Accelerating SSD & System Development with Test Automation

Andy Tomlin

andy.tomlin@devicepros.net

Testing challenge includes wide and deep problems

Wide

- Problem created by large surface area to test:
- Interface & protocol testing - large array of command and interactions between outside world and the device under test
- Don't go it alone, leverage as much as possible - 3rd party tools and labs

Deep

- Read and write, data related test
- Small surface area, but very deep due to enormous state space
- Requires white and black box approaches

For speed of development, Test Automation is essential to solve both these challenges

Challenge of bring Flash based storage products to market

Mapping systems:

- Data Base, Filesystem, FTL all create indirections between logical and physical locations of data
- Depending on requirements this can be simple (L2P in Client SSD) to complex (COW B-Trees in System)
- Mapping is used to provide an array of interesting features: Linear mapping, Volume management, Versioning (Snapshots & Clones), Thin provisioning, Dedup, Compression

All indirection systems end up creating two systems:

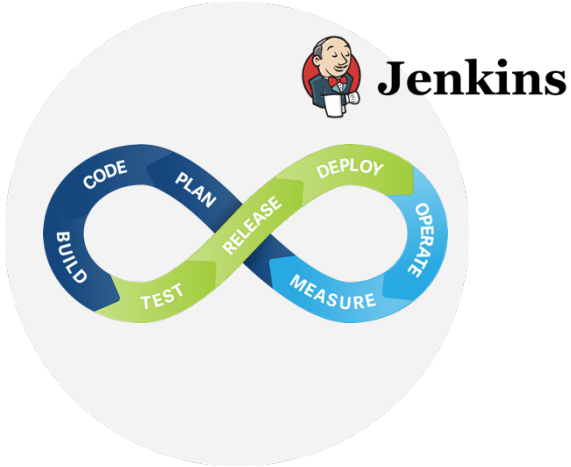
- System for user data, System for meta data
- These system must always be in synchronization from user perspective to function correctly

These two systems create development and test complexity:

- Solving power cycling challenge
- Maturing datapath

Development phase

Testing to Requirements



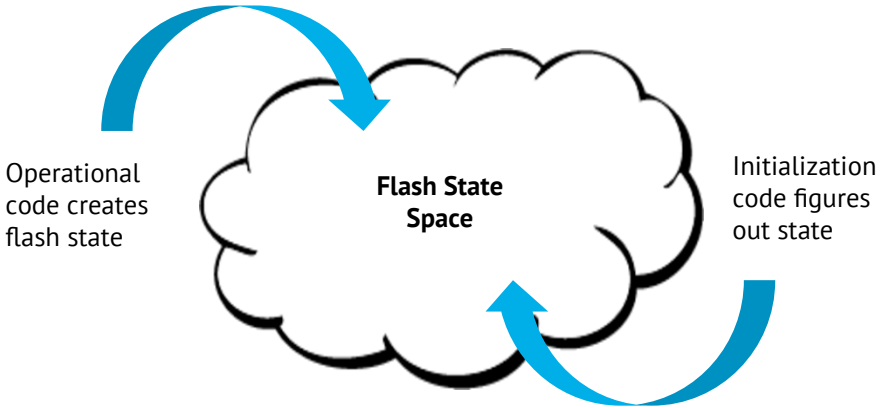
Continuous Integration

Agile development methodology

Best practices for FW development lead to Agile Development processes, Continuous integration, and requirements based testing. Rigor in testing optimizes TTM.
Automated testing increases equipment utilization, and enhances ability to find problems fast.

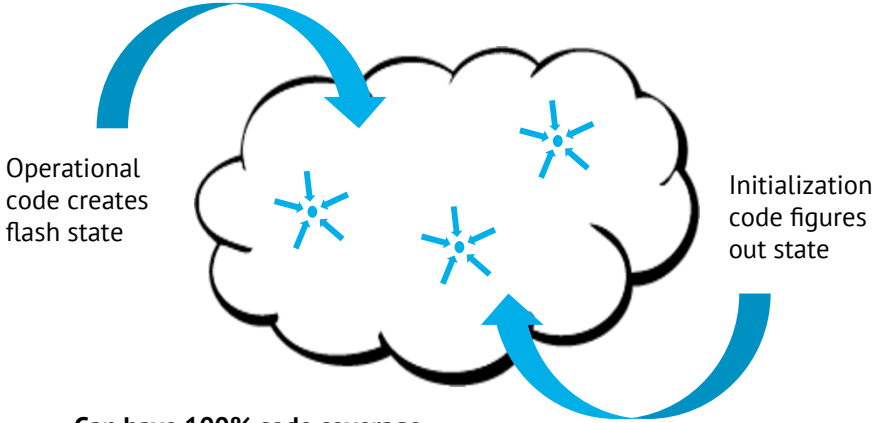
Why is power cycling difficult?

Permutations of flash state effectively infinite



NP-Complete problem to prove that operational code and initialization code are in synch

System works by collapsing the states down into smaller set of states through mutually understood rules



Can have 100% code coverage in both sets of code – but does not mean it works as the two pieces of code interface through infinite state space

Problem is hard, but testing it is hard as well

How do you create 'interesting' scenarios?

- Unit test? Error injection? State triggering?

When you perform test, how do you detect errors?

- Failures that start up code recognises are easy
- Ones that do not get detected during start up can be very problematic (solve with read scan after cycle with lba/version tagging in data field). Possible to also add additional checking in model environment
- HW related failures need to be guaranteed to be captured - eg Supercap didn't hold up long enough

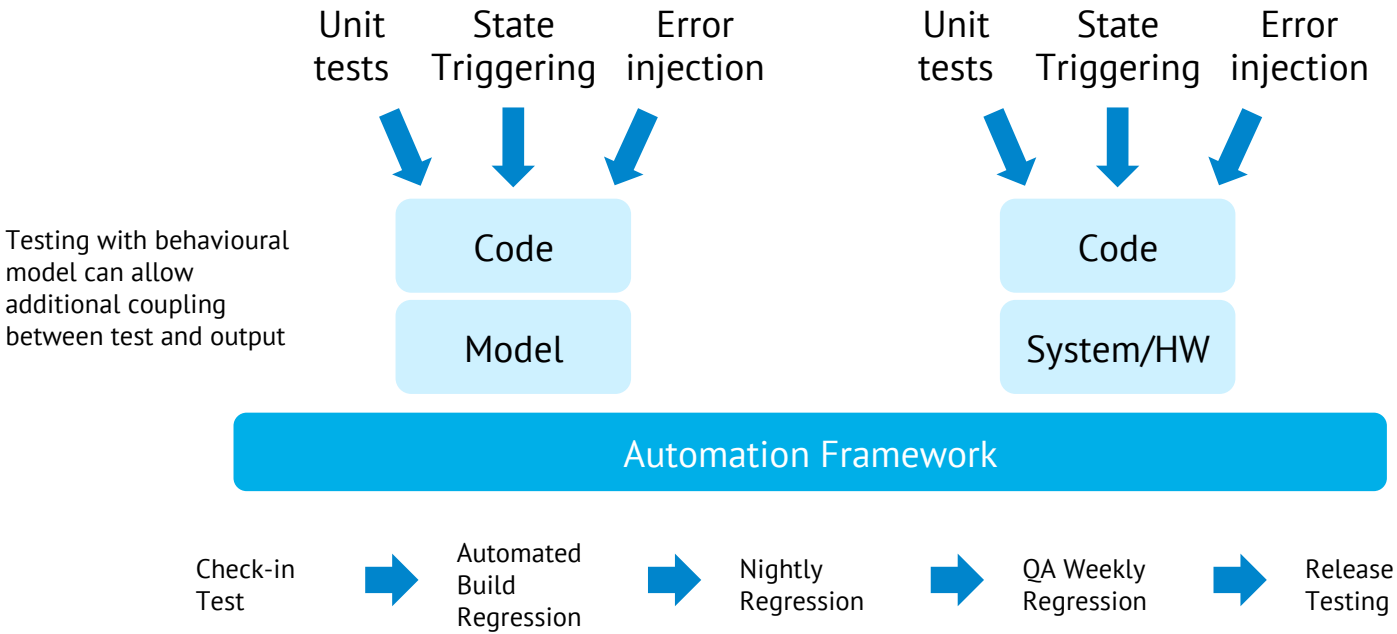
Time to loop can be long - especially for system (vs SSD)

Tests need to be regularly regressed

- You're never done, you can only achieve a level of confidence that it should work in the field
- For client SSD target 100K power cycles for release, with 20% of them random / unexpected
- For system solution, where cycle time is much longer less cycles are practical

Must approach systematically, repeatably, and automated to mature in 6 months

Holistic, layered approach is best solution



Data Path Testing

- Data path testing is challenging for similar reasons as power cycling. The amount of state permutations is effectively infinite

- Key differences between data path testing and power fail are frequency of use (data path code used every IO), and timing sensitivity (lots of things going on in parallel)

- Key capability needed: Data tagging with lba, and version/timestamp that is tracked by the test tool. Note that with larger storage systems this can be a hard problem in itself just due to size. Techniques using/tracking hashes of timestamps can help solve size of the data problem

Multi threaded operation

NVMe SSD's and storage systems have higher performance requirements than can typically be met with single CPU solutions.

Coherency between threads is difficult development problem and hard test problem.

Creating stress cases can be difficult and tests may be fragile, and have poor repeatability:

- Best practice is to ensure that stress is measured & monitored within the automation framework so that at least you can observe whether test is effective or not
- Goal of repeatability is less achievable, observability is new target

Devicepros

Straight Foward

Contact:
andy.tomlin@devicepros.net
www.devicepros.net

- **Straight Forward.** Our company motto embodies simplicity, efficiency, and transparency. We adopt “best practices” ruthlessly enabling speed, security and control as needed;
- Led by storage industry veterans, Derrill Sturgeon and Andy Tomlin. Based in Bay Area with development center in Minsk, Belarus providing expertise in Test Automation and Flash based systems to the Storage industry;
- Passionate about helping our customers achieve their goals! We aim to delight our customers with our services, not merely satisfy;
- Excellence in communication is one of our top priorities. Engineering Services requires robust and frank conversations from inception through execution.