FBOSS: Building Switch Software at Scale

Sean Choi, Boris Burkov, Alex Eckert, Tian Fang, Saman Kazemkhani, Rob Sherwood, Ying Zhang, James Zeng
Motivation
Scale of Facebook Community

- Facebook: 2.23B Monthly Active Users
- Messenger: 1.3B Monthly Active Users
- Instagram: 1B Monthly Active Users
- WhatsApp: 1.5B Monthly Active Users

2.5B people use at least one of these products
Our DC Network is Growing FAST!

Growth of FBOSS Deployments

- 1x at 0 months
- 5x at 6 months
- 10x at 12 months
- 15x at 18 months
- 20x at 21 months
- 30x at 24 months
**Challenges in Scaling the Network**

**Issues with Extraneous Features**

- Vendor switch software is built for *all* of their user needs
  - Most users have to deal with software with a lot of unused features

- Complexity in managing *excess* networking features
  - Increased downtime, operational complexity, and security holes

- **Slower** rate of change to validate a large range of features
Arrival of the White Box Switch

Customizable switch hardware and software

- **Customized** hardware
- **Pick** the **minimal** software needed for the specific network
- **Powerful CPU** to run more complex software
Existing Software Services

Reusing Existing Infrastructure

- Facebook has an infrastructure already in place for...
  - Monitoring and Data Analytics
  - Logging
  - Service Management and etc.

- Vendors generally do not have full access to these tools
FBOSS: Facebook Open Switching System
FBOSS: Facebook Open Switching System
is an experiment to discover if...

we can run switch software
in a similar way
we run our software services
FBOSS Design Principles

• **Switch-as-a-Server**
  • Continuous integration and staged deployment
  • Integrate closely with existing software services
  • Open-source software

• **Deploy-Early-and-Iterate**
  • Focus on developing and deploying minimal set of features
  • Quickly iterate with smaller “diffs”
Facebook Software Infrastructure

Some example of Existing Software Services

- **Scuba**: Real-time monitoring and data analysis (*VLDB 2013*)
- **Gorilla**: In-memory time-series database (*VLDB 2015*)
- **Robotron**: Network Management (*SIGCOMM 2016*)
FBOSS Overview

External Software

- Protocols (BGP, ECMP)
- Network Configurator
- Monitoring Service

Switch Software

FBOSS

Switch Hardware

Switch ASIC
FBOSS Architecture

- **Switch SDK**: Vendor provided software for ASIC interaction
- **Hardware Abstraction**: Hardware specific implementation
- **HwSwitch**: Generic interface for switch hardware, e.g., port control
- **SwSwitch**: Hardware independent switching logic, e.g., L2, L3 and ACL
FBOSS Architecture

- **State Observers**: Manage changes in the states within SwSwitch
- **Local Config Generator**: Generates local configuration from externally generated configuration
- **Thrift Management Interface**: Receives external commands via Thrift interface
- **QSFP Service**: Manages QSFP Ports
FBOSS State Management

Copy-on-write Tree
FBOSS State Management

Copy-on-write Tree

- **No** Read Locks
- **Easy** Debugging
- **Easy** Restarts
- **Complex** Implementation
- **More processing** per update
Testing, Deployment & Management
FBOSS Testing and Deployment

Sources of switch outages

- Software (60%)
  - Misc. Software Issues: 28%
  - Microserver Reboot: 24%
  - Kernel Panic: 8%
  - Microserver Unresponsive: 7%
  - Loss of Power: 14%
  - Bus Degradation: 5%
  - SSD Issue: 2%
  - PCI-E Timeout: 1%

- Hardware (40%)
  - Misc. Hardware Issues: 11%
  - Loss of Power: 11%
  - Bus Degradation: 7%
  - SSD Issue: 8%
FBOSS Testing and Deployment

3 Stage Deployment via fbossdeploy

- **Continuous Canary**
  - Deploy all commits continuously to 1-2 switches for each type

- **Daily Canary**
  - Deploy all of single day’s commits to 10-20 switches for each type

- **Staged Deployment**
  - Final stage to push all the commits to all the switches in the DC
  - Performed once every two weeks for reliability
FBOSS Management

How FBOSS interacts with network management system

- Configurations are generated by the network management system
- Configurations are staged locally
- Operational and Monitoring queries are passed through the Thrift interface
Experiences
Lessons from Deployment Experiences

- Side effect of infrastructure reuse
- Side effect of rapid deployments
- Resolving interoperability issues
Lessons from Deployment Experiences

• **Side effect of infrastructure reuse**
• Side effect of rapid deployments
• Resolving interoperability issues
Side Effect of Infrastructure Reuse

Issues with combining applications with different SLAs

• Switch software must be more reliable than most software services

• Warm boot: Retain ASIC tables and configurations at restart
• BGP graceful restart: Restores BGP states after warm boot
• New library caused warm boot to take longer than expected
  => Random BGP restart failures => Random network outages

• **Lesson Learned**: Be careful when using software with different SLAs
Conclusion
Conclusion

• Scaling massive DC networks requires leaner software with rapid updates

• FBOSS started 5 years ago as an experiment to see if we can build and deploy switch software as if we are running a generic software service

• Our experiences show that the experiment is quite successful
  • FBOSS is capable of quickly iterating, deploying and scaling switch software
Questions?

https://github.com/facebook/fboss