Understanding iSCSI Digests: Accurately Evaluating the Cost and Risk of Disabling Digests

By Jerry Daugherty, Test Engineer, JDSU Medusa Labs

The iSCSI protocol plays a major role in facilitating the deployment of flexible data storage networks. Designed to run over TCP and, to a lesser extent, over Ethernet, iSCSI introduces another protocol layer to the data transfer process. Part of the iSCSI specification provides for a digest to detect errors that occur at this iSCSI layer. Processing the iSCSI digest is believed by many to so adversely affect performance that it is common practice to disable the iSCSI digest, instead relying upon TCP and/or Ethernet error detection mechanisms to ensure data integrity.

Disabling digests, however, can impact the reliability of data transfers by leaving the network vulnerable to errors that occur between protocol transitions. In the case of TCP, data integrity is also degraded as error detection is left to the simple TCP checksum rather than the more robust 32-bit CRC iSCSI digest. Such an approach can potentially result in undetected data corruption, possibly manifesting as database integrity failures, garbled text, or even corrupted file tables.

Performance does matter, and so enabling iSCSI digests is not a foregone conclusion, even in mission-critical applications, because of its potential impact on performance, especially at high utilization rates. However, disabling iSCSI digests should not be a quick decision either. Rather, network administrators will want to first determine the actual effect digest processing has on performance for a specific application, taking into account factors such as the importance of data, how much data is involved, and how often it is accessed. Armed with such information, storage administrators can more accurately evaluate the true risk/costs associated with disabling iSCSI digests.

The Cost of Errors

Errors in the SCSI protocol data unit (PDU) can manifest a variety of problems that result in data corruption. For example, if a bit error occurs in the data portion of the PDU during a read transaction, the application requesting the data will receive corrupted data. If the data error occurs during a write transaction, corrupted data will be written to disk while the application believes the write was accurate and successful.

Bit errors in the header portion of the PDU are more problematic because there is no trail to expose the corrupted data. If the bit error occurs during a read transaction, the application will receive corrupted data (i.e., data from the wrong part of the disk). Potentially this data will be processed and then written back, overwriting the original value. Alternatively, if the header error occurs during a write transaction, the original data will be unchanged while a "random" block of data will be overwritten with the update value.

In all these cases, without a digest in place there will be no indication that any data corruption has occurred. For certain applications, this could have far-reaching effects. Considering a banking application where data corruption could result in "lost" funds or inaccurate stock purchases with no indication or reason to suspect data corruption has taken place. Header corruption leads to deferred problems that might only surface months in the future once the corrupted block of data is finally accessed. Even when data corruption is noticed quickly, it could be extremely difficult to track down how and when the corruption actually occurred so that it be rectified and the original data restored.

Network administrators may offer any number of reasons why they don't employ a digest. One of the more common justifications is that performance is all-important, fed by the competitive temperament of the storage industry. Certainly latency is undesirable, but a myopic focus on performance can lead to compromises in data integrity, reliability, and system robustness. Put another way, moving corrupted data quickly is not as useful as moving data in a slightly slower but more reliable fashion.

Others cite that SCSI has its own error checking mechanisms. While it is true that back in the days of parallel SCSI there were error checking mechanism in place, serialized SCSI did not carry these over. As a consequence, serial SCSI will write whatever it receives – accurate or corrupted – to disk.

Digest Robustness

The iSCSI digest provides a robust mechanism for detecting errors. Based on a 32-bit CRC algorithm, developers have the flexibility to compute the CRC over the entire SCSI PDU or either just its header or data portion. For the highest reliability, the digest should include the entire PDU to prevent both location and data corruption errors. While CRC algorithms do not support data reconstruction, they can detect multi-bit errors and are effective even with jumbo frames.

Figure 1 shows the several layers encapsulating an iSCSI packet. Transactions begin with just the iSCSI portion (in green) and build out through TCP (red), IP (light blue), and Gigabit Ethernet (dark blue). When the iSCSI packet is encapsulated by TCP, a TCP checksum is created. Likewise, the Gigabit Ethernet header contains a CRC for the entire packet.

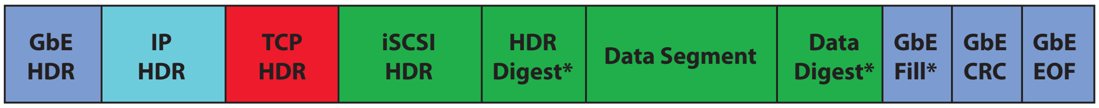

Figure 1: Shown are the several layers make up an iSCSI packet. Transactions begin with just the iSCSI portion (in green) and build out through TCP (red), IP (light blue), and Gigabit Ethernet (dark blue). While the TCP checksum and Gigabit Ethernet CRC can detect some bit errors in iSCSI headers and payloads, they do not detect errors generated at iSCSI protocol crossovers.

It is important to note that the Ethernet CRC does not guarantee accuracy at the iSCSI layer as lower layer protection mechanisms do not have the ability to completely protect data integrity at higher layers. To some degree, the Ethernet CRC will detect a subset of data corruption instances. For example, a bit error in the data portion of the iSCSI packet could be detected by the Ethernet CRC. However, corruption that occurs during the protocol crossover between Ethernet, TCP and iSCSI may not be detected.

Besides protocol crossover errors being outside the protection domain of the transport protocol, TCP uses only a simple 32-bit checksum, a significant degradation in robustness compared to the 32-bit CRC of an iSCSI digest. Checksums offer inferior error detection since 2 bit errors in the same packet can effectively cancel each other out, leaving both errors undetected.

The iSCSI protocol supports three levels of error recovery:

- Level 0: Any error results in the session being immediately dropped. The session must be started over by the application.
- Level 1: Rather than taking down the entire session, this level of error recovery initiates a simple re-transmission of the corrupted PDU in question. This process is effectively transparent to the SCSI layer.
- Level 2: For the most robust error recovery, this level initiates full connection recovery. While an entire I/O transaction must occur over the same connection, a session may have multiple connections per session (MCS). If a link fails, the I/O can be moved over to surviving connections in a mildly transparent manner.

Level 1 and Level 2 error recovery are more appropriate for mission-critical applications where a dropped session is highly undesirable. Level 1 comes at the cost of increasing HBA memory requirements because transactions must be tracked until they are completed. Level 2 introduces even more processing complexity and cost.

For this reason, as well reflecting the general lack of use of iSCSI digests, most targets offer only Level 0 error support. Level 0 technically doesn't recover from errors; its value comes in detecting errors and preventing data corruption from occurring. Given the potentially high cost of data errors, users would certainly rather deal with unexpected session loss than have their data corrupted. For many applications, Level 0 error recovery is sufficient.

Measuring Performance

Enabling iSCSI digests does affect performance. The question network administrators need to ask is whether the impact on performance is significant and if so, whether it is high enough to be worth risking data corruption.

The performance hit from implementing digests can be determined simply enough by turning on digests and measuring the real impact on the network and application performance. Typically the performance hit for digest processing ranges from 5% to 20%, depending upon the application, topology, and specific iSCSI implementation. This is a fairly wide range, so it's worth knowing what number applies to a specific network before making a decision to enable or disable digests. An example test was conducted for this white paper using a common setup of software initiator and hardware target (see Figure 2).

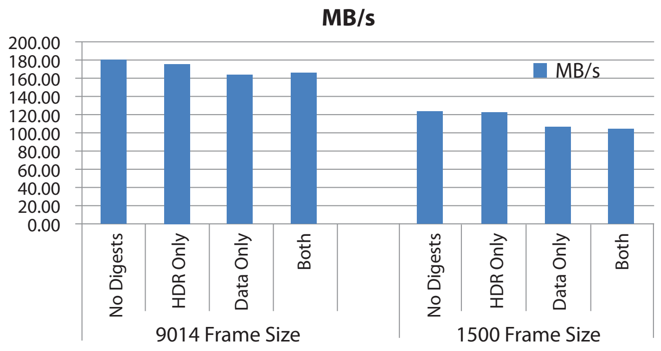

Figure 2: The effect of digests on performance can range from 5 to 20%, depending upon the application. Shown here are the results from a test bed comprised of an Intel S5000PAL server motherboard with (2) Intel Xeon 5070 dual-core processors at 3.46 Ghz and 4 GB RAM running Windows 2008 Server. All firmware and BIOS revisions were current from the manufacturers. The iSCSI client was the Microsoft iSCSI initiator connected to a Dell PowerVault MD3000i array. The MD3000i was populated with six 36Gb 10K RPM SAS drives. I/O was generated using the Medusa Labs Test Tools set for 50% read/write operation and 64k I/O size. Tests were allowed to run for 60 seconds, and the average transfer rate in MB/s was used. The identical test parameters were used with both 1500 byte and 9014 byte Maximum Transfer Units for comparison of the effect of jumbo frames on digest performance.

	9014 Frame size				1500 Frame size			
	No Digests	HDR only	DATA Only	Both	No Digests	HDR Only	DATA Only	Both
MB/s	179.92	176.24	163.92	166.04	123.91	122.77	107.79	106.21
Percentage difference		-2.08%	-9.76%	-8.36%		-0.93%	-14.95%	-16.66%

To verify data integrity requires a strict testing methodology since simply sending lots of data could fail to detect corrupted data. The test bed must confirm each write by reading the data back and verifying its accuracy. In addition, the data must contain signatures or patterns to confirm that it hasn't been accidentally overwritten. In fact, the entire drive should be verified at the end of the test to confirm that no data has been corrupted from header errors.

First run the test with digests turned off to capture a baseline performance measurement. Next run the test with digests turned on. In a broad sense, the difference is the cost in performance of using digests.

The worst-case for performance is when digest calculations are performed in software at both ends of a connection with no hardware offloading and a majority of transactions consist of single I/O commands. Digests implemented completely in software will experience a higher performance hit than digests accelerated with a hardware-based HBA with the iSCSI stack implemented in hardware. The performance hit will be substantially lower if either or both the target and initiator employ hardware-based digest processing.

If there is no significant difference in performance, digests can be enabled without cost or compromise of data integrity. If there is a substantial reduction in performance, a number of options are available for mitigating the hit:

Underutilization: Before deciding that a performance hit is too high, first determine whether the hit is actually worth reacting to. Consider a network that isn't operating at capacity (i.e., the network averages 60% utilization) and thus won't manifest any adverse effects even from a large digest hit. In this case, higher performance isn't what's needed, especially when data integrity is left vulnerable as a consequence.

Hardware-based processing: Typically, hardware-based targets and initiators perform digest calculations in hardware. Software-based targets and initiators, on the other hand, often implement digests in software. In general, software digests will experience a greater drop in performance than digests calculated using dedicated computational hardware. Performance on critical links can be improved by moving to hardware-based equipment.

I/O Load: I/O load also affects a digest's impact on performance. Applications that perform many single I/O operations will experience a higher performance hit. This happens because when only a single command is in play, additional commands cannot be initiated until the status of the current command is received. When multiple I/O operations are backed by deep queues, the system can mask the latency caused by waiting for the command status through concurrent processing other commands. I/O load is dependent upon the application and type of information typically being transferred. For example, database applications with many commands will be less likely to see an impact from digests than a single OS booting from a remote iSCSI drive.

Managing Risk

The question of whether to employ iSCSI digests comes down to evaluating the risk of not using them within the context of a specific application. Higher performance without digests reduces infrastructure cost, but this cost must be balanced against the potential losses of suffering data corruption, whether they are financial losses, missed opportunities, or the cost to correct the errors.

In general, the types of errors detected by iSCSI digests are fairly unlikely. These errors are usually the result of programming errors, often in the stack implementation. One might assume that manufacturers have thoroughly tested equipment before releasing it to market; however, these OEMs face their own market, performance, and risk considerations. Comprehensive testing is expensive, time consuming, and extremely difficult to verify across all possible applications and usage scenarios. Companies support hardware with software updates exactly for the reason of adjusting for any such hard to evaluate exceptions and corner cases they might have missed.

The problem with evaluating the risk associated with iSCSI digests is that there is no way to know that data has been corrupted at the time. In addition, when errors do manifest, they are easy enough to blame on some other cause that appears intermittent. Applications may also recover from errors transparently, eliminating any "paper trail" that could reveal the actual cause. As a result, it is difficult to accurately access the frequency of errors that could have been detected and prevented had an iSCSI digest been in use.

The cost of undetected data corruption is only slightly easier to evaluate. Bad data could crash a database by causing it to fail an integrity check. If the error that caused the integrity check to fail occurred even just a few days ago, it could be an onerous task to reconstruct the database to simply find the error, never mind the cause. Loss of business or productivity, in terms of cost, needs to be estimated on an application-by-application basis.

Performance is critical to daily operations, but in the end what matters most is the integrity of the data. The question to ask is whether any gains in performance are worth the risk of corrupted data. Skipping iSCSI digest processing means the network must rely on other mechanisms to detect errors and corrupted data that are not as robust nor designed to detect iSCSI protocol crossover errors. While a perfect network may be impossible to achieve, there are inherent costs in risking data corruption when these situations could be avoided. Rather than be blinded by a focus on performance, consider the data integrity angle so as to make the most informed decisions for a particular network. While there may not be a problem, it is prudent to confirm that this is the case rather than discover the true cost of data corruption. No amount of performance should allow for the writing of corrupt data.