Open. Together.
Large-scale operations for our next-gen Fabric and Fabric Hardware

Lalita Damaraju, Facebook
Ashik Ratnani, Facebook
Agenda

- Hardware Testing – New Platforms
- FBOSS Deployment Infra
Hardware Testing - New Platforms

- Validate new HW platforms with high confidence
- Provide reliable readouts in ever compressing time schedules
HW Design Cycle

RFP → SCH Layout Design → EVT Build → EVT Bringup → EVT HW Test

DVT → DVT SCH Layout → DVT Build → DVT Bringup → DVT HW Test

PVT → PVT SCH Layout → PVT Build → MFG Integration → MP
Revisit processes

Development

- HW Arch /Design Specs
- EVT
- DVT
- PVT

Software

- SW Arch / Design Specs
- SW Integration and Bringup
- SW Development
- Feature Freeze

SW Testing

- Test Plan Dev.
- Extended SW Enabled Platform Testing

MFG

- MFG and MFG test Processes

Production/Deployment

- MP

Open. Together.
Maintaining Quality w/ Speed

Build HW Validation pipeline where tests

• Leverage partner ODMs experience
• De-risk new designs/newer deployment use-cases with minimal extension on timelines
• are repeatable/re-usable across phases and platforms
• generate reliable results
Hacking Hardware Tests

1. Define Tests
2. Build Automation
3. Insert in hardware pipeline
Automatable HW Tests

• **Characterization Tests** (tracked for data sets throughout the program)
  • Link stability tracked against SERDES configuration parameters, Jitter – Phase Noise measurements as function of traffic/load, fan speed calibration

• **Functional Tests** (Common Tests Across Platforms)
  • EEPROM read/write tests, BMC functions

• **Stress Tests** (repeated stress in single iteration)
  • Sensors, fan speed variation, low level cpld functions

• **Regression Tests** (repeated over different hardware test phases)
  • Reboots, Resets, Sensors

• **Integration Tests** (multiple validations at same time)
  • Temperature, Fan, and power measurement with traffic load variation, Non disruptive upgrade or monitoring functions provided by BMC, link flap, loopback tests
Extending HW Test Cycle

- RFP
- SCH Layout Design
- EVT Build
- EVT Bringup
- EVT HW Test
- EVT Automation Prep Partial Automation
- EVT Regression
- HW Automation, Regression using diags
- DVT SCH Layout
- DVT Build
- DVT Bringup
- DVT HW Test
- Sampled RDT, Regression Tests
- PVT SCH Layout
- PVT Build
- MFG Integration
- MP
Results and Key Gains

- 2x reduction in manual review of logs
  - Clear failure signature in tests
  - Improvement in tools that help HW Debugging
  - Enhanced debug tools in diags
- Efficiency gains through Automation
  - Gaining 2x execution time using automated scripts
In Conclusion

• View **automation** as a resource that assists execution and **analysis**
• **Define tests from deployment perspective**
• View test unit at bench as a **remote DUT**
• Explore methods to extend **automation** deeper into **conventional EE validation /test cycles**
FBOSS Deployment Infra
What is FBOSS?

- FBOSS stands for “Facebook Open Switching System”
- It’s a software stack used for controlling and managing network switches
- FBOSS Agent Daemon manages the forwarding tables in the ASIC
Facebook’s Philosophy

• In 2016, Facebook moved from weekly release branch to a continuous push model for its web releases

  • Push code trunk automatically, directly, and regularly to production

• Benefits
  • Quicker turn around on rolling out bug fixes
  • Engineering effectiveness
  • Less changes → Less issues → Less time
Risks

• FBOSS is a Tier-0 service
  • Taking down a rack switch would disrupt multiple servers
Risks

- Update itself is a complicated workflow
- FBOSS Agent uses “Warmboot”
  - Restart without interfering the data plane
  - Subject to time limits
- Control plane disruption is expected
  - Routing daemons need to initiate a graceful restart
Update Workflow (Good Case)
Update Workflow (Bad Case)
Mitigating Risks

- Stay “trunk-stable” with high confidence
- Limit disruption below a fixed threshold
How to be “trunk-stable”? 

Testing, testing, testing

- Unit tests to validate FBOSS Agent
- Integration tests to validate SDK behavior
  - Run on all platforms
- Test on every code commit, and continuously on trunk
How to be “trunk-stable”? 

Canarying (Test in prod) 
- Test common operations on production switches 
  - Update, restart and roll-back 
  - Canary switches selected from a pool of ”Non-Critical” production switches 
- Tests are run hourly and daily
Limiting disruption

Monitoring, monitoring, monitoring

• Monitoring is built-in to our deployment infra
• Things we monitor:
  • Server reachability
  • Neighbor route updates
  • Switch State – ports, routes, peerings, etc.
• Automatically detect and stop if disruption exceeds our thresholds
Limiting disruption

Phased Roll-out

- Release broken up into multiple phases
  - Phase 1 - 10 devices per h/w platform
  - Phase 2 – 100 devices per h/w platform
  - Phase 3 – 1 cluster per h/w platform
  - Phase 4 – 10% of the fleet
  - Phase 5 – 20% of the fleet
  - Rest of the world!
Results

• Every traffic impacting service running on a FBOSS switch is supported
• Services are updated every 2-4 weeks as opposed to ~3-6 months
• Traffic disruption limited to <0.1% of the updates
Questions?