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?
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.)
Concurrent 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

Trace showing concurrency with just 9 picosecond 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
Real-World Simulation

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...
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
Conclusion

- 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?