

White Paper: The Unique Challenges of Testing Unified Storage

Alan Newman
VP Marketing and Founder

ABSTRACT

Testing unified storage presents a number of challenges for vendors, storage service providers, and enterprises that are validating storage products or storage infrastructure. Certain issues related to two basic test types—functional testing and load testing—are more complex in a unified environment. In addition, ever-increasing numbers of storage protocols and growing storage demands put significant pressure on under-resourced engineering teams.

This paper describes the unique challenges of performing functional and load tests on a unified storage infrastructure. Testing concurrent access to the same file and the effects of different mixes of protocol traffic across storage types is exceedingly difficult. Mastering such complexity requires purpose-built tools as well as advanced testing skills and protocol understanding.

Target Readership

This paper will be of interest to anyone with high-level company goals such as accelerating time to market, developing consistent testing practices, increasing quality, and ensuring competitive advantage.

It is also intended for technologists and managers who design, implement, and/or maintain significant unified storage infrastructure. Companies with this infrastructure include equipment vendors, storage service providers, and a variety of enterprises.

- *Storage Vendors: for QA directors, and engineering directors and managers*
- *Storage Service Providers: for storage architects, and QA directors and managers*
- *Enterprises: for IT directors and managers, and storage architects*

Table of Contents

Introduction	3
About unified storage.....	3
Functional Testing Overview	3
Multiple users.....	4
Multiple protocols	4
Functional Testing Challenges.....	4
Metadata mapping with multi-user and multi-protocol access.....	4
Maintaining data integrity with operations on the same file.....	4
Understanding and testing advanced features of a protocol.....	5
Functional Testing Answers.....	5
Load Testing Overview	6
Load Testing Challenges	7
Mixing scenarios and scaling a single protocol.....	7
Testing multiple protocols simultaneously.....	8
Comparing performances with uniformly detailed reporting.....	9
Load Testing Answers	9
Expertise Challenges.....	9
Protocol expertise.....	10
Testing expertise.....	10
Expertise Answers.....	11
Conclusion	11

Introduction

In this paper, we drill into the technical challenges of testing across protocols and across storage types, focusing on functional testing, and load or performance testing.

There are certainly many other types of tests including sanity, regression, and reliability, which for the purposes of this paper will be considered derivatives of functional and load tests. If you're smoke or sanity testing, for example, you'll typically run a suite of functional tests and a quick performance test. Other testing needs are usually met by employing hybrids or combinations of the two basic test types.

About unified storage

Today, IP storage environments typically support a number of data types: files or unstructured data, blocks or structured data, and objects. As a refresher, file level protocols include CIFS/SMB and NFS. Block protocols include iSCSI, FCoE, and Fibre Channel. Object protocols include HTTP. Simultaneously handling files, blocks, and/or objects presents unique testing challenges both across data types and among protocols. Plus, new protocols are being developed all the time, which adds to the complexity of the issues discussed in this paper.

Trends indicate that unified storage is increasingly sought as a solution for simplifying storage environments. This is from the user perspective. From the testing perspective, all the complexity that's now delivered as one system is still inherently intricate, and perhaps more so given the broader storage needs these systems address from one unified platform.

Functional Testing Overview

Functional testing for storage includes testing a system's ability to read and write data, and to implement metadata operations. Functional testing is characterized by operations performed sequentially on data or metadata within a single type of storage [Figure 1]. Testing becomes more challenging in multi-user and multi-protocol environments.

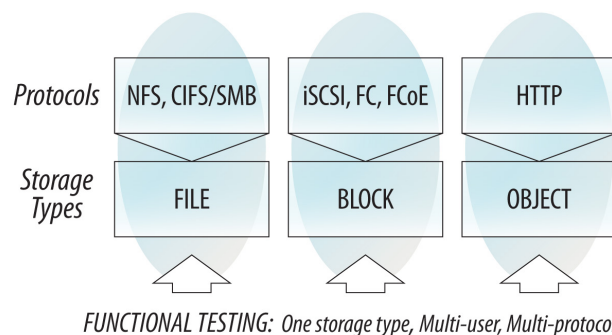


Figure 1: Functional testing among storage types and protocols.

Multiple users

Currently, a company testing the functionality of its products or storage infrastructure simulates multiple users either manually or using homegrown tools.

When two users are trying to access the same file to read, write, or delete, nothing prevents one user from impeding the other unless locking is present. But most storage products perform operations in parallel, so what if your storage system acts on both requests simultaneously? As you would expect, when you increase the number of users, the likelihood of conflict increases.

Multiple protocols

Multiple protocols can simultaneously operate on a single instance of data or metadata along with multiple users. Synchronizing access across protocols is essential.

Typically, a test team will write a test tool to test CIFS/SMB operations and another test tool to test NFS operations. These tools have different user interfaces and reporting mechanisms, and use different attributes that don't match. For example, does locking a file in SMB lock the same byte range in NFS? Are CIFS/SMB metadata operations mapped correctly to NFS metadata?

Functional Testing Challenges

Significant issues with functional testing in a multi-protocol environment can reduce effectiveness or render impossible complex testing at the file level. Three challenges are:

- Metadata mapping with multi-user and multi-protocol access
- Maintaining data integrity with simultaneous operations on the same file
- Understanding and testing advanced features of a protocol

Metadata mapping with multi-user and multi-protocol access

Metadata provides information about files and directories including names, access restrictions, and date/time stamps. Metadata is presented to the user in slightly different ways per protocol.

For example, Linux (NFS) has more sophisticated access controls than Windows (CIFS/SMB). Changes to these access controls must be mapped correctly between protocols even though they differ. Say that you want to test simultaneous access to files via both CIFS/SMB and NFS. You must ensure that the inconsistent file permissions map as needed for users of each protocol, that access rights, file locking, etc. are properly handled in all cases.

Maintaining data integrity with operations on the same file

When multiple users write to a file, data integrity is ensured using locking. Locking is inherently complicated when only one protocol is used, but when multiple users with various protocols are writing to the same file, the challenge is compounded. Data corruption results when requests collide and a client writes over a portion of a file locked by another user using a different protocol. The lock must be mapped correctly between protocols.

Another example is about granting to a group of users. In Unix/Linux NFS implementations, the Group Identifier or GID, is used to manage access. In Windows CIFS/SMB implementations group identification is maintained by the Security Identifier, or SID. Software independent of the underlying access methods must map these incompatible identifiers to ensure proper access is maintained for both NFS and CIFS/SMB clients.

"If your customers are pushing the envelope, you'd better be equipped to test for it."

Understanding and testing advanced features of a protocol

Most protocols, especially new ones and new versions such as NFSv4.1 and SMB 2.1, have advanced features that need to be tested. As discussed above, testing file locking is critical, and understanding other features such as asynchronous client notification of a state change is essential for successful functional testing.

An example of an advanced feature is the Oplock Break. The CIFS/SMB protocol uses a mechanism called an Opportunistic Lock, or Oplock, to grant temporary, exclusive access to a file while guaranteeing that the file has not changed. The benefit is that Oplocks reduce network traffic by allowing a client to read from a known version of a file or write to a file without constantly sending updates to the server. CIFS/SMB file servers use an Oplock Break to signal a client that a lock is being revoked when another client requires access to a file. The Oplock Break controls how a file lock is reclaimed by a server and what locking, if any, remains after the file lock is broken.

Another advanced feature is Lease, implemented in SMB 2.1. Lease is similar to Oplock, but offers greater caching flexibility than Oplock and is especially useful in high-latency networks.

Being able to test this type of functionality is essential in a multi-user, multi-protocol environment. It's exceedingly difficult to synchronize two or more users doing such things but you need to test your device's ability to handle simultaneous and conflicting requests gracefully and without data corruption.

Functional Testing Answers

As the number of users and protocols accessing the same data increases, homegrown tools can't handle functional testing in any practical way. Because this level of testing is so difficult, companies often don't do it. In-house tools typically can't do sub-millisecond synchronization—period.

The solution requires a single tool that can synchronize traffic across protocols, simulating multiple users making near-simultaneous requests. Systems such as those offered by SwiftTest include purpose-built software to specifically enable this level of functional testing.

SwiftTest enables you to test mapping of metadata across protocols including file and directory names, access restrictions, locks, and date/time stamps no matter how they're presented to the user. Testing conflicting metadata change requests is enabled by one tool that can simulate multiple users accessing the same file using multiple protocols.

We have shown that synchronizing two users that have their own TCP connection and that are being generated by different test programs is a complex software programming problem. SwiftTest provides a graphical user interface that gives you full control with drag and drop functionality to quickly create complex scenarios. You can synchronize commands for multiple users and protocols, all from one place.

Synchronizing commands is accomplished using SwiftTest Events [Figure 3]. One user waits for the second user to give a quick nod to proceed. With these super fast wait and send instructions, two users can perform commands within microseconds of each other, and will appear as essentially simultaneous to the storage equipment. SwiftTest Events enables QA engineers to test simultaneous data and metadata operations with single or multiple protocols.

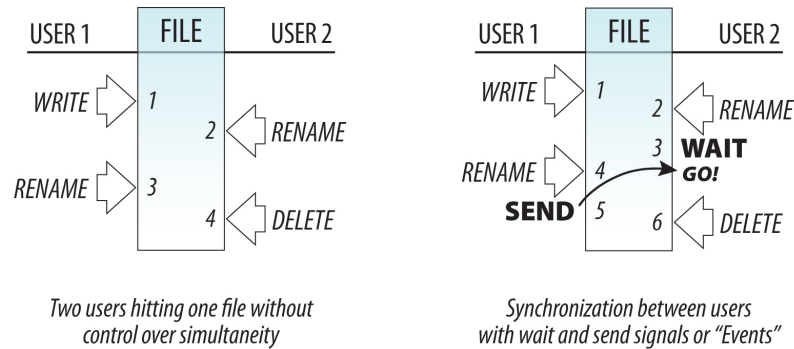


Figure 3: Flow control using SwiftTest Events enables near-simultaneous operations.

Load Testing Overview

Load testing assesses a system’s capability to sustain extreme load, while performance testing measures a system’s response time at varying loads. In this paper we refer to both as load testing [Figure 4].

A typical test scenario is to simulate tens or hundreds of thousands of simultaneous TCP connections—customers hitting your storage systems concurrently from across the globe using a variety of protocols. Load testing is ubiquitous among end users and QA teams in storage equipment and storage service provider environments, requiring extreme scalability with detailed visibility into performance data.

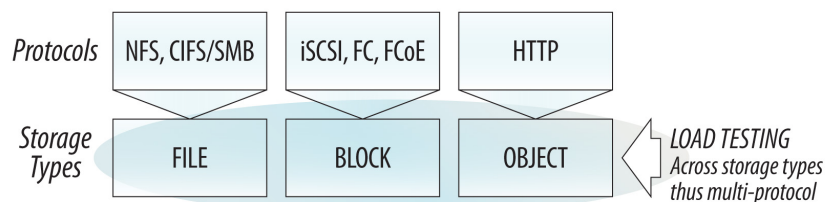


Figure 4: Load testing among storage types and protocols.

Load Testing Challenges

Since unified storage systems support multiple protocols simultaneously, it is important to test them that way. Issues with load testing in a unified, multi-protocol storage environment tend to be interrelated, stemming from inherent variation among protocols. Three challenges are:

- Mixing scenarios and scaling a single protocol
- Testing multiple protocols simultaneously
- Comparing performance with consistent, detailed reporting

Mixing scenarios and scaling a single protocol

Today, generating enough traffic to adequately load systems under test often requires spinning up virtual machines and getting them to run consistently. The challenge is to scale up to massive numbers of operations while collecting meaningful and repeatable performance measurements. You also need the flexibility to set up multiple traffic patterns.

For a single protocol, consider the elements tested in a typical load test. You have TCP connections, bandwidth, and metadata operations. How many connections can you keep open? How many bytes can you squeeze through? How many metadata operations can you perform while squeezing those bytes through those connections?

The area of interaction among these three elements can be thought of as your “test envelope” with the points of the triangle representing the extremes that you need to test [Figure 5]. Typical usage patterns fall somewhere in the middle—a mix of the three.

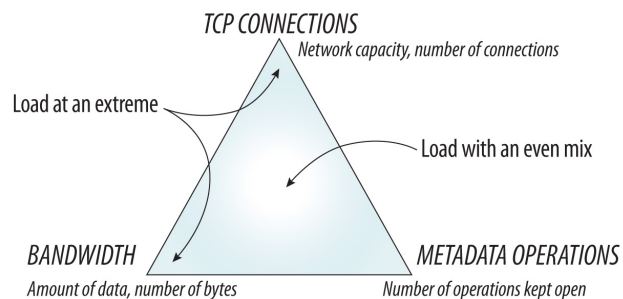


Figure 5: Test envelope for a single protocol. The middle signifies a mix of usage, the points represent extreme load in one aspect or another.

The testing challenges are different depending on where in the test envelope you focus. The extremes of the points and the middle of the triangle are difficult to test for different reasons.

Testing operations scalability

Your product or infrastructure needs to be strong on all three points—connections, throughput, and metadata operations—but it's challenging to scale operations to the necessary extremes for each, typically beyond the capabilities of homegrown tools:

- Scaling requires many more physical servers or virtual machines to run tests.
- Consistent repeatability is practically impossible when setup is so difficult.

Testing scenario variety

In the center of the envelope, you need flexibility and control of user scenarios. For example, you may need to test bandwidth-heavy imaging operations, transaction-oriented databases, or a burst of users that taxes your connections. Dialing individual parameters up and down to test these capabilities with an adjustable mix is extremely complex. Homegrown tools typically lack such flexibility:

- Handling complexity of various data types and network activity is hard, even for a single protocol.
- Changing parameters and attributes easily for varied scenarios is required so you can quickly re-run tests.

Testing multiple protocols simultaneously

Add multiple protocols, and you have a new set of challenges. Not only do you need the flexibility to change the mix in your scenario for metadata, bandwidth, or connection requirements, you need to do so for different protocols. For example, you may need to set up a manufacturing industry profile, and within that you have Exchange servers using iSCSI, marketing using CIFS when writing Word docs, and engineering teams storing big datasets with NFS.

Take our test envelope triangle and add depth [Figure 6]. This illustrates the compounding of the complexity as you add protocols.

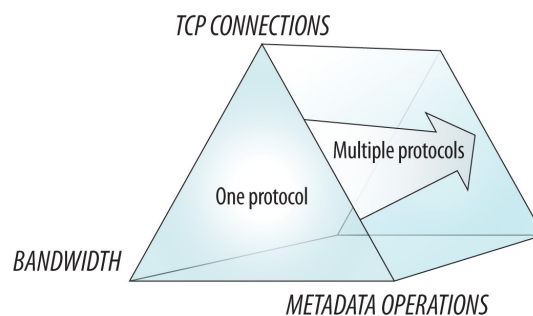


Figure 6: Complexity volume showing depth of difficulty when you go from single to multiple protocols.

Here's how the challenge is volumetrically increased. Say you start testing performance with lots of CIFS/SMB file Reads and an equal number of NFS file Reads. What happens to NFS performance when you scale up to twice the CIFS/SMB Reads? Regarding users, do you get more throughput with 200 NFS users plus 100 CIFS users, or the reverse?

Currently it's very laborious to answer these questions. If scaling up using one homegrown tool is hard, with more it becomes prohibitive. It's difficult or impossible to compare the performance data from two different test tools—today, testers are likely to skip this level of testing.

Comparing performance with consistent, detailed reporting

Reporting and measurement of the interaction among protocols is often a barrier to effective testing. How do you know if one protocol uses more CPU than another? How do you discover which commands are the slow ones—the ones that you might be able to accelerate to improve overall performance?

Answering these questions is exceedingly challenging, requiring a persistent, compatible measurement system that both creates apples-to-apples comparisons across protocols and creates meaningful reports that show results correlated to each protocol, granular down to the command.

Load Testing Answers

Load test issues boil down to scalability of a single protocol, and measuring the impact of multiple protocols running simultaneously. The optimal solution is a single test tool that provides flexible traffic patterns and consistent reporting across all protocols.

Ideally, it provides both power and flexibility. You are likely to have testing objectives in a certain area of the testing envelope depending on user scenario. You may need to test iSCSI for Exchange, storage backup provider capabilities, moving of large datasets in a render farm, or other customer case. You'll want a specific, dialed-in combination of bandwidth, connections, and metadata so you can test your ideal scenario-plus-protocol mix.

Systems such as those offered by SwiftTest include purpose-built features that provide a clear view into the complexity volume, so you can see where one protocol's activity happens relative to another.

Say you're setting up a custom test suite for a customer scenario, you'll want to test certain usage profiles individually first. Then test all together to see the interaction, scaling up as much as you wish. Just like your customer would do, using your product.

With SwiftTest, setting up a perfect mix is straightforward and visual, thanks to a UI that features dialing up and down of protocols—you can weight them, say, with 20% of one and 30% of the other—all from within one tool. You can also compare performance with uniformly detailed reporting and see the outcomes of every usage profile so you can compare the results meaningfully—and take action quickly.

Expertise Challenges

The value of knowledge is increasing at an extraordinary rate. Testing storage systems requires a high level of expertise and this is never more critical than today. Systems are scaling to petabytes and beyond, with complexities scaling to match. Currently, we find a shortage of talent with the specialized

"My critical infrastructure needs to work as expected every time—under load, and under stress."

"It's not news to anyone that the explosion of business-critical data is here. Our increasing dependency on reliable, anytime access to it, advanced storage expertise pops up to top priority."

engineering skills required to run unified, multi-protocol storage testing. Whether for enterprises, storage service providers or vendors, the requirements for test expertise has jumped to a new level.

What's the big deal with testing? Simple tests are easy to run. But as described above, that's no longer enough. To maintain competitiveness, storage vendors and service providers must simulate user scenarios that are extremely complex and difficult to plan, write, and execute.

Test teams find they need expertise in two areas. They need technical depth and a high level of understanding of testing processes.

Protocol expertise

Critical technical knowledge is difficult to attain as storage protocols become more complex. New versions of existing protocols are emerging—and, of course, the old ones don't go away. For example, NFS 4.1 separates the metadata path from the data path. SMB2.1 adds asynchronous notifications of changes to files and directories. These new features require understanding.

Terabytes have become petabytes, and will soon become exabytes. Never before have companies tested at this scale, and with multi-protocol access, testing requirements are not only huge, they're multi-dimensional. So it's now essential to employ talent capable of handling depth of features across multiple protocols... at scale.

Testing expertise

Given the enormous number of potential tests that can be executed, selecting an effective set that can be executed in a reasonable amount of time requires significant expertise. For example, load tests generate multi-protocol traffic with a mix of open TCP connections, R/W throughput, and metadata operations. Can you figure out which protocol is degrading your performance? Or is it TCP connection processing that is dragging it down? Which test should you run next to isolate the problem?

Consider the steps in the testing process:

Selection and Planning

Selecting the right test for the job is paramount. Say you need to develop a suite of tests for SMB2.1. Given you understand the protocol, you must also have the experience and judgment to select the right test, such as validation of a Change Notify command. Out of many possible permutations, what's the correct set of tests that will be sufficient to validate this complicated feature?

Implementation and Execution

With validation of Change Notify for SMB2.1, for example, writing such a program is complicated, and takes a senior engineer to create a test tool for automating the commands so you can execute quickly and repeatedly.

Analysis and Reporting

Once you have your reports, you need deep analysis and thorough understanding of the statistical data to identify bottlenecks and accurately triangulate your way to a problem. Given adequately detailed reporting, it's then up to you to figure out how to properly iterate after isolating your problem.

Expertise Answers

So what to do? Leverage industry expertise. The key to beefing up the capabilities of your team is to engage a company with history and reach into a variety of environments and testing cultures.

SwiftTest offers both technical and process expertise with the added advantage of deep experience and an active, current practice across a range of environments. Such a combination is invaluable in two ways:

1. SwiftTest specializes in storage testing with many years experience so the level of technical expertise is beyond what most labs sustain. Such focus keeps the company ahead of the curve so that skills with protocol features and testing strategies are readily passed on.
2. Working in a variety of environments, SwiftTest has ongoing understanding of the unique needs of many different testing cultures. Breadth of knowledge enables SwiftTest to be an aggregator storage testing best practices, bringing something extra to the table, whether the need is around functional, or load testing, or general test planning and lifecycle management.

If you have neither bodies nor expertise, you can accomplish the level of testing you need by taking advantage of services offered in the marketplace. It's apparent that increasingly, storage testing services and training will be a regular part of a typical QA lab's business objectives and planning.

Conclusion

You've seen how the challenges in testing unified storage will only increase over the coming years. The need for leading-edge test technologies and expertise is growing like mad. And merely adequate testing of your systems isn't enough—you no doubt wish to test comprehensively and absolutely.

This paper has described how functional testing and load testing can become prohibitively complicated with multiple users and protocols in a unified storage environment. In addition, these tasks require knowledge of multiple protocols and advanced testing skills—not easily found in the talent pool today. Such difficulties can represent a large budgetary and general resource liability for many companies.

These complexities and barriers can be managed and overcome by employing a single test tool that enables multi-protocol testing across varied types of storage. The requirements for successful implementation of a such system are:

- Single user interface for configuring and running tests.
- Support for complex scenarios with full flexibility of the mix.
- Synchronized command lists for simultaneity across multiple users and protocols.
- Full automation framework for quick iteration.
- Detailed, consistent, and meaningful reporting for multiple protocols.

The predicted continuation of exponential growth of storage demands will no doubt be accompanied by much clamor for faster networks, greater bandwidth, and cloud-scale capacities and services. With the right testing solution you can stay ahead of the game, get your products to market faster, and ensure that your infrastructure perfectly meets your needs.