

The Challenges of Testing Unified Storage

Peter Murray
SwiftTest

pmurray@swifttest.com

Unified storage is now the way of the world.

Why challenging?

Because unified storage means multiple protocols and traditional test tools can't handle apples-to-oranges environments.

The problem promises to only get more complex and testing more critical as storage requirements grow.

What is Unified Storage?

- ❑ Any combination of storage types:

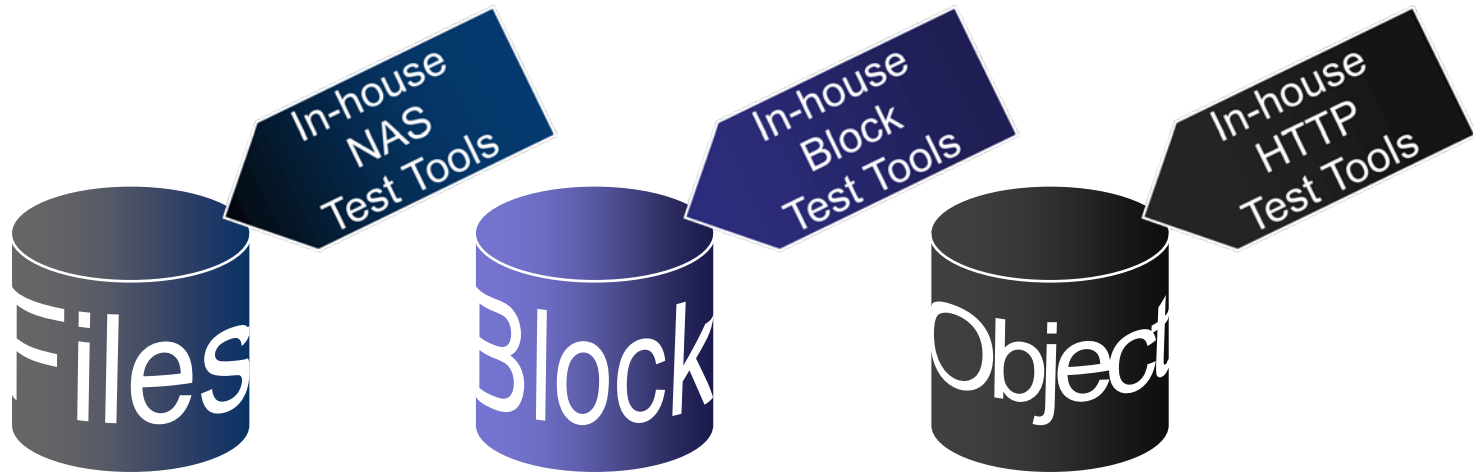
- ❑ Files (NFS, SMB/CIFS)
- ❑ Blocks (FC, FCoE , iSCSI)
- ❑ Objects (HTTP)



- ❑ Supports varieties of data types, multiple protocols
- ❑ Customers **think** that it's a simplified environment
- ❑ Testers **know** that inherent complexity is now just delivered in one system

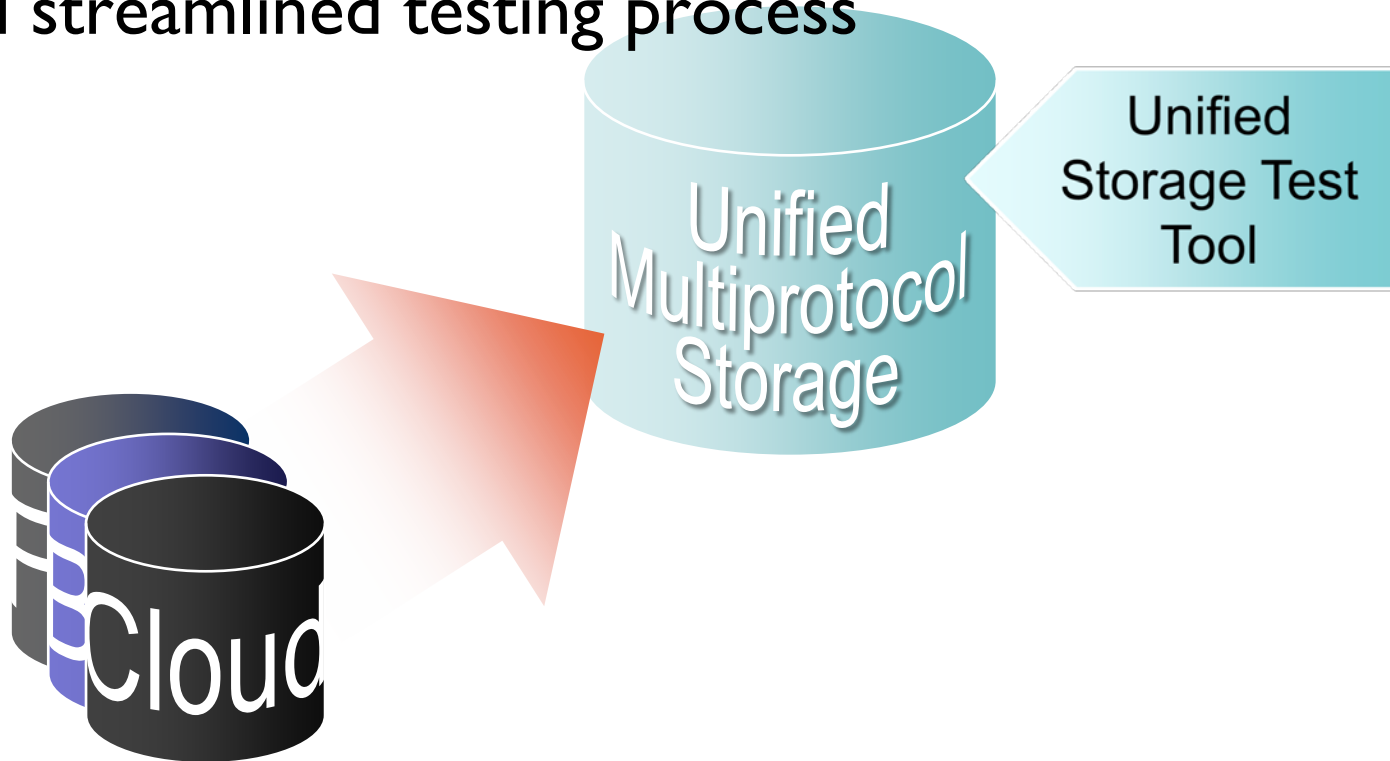
Testing Traditional Systems

- ❑ Protocol silos
- ❑ In-house, single-protocol test tools



Testing Unified Systems

- ❑ Need multi-protocol test tools
- ❑ Need streamlined testing process



Unified Storage Testing is a Must

- ❑ Only way to understand system-level behavior
- ❑ Only way to understand protocol interactions

**Gotta do it (everyone else is!)
and here's how...**

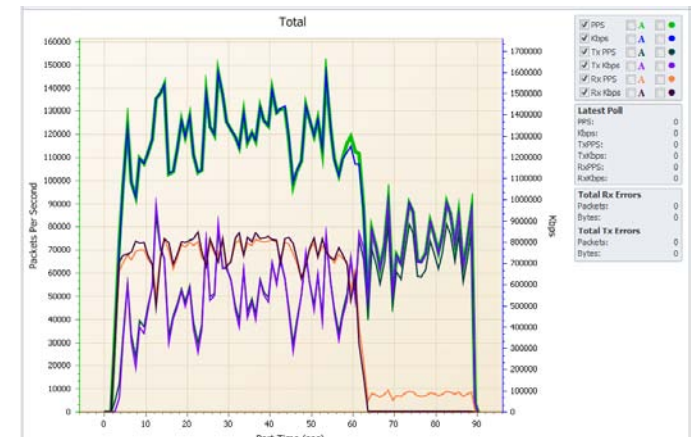
Step 1: Before you Start

Use a clean test bed

- ❑ Configure adequate IP address space
- ❑ Minimize network devices
- ❑ Avoid shared links
- ❑ Avoid corporate domains / controllers
- ❑ Use robust authentication
- ❑ Test using access rights
- ❑ Be careful with VM targets

Step 2: Pick a protocol

- ❑ Start by testing one protocol alone
- ❑ Helps ensure correct protocol implementation
 - ❑ Ensures basic functionality of a protocol
 - ❑ Measures performance:
 - ❑ Under normal conditions
 - ❑ Under stress
- ❑ Retain results for comparison



Test Elements

- ❑ Protocol functionality
- ❑ Common command combinations
- ❑ Concurrency or collisions
- ❑ Arbitrary combinations of commands
- ❑ Authentication
- ❑ Capacity
- ❑ Stress

Protocol Command Functions

- ❑ Test each element of a command
 - ❑ Proper parameter function?
 - ❑ Flags function properly?
 - ❑ Including error codes?
- ❑ Test command functionality
 - ❑ Does the command do what it should?
- ❑ Test command dependencies
 - ❑ Do other commands interoperate properly?

#	Protocol	Name	Name	Value
1	SMB	Open SMB TCP Connection	Input	Default
2	SMB	Negotiate	Connection Handle	
3	SMB	Session Setup	Diabets	
4	SMB	Tree Connect	SMB 2.001	False
5	SWT	Wait for Event	NT LM 0.12	True
6	SMB	Create Or Open File	LANMAN2.1	False
7	SWT	Raise Event	DOS LANMAN2.1	True
8	SMB	File Write	Samba	False
9	SMB	File Close	Out: SMB_COM_WRITE_ANDX Request (LANMAN 1.0)	True
10	SMB	Tree Disconnect	DOS LVE 2000Z	False
11	SMB	Session Logoff	Windows for Workgroup 3.1a	True
12	SMB	Close SMB TCP Connection	LANMAN1.0	True
			MICROSOFT NETWORKS 3.0	False
			MICROSOFT NETWORKS 1.03	False
			PCLAN1.0	False
			NTVX 1.1	False
			PC NETWORK PROGRAM 1.0	True
			Flags	
			Packet Signing	Disabled
			Caseless Strings	True
			Canonical Strings	True
			Unicode Strings	False
			NTLM Extended Security	Disabled
			Response Handlers	

Command Combinations

□ Protocol coverage test groups:

- Connection processing
- Metadata Operations
- Navigation
- I/O

□ Multiple tests per group

- Single User (functional)
- Multiple Users (scaling)
- Single operation (file, file size, file flags, etc.)
- Multiple operations (scaling file sizes, numbers, flags, etc.)

#	Protocol	Name
1	PMAPv2	Open Portmapper TCP Con...
2	PMAPv2	Null
3	PMAPv2	Get Program Port
4	MNTv3	Open Mount TCP Connection
5	MNTv3	Null
6	MNTv3	Mount Directory
7	PMAPv2	Get Program Port
8	NFSv3	Open NFS TCP Connection
9	PMAPv2	Close Portmapper TCP Con...
10	MNTv3	Close Mount TCP Connection
11	NFSv3	Null
12	NFSv3	Access
13	NFSv3	Create File
14	NFSv3	Lookup
15	NFSv3	Write File
16	NFSv3	Commit File
17	NFSv3	Read File
18	NFSv3	Remove File
19	NFSv3	Lookup
20	NFSv3	Close NFS TCP Connection

Concurrency or Collision Testing

- ❑ Requires near-simultaneous resolution (<1 millisecond)
- ❑ Determine what happens when protocols interact
- ❑ Force race conditions
- ❑ Find conditions that cause data loss
- ❑ Determine which combinations cause problems

35	6.033502	172.16.240.1	172.16.1.23	SMB	166 NT Create AndX Request, FID: 0x0003, Path: FILE01.TEST
36	6.033659	172.16.1.23	172.16.240.1	SMB	161 NT Create AndX Response, FID: 0x0003
81	6.033961	172.16.240.1	172.16.1.23	SMB	1382 Write AndX Request, FID: 0x0003, 65500 bytes at offset 0
82	6.033970	172.16.240.2	172.16.1.23	SMB	99 Close Request, FID: 0x000d
106	6.034596	172.16.1.23	172.16.240.2	SMB	93 Close Response, FID: 0x000d

Trace showing concurrency with just 9 microsecond delay between Write and Close

Arbitrary Combinations

- ❑ “What if” scenarios can show how
 - ❑ Applications may unexpectedly mix commands
 - ❑ E.G. compound commands
 - ❑ Devices may alter protocol streams
 - ❑ E.G. WAN acceleration or file virtualization
- ❑ Prevents data corruption
- ❑ Ensures robustness

Authentication

□ NTLM

- Server-based
- Domain-based

□ Kerberos

- Ensure adequate authentication performance
- Controllers may be performance bottleneck
 - Controller capacity may be inadequate
 - Communication with the controller may be slow

#	Protocol	Name	Name	Value
1	Kerberos	Open Kerberos connection	Input	
2	Kerberos	AS-REQ	Connection Handle	Default
3	Kerberos	TGS-REQ	Client principal	= @UP(0,A) + @STRING(@STORAGE.QAD)
4	Kerberos	Close Kerberos Connection	Password	= @UP(0,B)
5	SMB	Open SMB TCP Connection	Service principal	krbtgt/STORAGE.QAD@STORAGE.QAD
6	SMB	Negotiate	Proxiabile	False
7	SMB	Session Setup	Renewable	False
8	SMB	Tree Connect	Forwardable	False
9	SMB	Create Or Open File	Anonymous	False
10	SMB	File Write	Output	
11	SMB	File Close	Output Handle	2: Kerberos TGT
12	SMB	Tree Disconnect		
13	SMB	Session Logoff		

Combine commands to show real workloads

- ❑ Time-dependent actions like morning login
 - ❑ How does authentication affect performance?
- ❑ Metadata combined with I/O operations
 - ❑ How do lookups affect throughput?
- ❑ Capacity testing
 - ❑ What is the command-level performance at peak load?
 - ❑ What happens beyond peak load?
 - ❑ How many clients per server?
 - ❑ What is the capacity of a server vs. a cluster?

Step 3: Test Protocol Combinations

OK, so on to unified testing!

- ❑ Now, test:
 - ❑ Multiple versions of the same protocol
 - ❑ One version each of two protocols
 - ❑ Multiple versions of multiple protocols

- ❑ Use consistent methodology...

- ❑ Use the same methods as single-protocol tests
 - ❑ Start with individual commands
 - ❑ Combine commands
 - ❑ Test with authentication
 - ❑ Test with collisions
 - ❑ Test real-world scenarios

- ❑ Compare with single-protocol test results
 - ❑ Functional impacts?
 - ❑ Performance impacts?

Testing Tips Summary

- ❑ Test protocols individually and with consistency
- ❑ Test with real-world elements
 - ❑ Authentication, collisions, arbitrary combinations
- ❑ Test with real-world scenarios
 - ❑ Daily operations, storms
- ❑ Test protocol combinations

And Why?

- ❑ Increase quality
- ❑ Ensure highest performance
- ❑ Prevent data loss or corruption
- ❑ Contain costs
- ❑ Decrease time to market

- ❑ Testing unified storage is complex
- ❑ Individual protocols must be tested:
 - ❑ Thoroughly
 - ❑ Repeatably
- ❑ Protocol combinations characterize the full system
 - ❑ Performance variations
 - ❑ Potential bottlenecks
- ❑ A single test platform is an advantage

Thank You!

Questions?