docs - Device Mapper Multipathing - Introduction

- Wikipedia

  Linux DM Multipath

Chapter 6: Windows Reference

In Windows, there are several load-balancing policies that can be configured for the multipath I/O (MPIO) driver associated with a storage device. The supported policies are as follows:

- Fail Over Only

  The fail over only policy does not perform load balancing. It uses a single active path and categorizes the remaining paths as standby paths. The active path is used for all I/O until it fails, and then one of the standby paths is used. When the path that failed is reactivated or reconnected, the standby path that was activated returns to standby.
- Least Blocks

  The least blocks policy uses the path with the least number of data blocks currently being processed.
- Least Queue Depth

  The least queue depth policy uses the path with the fewest currently outstanding I/O requests for the next I/O.
- Round Robin

  The round robin policy uses all available paths for MPIO in a balanced way. It is the default policy for the active-active storage controller model.
- Round Robin with Subset

  The round robin with subset policy allows the application to specify a set of paths to be used in a round-robin fashion and to reserve the remaining paths as standby paths. The paths in the primary pool are used until all of the paths fail, and then one of the standby paths is used.
- Weighted Paths

  The weighted paths policy assigns a weight to each path that indicates the relative priority of the path (the larger the number, the lower the priority). I/O is sent on the path with the least weighted priority from among the available paths.

6.1 Configuration Example

The following is an example of configuring the MPIO load-balancing policy using the Windows powershell:

```
PS C:\> Set-MSDSMGlobalLoadBalancePolicy -Policy LQD
```

This example sets the policy to least queue depth. Other policies can be selected by specifying the appropriate policy parameter.

Policy	**Description**
None	Clears any currently configured default load-balancing policy
FOO	Fail Over Only
LB	Least Blocks
LQD	Least Queue Depth
RR	Round Robin

The following is an example of configuring the MPIO load-balancing policy using the Windows `mpclaim` command:

```
C:\> mpclaim.exe –L –M 4
```

This example sets the policy to least queue depth. Other policies can be selected by specifying the appropriate load-balancing parameter.

Parameter	Definition
0	Clear the Policy
1	Fail Over Only
2	Round Robin
3	Round Robin with Subset
4	Least Queue Depth
5	Weighted Paths
6	Least Blocks
7	Vendor Specific

6.2 Recommendation

Broadcom recommends configuring the multipath load-balancing policy to least queue depth for Windows environments, which typically produces a significant improvement in the multipath driver load-balancing behavior.

6.3 References

The following are additional references to multipath configuration for Windows environments:

- Microsoft

  Microsoft Documentation - Windows Server 2008 R2 and Windows Server 2008 - Installing and Configuring MPIO - Change the load-balancing policy settings

  Microsoft Documentation - MPIO - Set-MSDSMGlobalDefaultLoadBalancePolicy

Chapter 7: VMware Reference

In VMware, Multipathing Properties controls the path selection algorithm for the multipath driver. Multipathing Properties can be edited to one of the following values:

- RoundRobin

  With the RoundRobin setting, the host uses an automatic path selection algorithm rotating through all active paths for active-passive configurations or through all available paths for active-active configurations. The RoundRobin setting is the default for many storage devices and can be used with both active-active and active-passive configurations.

- MostRecentlyUsed

  With the MostRecentlyUsed setting, the host selects the path that it used most recently for the next I/O. If the path becomes unavailable, an alternative path is selected and is used for all subsequent I/O (until it becomes unavailable). This setting does not cause the path selection algorithm to revert back to the original path when it becomes available again.

  Note: There is no preferred path setting with the MostRecentlyUsed policy, and it is the default policy for most active-passive configurations.

- Fixed

  With the Fixed setting, the host uses the designated preferred path. If the preferred path has not been configured, the first working path discovered at system boot time is used.

  Note: To use a particular preferred path, it must be specified manually. The Fixed setting is the default policy for most active-active configurations.

7.1 Configuration Example

The path selection policy can be changed in the vSphere Web client by using the **Manage Paths** dialog box. Use the following procedure to select the policy:

1. Access the vSphere Web client navigator.
2. Select the **Manage** table and click **Storage**.
3. Select **Storage Devices**.
4. Select the storage device to be changed.
5. Select the **Properties** tab.
6. Under **Multipathing Properties**, select **Edit** and select the path selection policy.

   Note: If the Fixed policy is selected, specify the preferred path as well.
7. Select **OK** to save the changes and exit the dialog box.

VMware supports Fixed, MostRecentlyUsed, and RoundRobin. It is also possible to select a third-party policy if a path selection policy is installed on the host. The third-party policy will appear as one of the options when editing the Multipathing Properties.

7.2 Recommendation

Broadcom recommends editing the Multipathing Properties to RoundRobin for VMware environments, which typically significantly improves the multipath driver load-balancing behavior.

7.3 References

The following are additional references to multipath configuration for VMware environments:

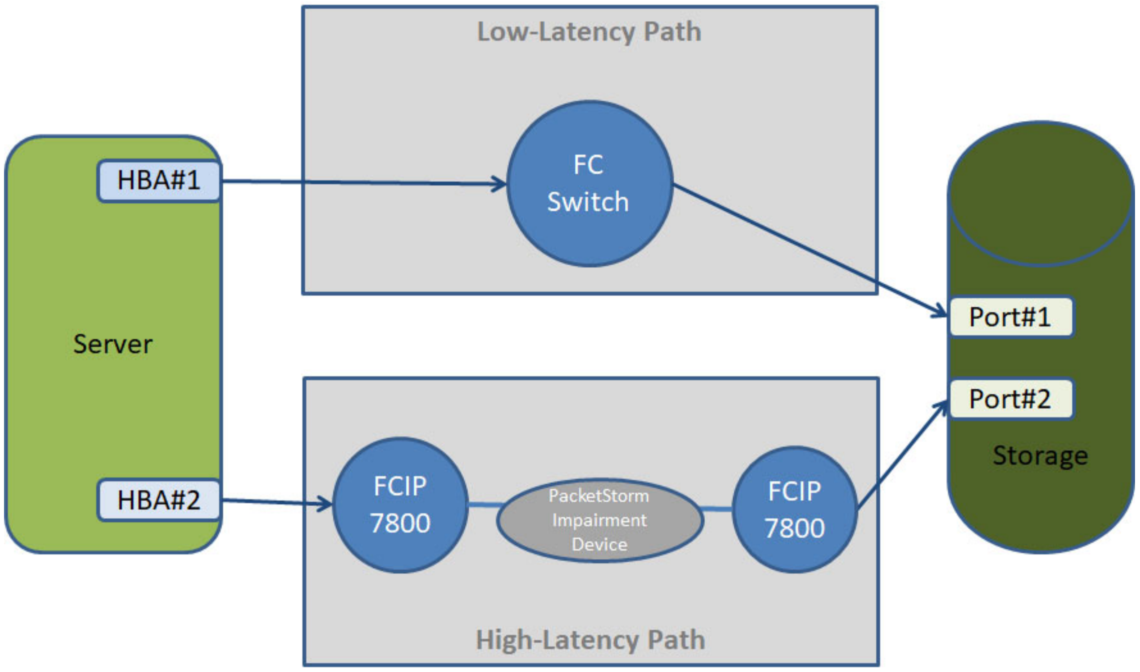
- VMware

  VMware Docs - vSphere - VMware Native Multipathing Plug-In

  VMware Docs - vSphere - Change Default Parameters for Latency Round Robin

Chapter 8: Notes and Observations

The recommendations provided in this technical brief are based on tests performed by Broadcom's OEM Test Lab. The test procedure focused on the effects of latency on multipath drivers in an A/B fabric configuration where latency was introduced on one path with an increasing delay (see the following diagram).



The primary objective of this testing was to evaluate the effectiveness of the *path_selector* variable settings for Linux. In addition to the *path_selector* variable evaluation, Broadcom's OEM Test Lab also examined the effect of the following variables on the outcome of the *path_selector* variable setting:

- *SCSI timeout*: Reduced from the default value of 30 seconds to 4 seconds.

  For example, SCSI timeout=4 (set in `/sys/block/sd[x]/device/timeout`)
- *Queue depth*: Reduced from the default value to 16.

  For example, QUEUE depth=16 (set in `/sys/module/qla2xxx/parameters/ql2xmaxqdepth`)
- *rr_min_io*[2]: Increased from a default value of 1 to 16.
- *polling_interval*: Reduced from a default value of 30 seconds to 15 seconds.

The purpose of this portion of the investigation was to determine if changes in these values resulted in any additional changes in the behaviors observed by the various *path_selector* variable settings. Although the test procedure did not perform an exhaustive examination of these variables, the performance results during the latency test indicated that there was no significant change in behavior when these variables were modified.

2. The *rr_min_io* variable determines the number of I/O requests that are assigned to the selected path before moving to the next path as dictated by the *path_selector* algorithm (see Chapter 5, Linux Reference).

The test team concluded that further adjustment of these particular variables would have no significant effect on the primary test results and, therefore, further manipulation of these variables was not included in the primary test parameters. It was also noted that the settings associated with these variables could improve behaviors that are beyond the scope of the *path_selector* latency evaluation.

Revision History

MP-Load-Bal-OT101; March 17, 2021

- Updated the VMware ESXi version.

MP-Load-Bal-OT100; June 5, 2019

- Initial release.

BROADCOM®