

Queue Depth

Author Glenn Brackenridge,
Performance Consultant, EMEA
Date Saturday, 14 February 2009
Revision Number V01.1



The “Active Queue” represents the customers who managed to get inside the door. The “Wait Queue” is outside.

Notices and Disclaimer

Copyright© 2009 Hitachi Data Systems, Inc.

No part of this document may be reproduced or transmitted without written approval from Hitachi Data Systems, Inc.

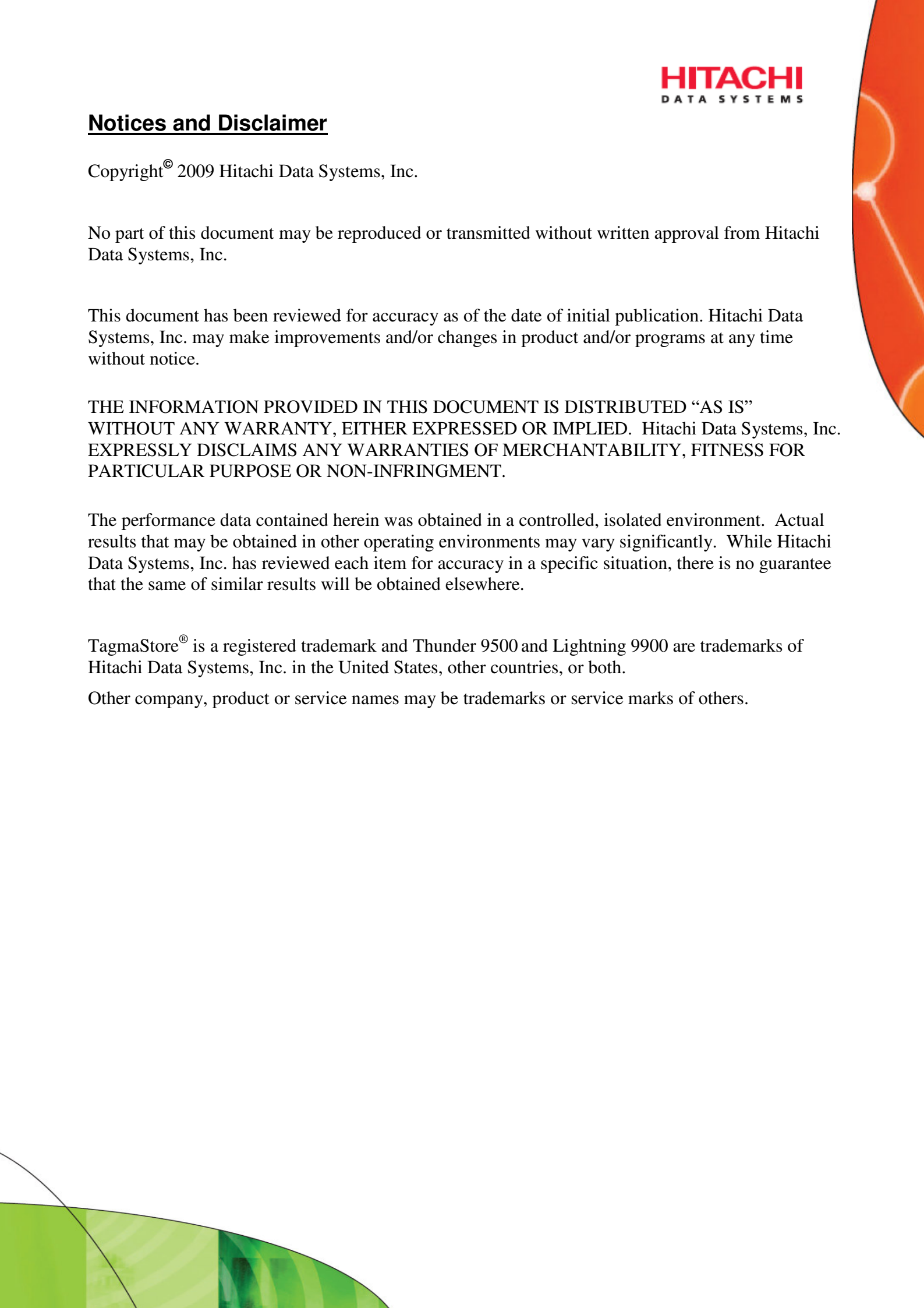
This document has been reviewed for accuracy as of the date of initial publication. Hitachi Data Systems, Inc. may make improvements and/or changes in product and/or programs at any time without notice.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED “AS IS” WITHOUT ANY WARRANTY, EITHER EXPRESSED OR IMPLIED. Hitachi Data Systems, Inc. EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR PARTICULAR PURPOSE OR NON-INFRINGEMENT.

The performance data contained herein was obtained in a controlled, isolated environment. Actual results that may be obtained in other operating environments may vary significantly. While Hitachi Data Systems, Inc. has reviewed each item for accuracy in a specific situation, there is no guarantee that the same of similar results will be obtained elsewhere.

TagmaStore® is a registered trademark and Thunder 9500 and Lightning 9900 are trademarks of Hitachi Data Systems, Inc. in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.



Abstract

- On the face of it queuing is a simple concept. We have no difficulty understanding that we should stand in the shortest queue at a supermarket. Some other factors might influence our decision over the anticipated performance of the queue, for example how much each of the customers has in their trolley, whether are they buying bulk packs or many individually priced items etc. We might choose a checkout near the exit or we might try to avoid the trainee operator, and so on.
- So we can appreciate that queuing is a simple concept with complicating factors. The same is true with Storage subsystem queuing, and the complicating factors include the need to specify queue depth limits and operate within parameters such as timeout values. Getting it wrong can have minor repercussions such as lower than expected levels of performance right to the other end of the scale of seriousness where a SCSI timeout might cause loss of access to a Lun and risk application failure.
- A SCSI timeout and subsequent application failure is the supermarket equivalent of the customer waiting at a checkout so long that they die before they get served. The primary objective of this document is to help the reader avoid these critical situations. The secondary objective is to explain how queuing can be manipulated to enable maximum performance while maintaining efficient utilization of available resources.
- A final objective is to introduce related topics that can both educate and invite further discussion. This author appreciates that although the information may be the best that is available at this moment there is always room for further detail, correction of errors and clarification through actual example. In this respect I encourage feedback and constructive comments.
Glenn.brackenridge@hds.com

Contents

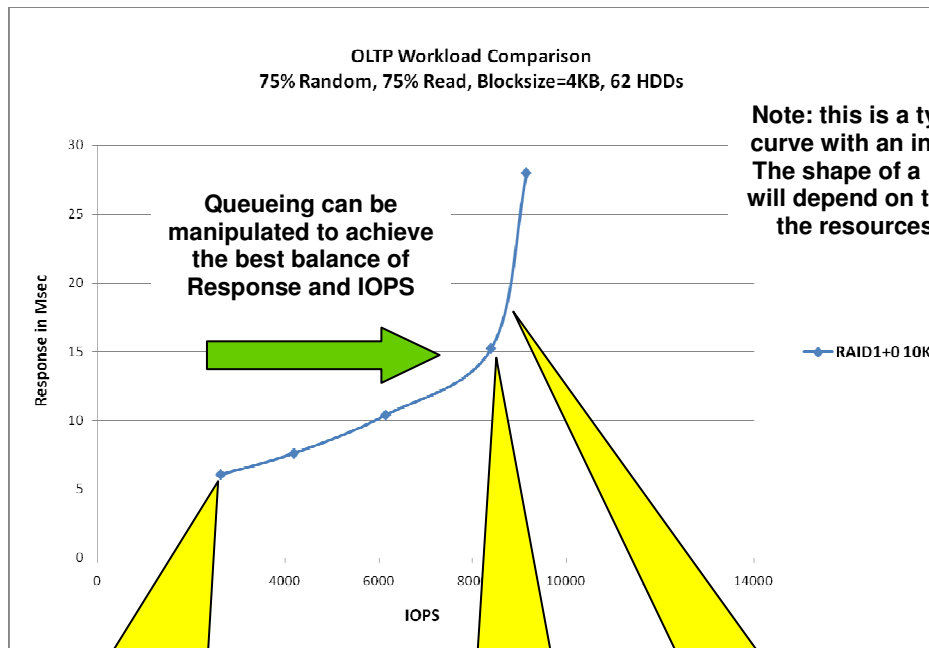
- Tagged Command Queueing (TCQ) overview
- The Value of TCQ: A queue may not always be bad – graph
- Queue Depth and Number of Resources
- Workload Queueing attributes
- Impact of HBA Lun Queue Depth – graph
- The “magic” optimum queue sizes
- Storage Port Tags
- What happens when the Tag Pool is exhausted
- Ensuring we don’t exhaust the Port Tags
- Effective impact of exceeding the Port Tag count
- Reaching the Lun Queue Depth limit
- Lun Queue Depth calculation and examples
- Target Mode Queue Depth calculation and examples
- Example of AMS Queue Depth configuration options
- Queue Tiering
- Setting Queue Depth as a fairness mechanism
- Impact of LUSE and VDEV Striping on Queue Depth
- LUSE and VDEV Striping workload distribution
- Little’s Law and Queueing Calculation
- Case History 1: SCSI Timeouts
- Case History 2: Lower than expected performance on an AMS500
- Taking the risk ...
- Queue Depth, Target Mode and Clustering
- Storage reporting of Queue Depth
- Monitoring Queue size
- The Importance of Differential Response
- Previous HDS Queue Depth documents
- Queue Jumping

Tagged Command Queuing (TCQ) overview

- Tagged Command Queuing (TCQ) is a technology built into disk subsystems that allows them to receive multiple read and write requests and service them in any order with the objective of increasing levels of performance. Performance being defined here as more IOPS, lower response times and greater throughput (MB/sec).
- The order of completion of queued elements is influenced by several factors:
 - Elevator Optimization – the sequence of physical disk accesses may be changed in order to reduce the overall time spent seeking.
 - Note: Elevator Optimization will usually reduce overall time spent seeking but may result in some individual I/O's having longer Response (the I/O at the end of a sequence).
 - Cache Hits – data can be read or written to Cache while the storage subsystem is waiting for other disk access to complete.
 - Native Command Queuing (NCQ) is the term for SATA queue handling. In HDS Storage systems TCQ is implemented at the host-storage interface and NCQ implemented for the internal RAID SATA disks.

The Value of TCQ: A queue may not always be bad

- Systems with multiple resources, for example RAID Arrays with several disks supporting each Lun, will exhibit the lowest Response Times when the I/O queue is low, but will equally exhibit low utilization levels at the RAID Group level, leading to the situation where not all the potential performance can be extracted within any target Response Time - IOPS are left on the table.
- Increasing the I/O queue past the point where the RAID Groups start to become heavily utilized usually results in a sharper rate of increase of Response Time compared to IOPS. The point at which this change occurs is called the “knee of the curve”, or “hockey stick” if the transition is sharp.
- Implementing a level of queuing that approaches but does not pass the knee of the curve would be the objective of the storage architect looking for a balance between performance and cost.



Note: this is a typical Response curve with an increasing queue. The shape of a Response curve will depend on the workload and the resources being tested.

Queueing can be manipulated to achieve the best balance of Response and IOPS



Response is lowest at this point but the IOPS Rate is also lowest. Queueing increases IOPS where resources are available

The "Knee of the Curve" or "Hockey Stick"

Queueing after this point increases Response faster than IOPS

Not all Queueing is bad – provided contention can be avoided

Queue Depth and Number of Resources

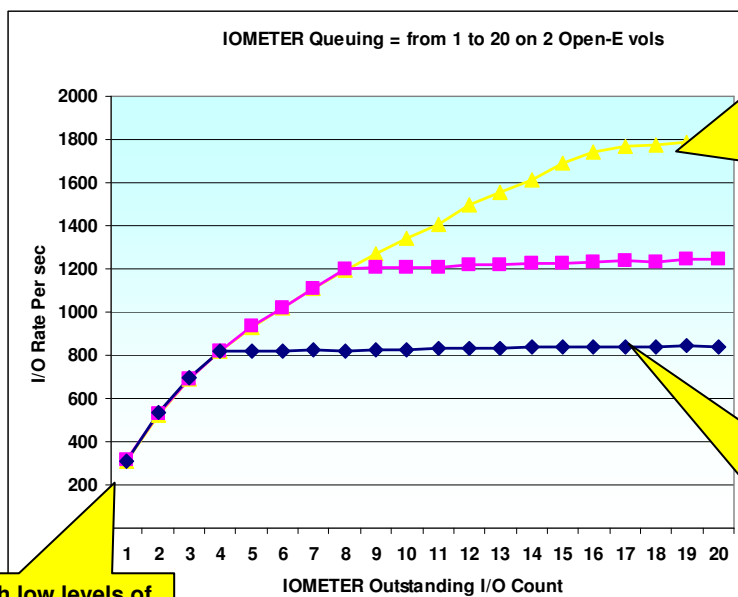
- If the active Queue size is significantly lower than the number of resources, then resource utilization is likely to be low. For example an I/O queue of 4 on a Lun from an 8D+8D RAID1+0 Group with 16 disk drives would potentially be leaving 12 disks idle (with Random Reads). This would present the opportunity to increase the size of the I/O Queue or drive I/O to 4 times the number of Luns on the same RAID Group to increase the IOPS rate without any severe impact to RAID Group utilization or Response Times.
- If the active Queue size is greater than the number of resources then resource utilization will be high and contention past the knee of the curve is likely to be present. For example an I/O queue of 16 to a Lun on a 4D+1P RAID5 RAID Group with 5 disk drives will create a queue of at least 3 on each disk drive. With Random Read access the last I/O in the queue will experience three times the normal mechanical disk latency. From a design perspective, excessive queuing indicates that insufficient resources are available to service this workload.

Workload Queueing attributes

- Some workloads naturally generate multiple concurrent commands, for example a Multi-User Mail or OLTP system. These applications are usually Response-critical, where the level of queuing needs to be kept low – and certainly not go past the knee of the curve. Assigning sufficient resources to accommodate the workload is a key part of design for these applications.
- Response Time and RAID Group Utilization are key metrics that can indicate whether HBA Queue Depth limits are set appropriately.
- Other workloads may be serial or synchronous, such as Batch operation, backup or video streams, and Queue levels will always be low. In these situations increasing HBA Queue Depth values may not have any impact.

Impact of HBA Lun Queue Depth

- The HBA Lun Queue Depth is a throttle mechanism that limits the number of concurrent commands that can be issued to a Lun. Since the storage subsystem can only optimize I/O's that are passed to it then it is no surprise that achievable IOPS will increase as the HBA Lun Queue Depth is increased. Increasing the HBA Lun Queue Depth will continue to enable more and more IOPS provided that 1) the application generates an appropriate I/O queue itself, and 2) that there are sufficient storage resources available to service the I/O demand.



The higher the Lun Queue Depth value the more optimization is possible and therefore greater levels of performance ...

... But high Lun Queue Depth values risk breaking the #Tags / #Luns formula

Performance increases with queued workloads are constrained by the HBA Lun Queue Depth

Storage subsystems will not get to process and optimize more commands than are allowed by the Lun Queue Depth value

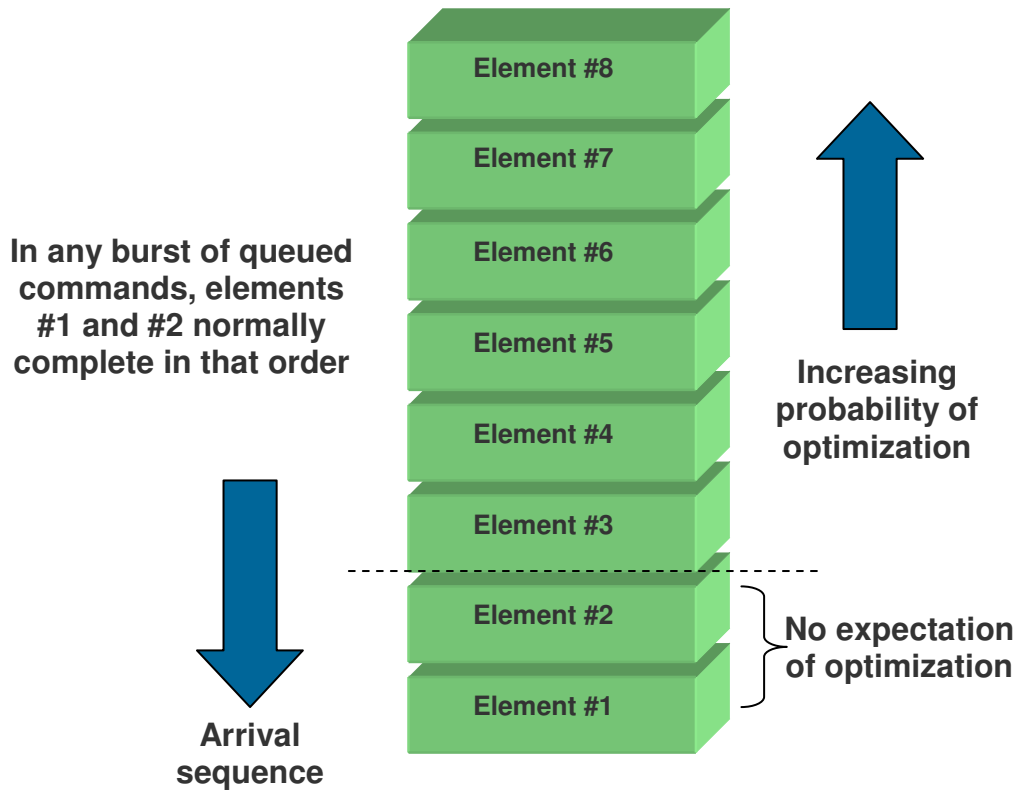
Applications with low levels of Queuing depend on the latency of the resources rather than Queue limits.

The HBA Lun Queue Depth value restricts the number of I/O's that can be optimized

The “magic” Optimum Queue size – “and 4 shall be the number”

- If only one queue element exists for a resource then it will be executed without any Queue optimization and will not be subject to any optimization processing.
- If a second queue element arrives, it becomes the de-facto head of queue and many optimization processes ignore the head of the queue – they can become executable at any instant. This depends of the arrival rate of new elements and the latency of the element being executed.
- Subsequent queue elements from #3 onwards are considered for optimization and elements #1 and #2 usually complete in the same order. That is, as the elements pass down the queue position they become locked in place when they reach position #2.
- This leads to the situation where a Queue limit of 3 is unlikely to deliver any significant benefit from optimization and observable optimization will be more apparent with a Queue limit of 4. For this reason the lowest “magic” HBA Lun Queue Depth value is 4.

- Clearly, a larger Lun Queue Depth will enable more optimization but 4 remains the lowest value that we should allow.



Optimization requires a Queue of larger than 2 before any optimization can be expected



“Minimum Queue Depth – and 4 shall be the number, thou shall not go to 3 unless it is to immediately proceed on to 4. 2 is definitely out!”

Optimum “large” Queue size

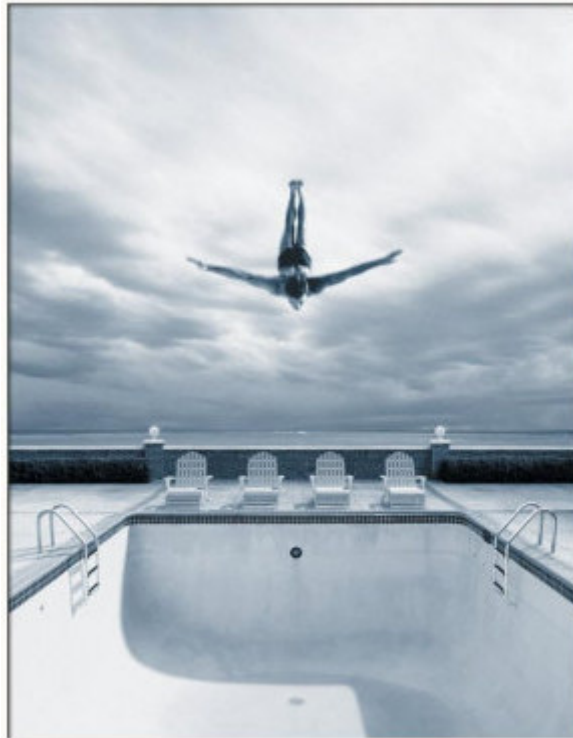
- Although IOPS rates increase with larger queue sizes, the rate of increase is not linear and there is a progressive loss of efficiency in the optimization process. Issuing 16 commands to a set of 16 disk drives will not result in 1 I/O per disk drive. The statistical and actual result is that some drives will have to handle multiple I/O's and others none. I.e. a queue of 16 will never evenly distribute itself across 16 resources.
- In fact it is necessary to have a queue a lot larger than 16 to ensure that all 16 resources will be active and this leads to the phenomenon of increasing Response Times as we approach the knee of the curve even though common sense tells us that Response Times could be lower at this level of queuing.
- Some resources may acquire an individual queue of 2, 3 or even 4 before all resources are active.
- Lastly, Processor cycles are consumed managing the Queue itself and represents an additional factor in not obtaining linear increases of performance with Queue Depth.
- There will come a point where splitting a large Queue into two smaller queues will deliver a beneficial overall increase in IOPS. This will, of course, require dividing up the resources into two sets.

Storage Port Tags

- The mechanism that allows I/O commands to be completed in any order is the use of Tags. Each command is given a **Tag** number by the Storage subsystem to enable the server to recognize which processing thread generated that command. The number of Tags, or the size of the Tag pool, defines the number of concurrent commands than can be supported by the port.
- Each Storage Port can handle a specific number of concurrent commands (Tags) depending on the machine type.
 - 9500V/AMS has a pool of 512 Tags per Port
 - USP has a pool of 1024 Tags per Port
 - USPV has pool of 4096 Tags per MP (Port pair)
- In all current HDS Storage subsystems the Tags are a pooled resource and Tags are returned to the pool immediately after a command is completed. Tags are not used or reserved until a command is received and mapping Luns to a Port does not consume Tags.
- Note: The number of Luns supported by a Storage Port is not related to the number of available Tags. Any instances of the same values are architectural design co-incidence.

What happens when the Tag Pool is exhausted

- If all the Tags are in use, then no more commands can be accepted by the port until one of the current commands has been completed and its Tag has been freed-up.
- If the pool becomes exhausted the Storage Port can report Queue Busy, Queue Full or simply not respond to new command requests.
- Some Operating Systems and HBAs may tolerate or ignore Queue Full/Busy until SCSI timeouts occur. Other Operating Systems are less tolerant and loss of access to the Lun may occur before any timeout.
- In some cases the Storage Port can implement its own counter-measures to prevent the exhaustion of Tags but this typically results in the same end result but without reported events. Irrespective of the actual condition, running out of Tags is undesirable



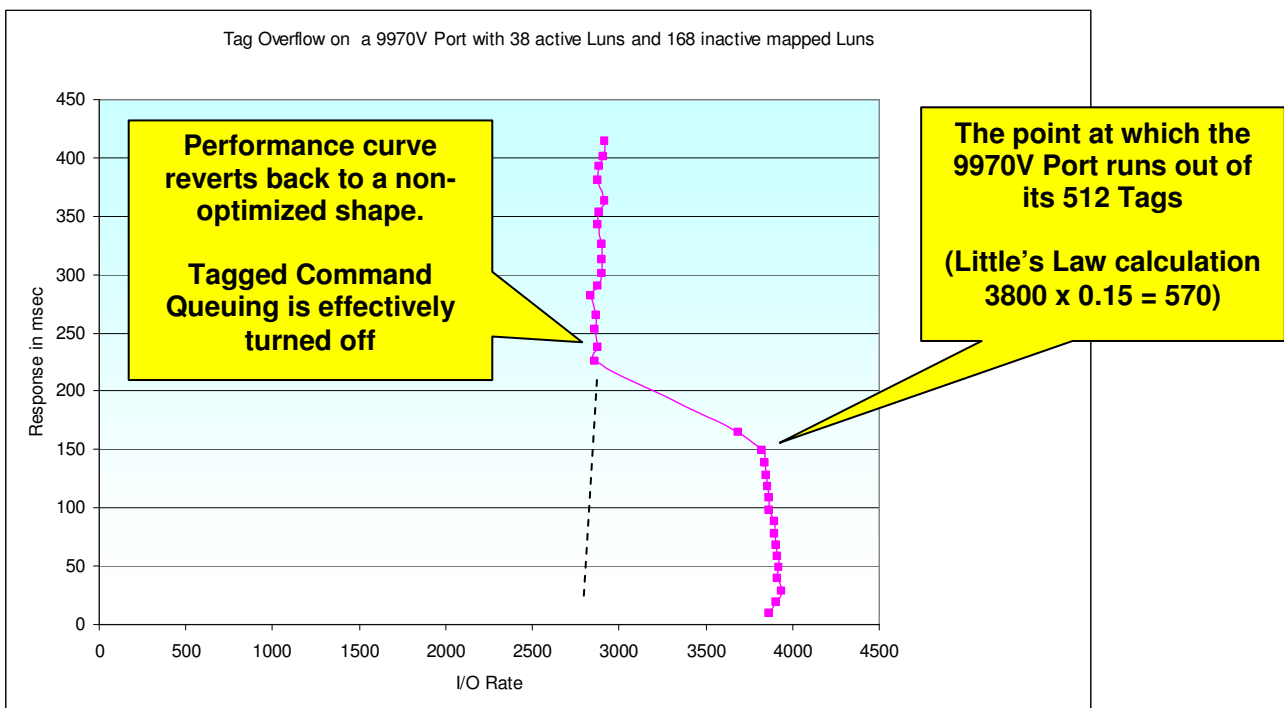
“Sorry, the Tag Pool is empty”.

Ensuring we don't exhaust the Port Tags

- Queue Depth limits should be defined by the server/HBA combination to make sure that the number of queued or concurrent commands issued does not exceed the Tag count of the Storage system. Three mechanisms exist to limit the Queue Depths:
 - **HBA Lun Queue Depth** – the HBA only allows this number of concurrent commands to be issued to a Lun, holding further commands in the server. The Port Tag count must be divided by the number of Luns to obtain the recommended HBA Lun Queue Depth value
 - **HBA Target Queue Depth** – the HBA only allows this number of concurrent commands to be issued to the Target. The Port Tag count must be divided by the number of servers accessing the Port.
 - **O/S Level Throttling** – the O/S is responsible for defining the maximum number of commands to a Lun. The Port Tag count must be divided by the number of Luns to obtain the recommended Throttle value

Effective impact of exceeding the Port Tag count

- If the Tag pool is exhausted then new commands are only accepted on a one-at-a-time basis as another command completes. The shape of the performance curve implies that Command Queuing is effectively turned off in these situations, further adding to any performance impact.
- If there are multiple servers attached to different Host Groups on the port then they are serviced in a Round Robin mode. This can result in non-intuitive sharing of the queue resources. For example, a lightly queued workload will start to receive the same level of service as a heavily queued load because they each have an I/O pending
- Little's Law can be used to double check the validity of any suspected over-queuing against IOPS and Response.



“The performance curve reverts back to a non-optimization”

Reaching the Lun Queue Depth limit

- Reaching the Lun Queue Depth does not necessarily involve any serious compensating actions, for example exceeding 32 commands to a Lun on a USPV Port does not turn off Command Queuing. The Port does not accept

any new commands over the 32 and the HBA does not expect to issue more than 32 commands to a Lun on a USPV. The additional queuing occurs in the driver and O/S.

Lun Queue Depth calculation and examples

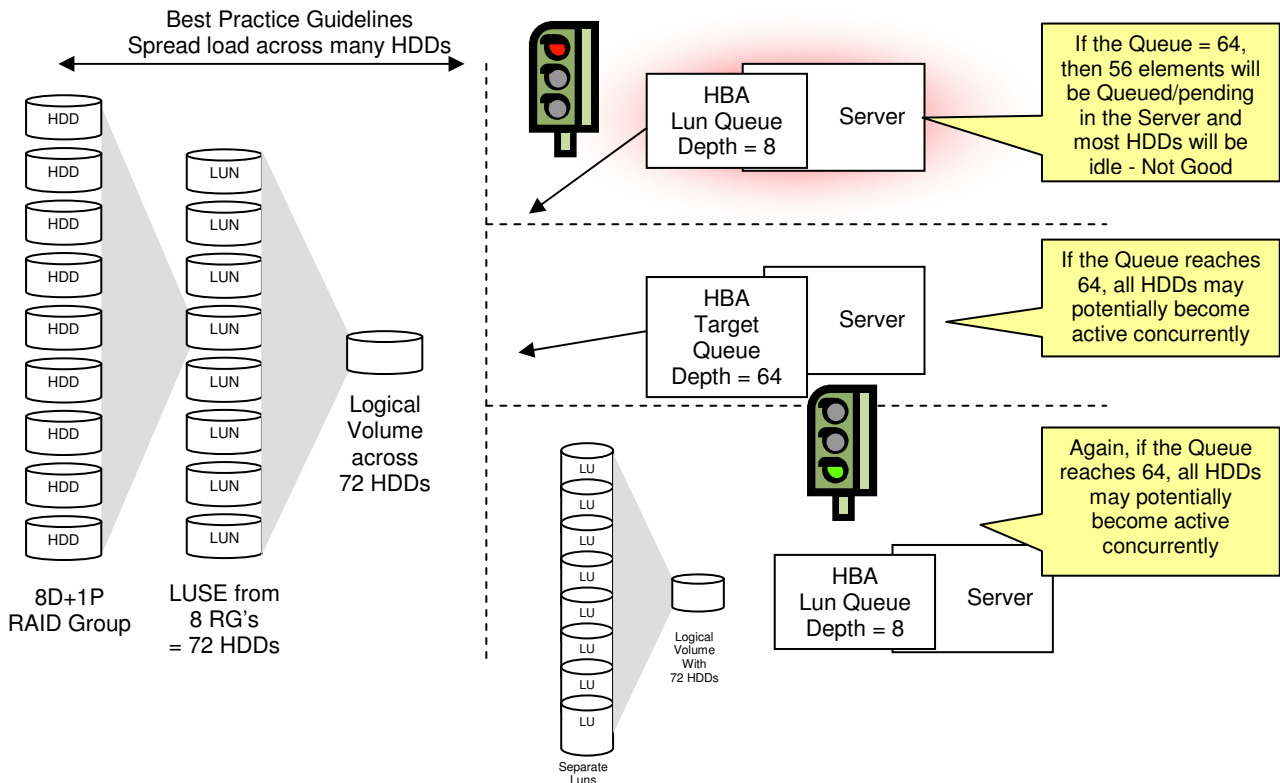
- Recommended HBA Lun Queue Depth calculation:
 - $\#QD = \#tags / \#luns$, or ...
 - $\#QD = \#tags / (\#luns \times \#clustered\ servers)$
- Note: if this results in the Lun Queue Depth being a very low value it may be better to use Target Mode.
- Example 1: USP (1024 Tags) port with 10 servers each accessing 10 different Luns:
 - $1024 / 100 = 10.2 \rightarrow$ set the HBA Lun Queue Depth to 10 for safe operation
- Example 2: USP (1024 Tags) with 2 servers accessing the same 100 luns:
 - $1024 / (100 \times 2) = 5.2 \rightarrow$ set the HBA Lun Queue Depth to 5 for safe operation
- Luns that are not active can be excluded from the #Luns in the Queue Depth calculation

Target Mode Queue Depth calculation and examples

- Recommended Target Mode Queue Depth calculation:
- $\#QD = \#tags / \#servers$:
- Example 1: AMS (512 Tags) port with 10 servers each accessing 10 different Luns (Note: the number of Luns is not important!)
- $512 / 10 = 51.2 \rightarrow$ set the HBA Target Mode Queue Depth to 51 for safe operation
- Example 2: AMS (512 Tags) with 2 servers accessing the same 100 luns (Note: the number of Luns is not important!):
- $512 / 2 = 256 \rightarrow$ set the HBA Target Mode Queue Depth to 256 for safe operation.

- Servers that are not active can be excluded from the #servers in the Queue Depth calculation.
- Target Mode can simplify the calculations, especially for clustered configurations.

Example of the AMS Queue Depth configuration options



“LVM Striping can resolve Queue Depths with large volumes”

Queue Tiering

- Queue limits should not be designed to exceed the available resources. In a Storage environment Queue limits should be designed to maximize, but not exceed the available resources at any Tier.
- Once the Queue limit has been reached for a Storage resource, it is more efficient to manage extra queuing in higher levels. Server CPU cycles are less expensive than Storage CPU cycles and Queues in the server may even help self-regulate the Applications generation rate of new I/O commands.

Setting Queue Depth as a fairness mechanism

- Setting Queue Depths to their highest workable value may result in busy Luns hogging too much of the resources and preventing other Luns on the same Port from receiving 'sufficient' attention.
- Reducing the Lun Queue Depth for non-critical Luns can preserve resources for the Performance-critical Luns.

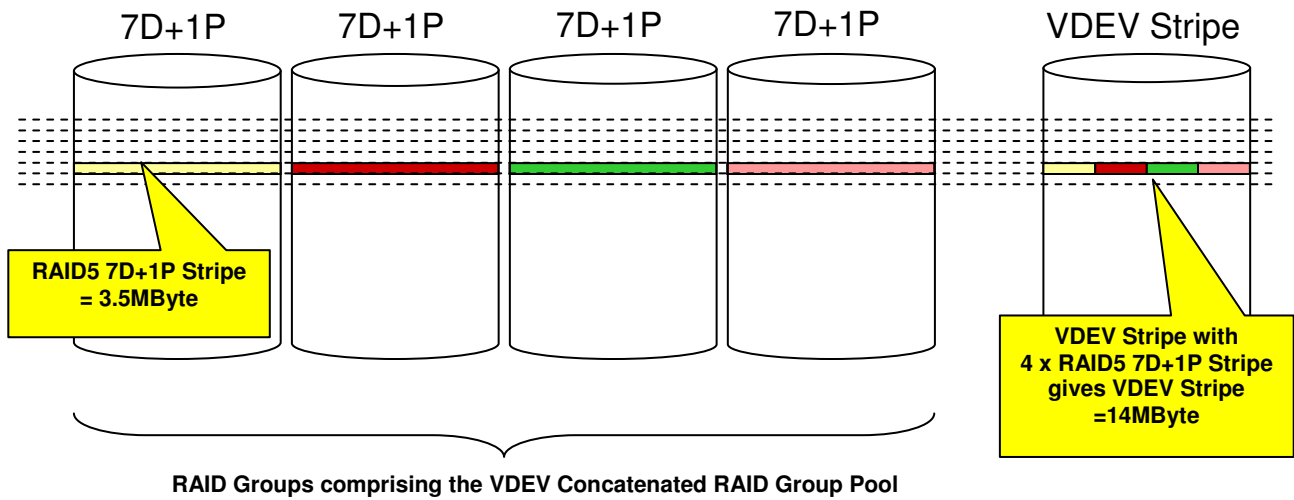
Impact of LUSE and VDEV Striping on Queue Depth

- LUSE and VDEV Striped Luns are subject to the HBA Lun Queue Depth setting. The entity that is subject to the HBA Lun Queue Depth is the mapped Lun and the HBA has no perception of the internal structure of a Lun even though the LUSE Lun, or VDEV Striped Lun, may be comprised of LDEVs from many RAID Groups.
- In such cases the HBA Lun Queue Depth may actually restrict the LUSE Lun RAID Groups from being fully utilized unless additional Luns are mapped from the same RAID Groups. This increases the level of concurrency and boosts levels of utilization. Queueing on the component disk drives will be increased but smaller queues will exist on the mapped Luns.
- Target Mode can enable higher peak queue depths to exist on a LUSE Lun:
 - AMS supports full target limit per Lun (512)
 - USPV supports maximum of 32 per Lun irrespective of mode

LUSE and VDEV Striping workload distribution

- LUSE concatenates the capacity of its component LDEVs so that there is a dependency on the Application to distribute its workload randomly across the LUSE Lun capacity in order to achieve even load distribution. Individual LDEVs in the LUSE may still be subject to excessive Queuing and utilization if load distribution is not fully random.
- VDEV Striping distributes its workload evenly across all the member RAID Groups with a RAID Stripe equal to the sum of the component RAID Stripes from each RAID Group in the pool.



- Note: VDEV/LDEV/RAID Group Concatenation refers to the concatenation of the individual RAID Stripes, **NOT** concatenation of the overall space.



“Concatenated RAID Groups provide Striping with a new RAID Stripe of up to 14 MB”

Little’s Law and Queuing calculation

- Little’s Law can be used to perform a quick check of the validity of Queue size measurements and to estimate Queue sizes when other metrics are not available.
- Queueing Depth = Throughput Rate x Response Time
- Example: QD = 9,000 IOPS x 0.010 secs = 90
- Note: In order to exhaust the Tags on a Port is it usually necessary to have both a high IOPS Rate combined with a high Response Time.
- Little’s Law can be structured to calculate IOPS, Response or Queue Depth if the two other values are known.

The "What's My Response/Queue Depth" Calculator

Enter IOPS >	<input type="text" value="3800"/>	Enter Response in ms >	<input type="text" value="150"/>	=	<input type="text" value="570"/>	Queue Length
Enter IOPS >	<input type="text"/>	Enter Queue Length >	<input type="text"/>	=	<input type="text"/>	Response in ms
Enter Response in ms >	<input type="text"/>	Enter Queue Length >	<input type="text"/>	=	<input type="text"/>	IOPS

Note: this is based on Little's Law, where you can work out any one of Queue Length, Response Time and IOPS if you know the other two.

“Little’s Law can provide a useful verification of queue lengths”

Case History 1: SCSI Timeouts

- Problem Scenario: SCSI Timeouts, loss of access to Luns, Application interruption:
- Configuration: 3 clustered servers each with 6 HBA paths each path accessing 170 Luns mapped on all 6 storage Ports with a default Lun Queue Depth of 32, USP Port Tag count of 1,024
- Total of $180 \times 32 \times 3$ (servers) = **17,280** possible concurrent commands per Port and a high probability of reaching and exceeding the 1,024 concurrent commands supported on the USP on occasions.
- Resolution: Reduce the Lun Queue Depth and reduce the number of Luns on each Path.
- New Configuration: 3 sets of 60 Luns on pairs of Ports, Lun Queue Depth reduced to 6.
- Total of $60 \times 6 \times 3$ (servers) = **1,080** possible commands per Port and a much lower probability of reaching and exceeding 1,024 concurrent commands.

Case History 2: Lower than expected Performance on an AMS500

- Problem Scenario: disappointing IOPS rates but low utilization in the Storage:
- Configuration: AMS500, Qlogic HBA Target Queue Depth (Execution Throttle) set to 16. 12 RAID Groups (108 HDDs) were sharing the 16 available queue elements.
- Resolution: Increase the Execution Throttle to 256

Taking the risk ...

- The use of the formula $\#Tags/\#Luns$ to prevent Port Tag exhaustion assumes that all the Luns will be equally active at the same time and does not allow for Luns that are less utilized than others. If it is known that the workload distribution is **NOT** equal, and that some Luns may be busy while others as less busy, then it is possible to increase the Lun Queue Depth value indicated by the formula.

- The user needs to make a judgement over the likely levels of queuing in order to estimate the optimum over-specification of Lun Queue Depth.
- This is a situation where Little's Law can help estimate how close the level of queuing is actually getting to any limits.

Queue Depth, Target Mode and Clustering

- It can be difficult to define Lun Queue Depth in clustered environments where several machines have potential simultaneous access to the same Luns. The Lun Queue Depth formula may result in low Lun Queue Depth values if there are large numbers of Luns or servers.
- $\#tags / (\#luns \times \#servers)$
- Eliminating inactive Luns and servers from the calculation can help, but it may be impossible to do this accurately.
- Target Mode should be considered in these situations as this will eliminate the need to determine the number of active Luns, just the number of active servers.
- Where a lack of precision over the number of active Luns and servers can't be avoided then a retrospective analysis of Response Time, IOPS rate and resource utilization will indicate whether Queue Depth problems exist in a running system/cluster - See Little's Law

Storage Reporting of Queue Depth

- The AMS Performance Monitor (PFM) facility reports the average Lun Queue Depth (LU Tag Count). The USP does not report Queue Depth.

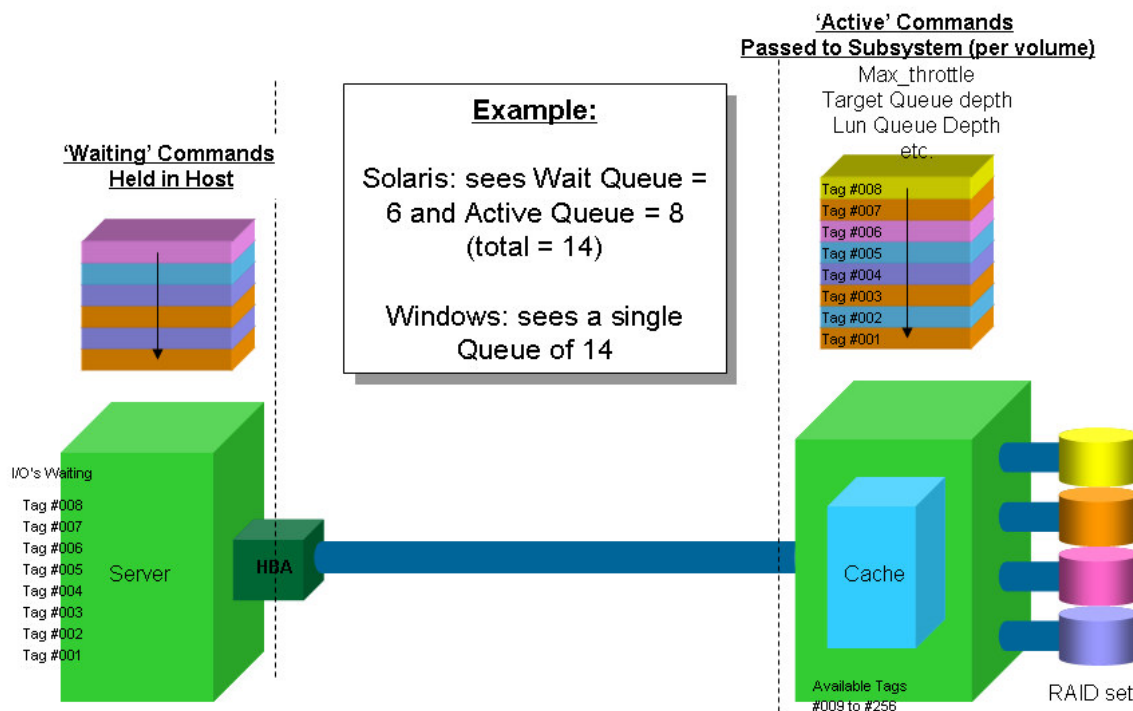
CTL	LU	Tag Count
0	0	0
0	1	0
0	30	0
0	31	0
0	32	0
0	33	0
0	34	11
0	35	0
0	36	12
0	37	0
0	38	11
0	39	16
0	40	0
0	41	15
0	42	0
0	43	0
0	44	0
0	45	0

“The AMS PFM Output File reports Lun queue lengths”

Monitoring Queue size

- Most Operating Systems measure Queue lengths. In some cases this is a single consolidated Queue measurement, and in others the Queue is split into the number of commands already accepted by the Storage and the number of commands waiting in software.
- Solaris IOSTAT reports the Active (**actv**) and Wait (**wait**) Queue lengths. Where the HBA Lun Queue Depth is defined this represents the largest size of the active Queue. The Wait Queue will only become non-zero and start to increase when the active queue is full.
- If the Wait queue becomes non-zero while the Active Queue is lower than the Lun Queue Depth limit then this can indicate momentary bursts of very high Queue depths or a starvation of Tags at the storage port.
- Note: Solaris does not report accurate values when the disk utilization is low. Disks need to be 5% busy before any high queue or Response levels should be investigated.

Monitoring Queue size, example of I/O Queue of 14 with HBA Lun Queue Depth = 8



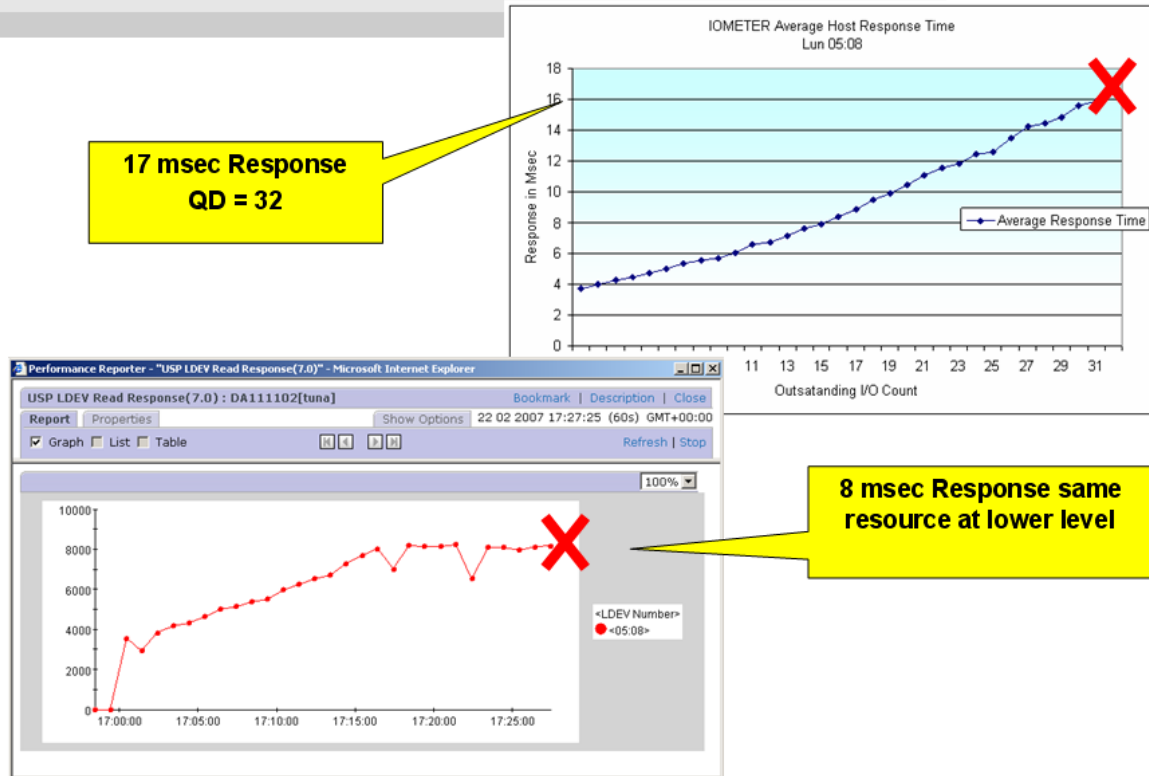
“Windows will see a single Queue but Solaris splits this into the active and wait Queue”

The importance of Differential Response

- In situations where the actual Queue Length is difficult to measure then Differential Response is the key method for determining whether a Queue Depth problem exists. These situations include VMware with multiple virtual machines accessing the same physical resources or External Storage where host mapped Luns do not have a 1:1 relationship with the external luns.
- Response Times can be measured for the same set of resources at different points in the storage architecture using HTnM, Storage Monitoring and O/S level monitoring tools.
- We expect Response Times to increase gradually as the resource is viewed at each new higher level. If moving to a higher level is accompanied by a jump in Response then it is likely that we have identified a Queuing problem.
- The actual reason for the problem will then need to be worked out along with any potential solution, but at least the source of the problem will have been identified.

- Little's Law can be used to calculate the likely actual Queue Depth values.

Example of Differential Response to show Queue Depth saturation



“Differential Response measurements can show where a Queuing problem exists”

Previous HDS Queue Depth Discussion documents

- Lun Queue Depth Benefits, 2003 – Glenn Brackenridge
- Storage System Solutions Performance Handbook, 2001 – John Webb
- Command-Tag Queueing, 1998 – Ken Wood

Queue “Jumping”



“Queue Jumping – a form of queue optimization”