The History of Logging @ Facebook (Abridged)

KC Braunschweig
Production Engineer
Who Are You?

• Facebook Production Engineer since 2012
  • OS & Config Management (Chef) 2012-2015
  • Scribe 2015-2018
#movefast

- Reference links at the end of the slides
Anyone remember 2007?
Anyone remember 2007?
Who cares about logging anyway?

- Hannah Montana tour goes on sale
- Ticketmaster LLC v. RMG Techs. Inc.
  - Someone might try to abuse your app for their own gain
  - You may have to defend your app in court
- Your logs might impact how you charge customers
  - Compliance requirements?
Logging at Ticketmaster in 2007

Ticketmaster

syslog → network → syslog → nfs → storage appliance → tail → {perl}
Agenda

- The industry in 2007
- 2007-2010 – Open Source Scribe
- 2010-2015 – Scribe on HDFS
- 2015-2018 – Scribe on LogDevice
2007-2010 Open Source Scribe
Scribe 2007-2010

- scribed – c++ thrift server
- Scribe open sourced 2008
  - now archived 😞
- Scribe Tech Talk 2/27/2009
  - Robert Johnson & Anthony Giardullo
Scribe 2007-2010

Ticketmaster

syslog

network

syslog

nfs

storage appliance

tail

{perl}

Facebook

scribed

network

scribed

nfs

storage appliance

tail

{perl}

php?

So what’s different?
Data Model

struct LogEntry
{
    1: string category,
    2: string message
}

service scribe extends fb303.FacebookService
{
    ResultCode Log(1: list<LogEntry> messages);
}
Scribe 2007-2010

- Unix 101 - Do one thing and do it well
- Implications
  - Scribe is a transport layer
  - Never inspect or manipulate the payload
  - A log is a series of newline terminated strings, but that doesn’t matter
2010-2015 Scribe on HDFS
Scribe 2007-2015

scriber (fan-in) -> scribed -> storage appliances
Meanwhile... “big data”

- Hadoop – big data ecosystem
- HDFS – hadoop distributed filesystem
  - Patterned after Google File System paper
- Hive – SQL like queries on top of HDFS
  - Originated at Facebook
Scribe 2007-2015

scribed

calligraphus (java)

HDFS

scribed
Scribe 2007-2015

calledgraphus (java)

HDFS

scribe

scribe
Scribe 2007-2015

- scribed
- scribed
- calligraphus (java)
- HDFS
- ptail (here be dragons)

HIVE
Scribe Ecosystem in 2015

scribed

“Scribe service”

hive loader

ptail
Scribe Ecosystem in 2015

Not pictured:
- Category registration
- Blacklisting
- Sampling
- more
Scribe Ecosystem in 2015

Region 1
- scribed
- scribe backend clusters (calligraphus/HDFS)
- hive loader

Region 2
- scribed
- scribe backend clusters (calligraphus/HDFS)
- ptail
Scribe Ecosystem in 2015

Hitting limitations

- Scribe writes hundreds of GB/s with thousands categories
- Scary growth rates -> we need to scale 10x
- Scaling metadata was harder than scaling writes
  - Mitigated with “scopes” but not solved
- Streaming was increasingly important
  - Built on ptail (a hack)
- HDFS support was being deprecated inside Facebook
Meanwhile...

Enter LogDevice

- Logdevice
  - Started early 2013
  - Open sourced 2018
- HDFS is a distributed filesystem we used to store logs
- LogDevice is a distributed log system
Meanwhile...
In LogDevice and the industry

• Logs get an explicit definition
  • A log is a series of newline terminated strings
  • A log is a record-oriented, append-only, trimmable stream
  • The Log: What every software engineer should know about real-time data’s unifying abstraction (Jay Kreps)
  • No problem for Scribe’s data model!
Meanwhile...

Enter LogDevice

- Streaming is first class
- Built in trimming (by time or size)
- Built in transport encryption
- Supported by our own dev team, dedicated to our use case
- Distributed metadata (no Namenodes)
2015-2018 Scribe on LogDevice
Scribe Ecosystem with HDFS
Scribe Ecosystem with LogDevice
Scribe Ecosystem with LogDevice

Write-side migration
Scribe Ecosystem with LogDevice

- scribed
- scribex router
- hive loader
- ptail

Write-side migration

- scuba
- stylus
- swift
- puma
Scribe on LogDevice 2015-2018
Migration Plan

- Write-side constraints
  - Thrift
  - FB service discovery & routing
  - Scopes used to scale metadata
Scribe on LogDevice 2015-2018

Migration Plan

- Write-side constraints
  - Thrift
  - FB service discovery & routing
  - Scopes used to scale metadata
- Use scopes to migrate by category
- Independent ”scribe” clusters
- We can even double write*
Scribe Ecosystem with LogDevice

- scribed
- scribex router
- hive loader
- ptail
- scuba
- stylus
- swift
- puma
Scribe Ecosystem with LogDevice

- scribed
- scribex router
- LogDevice
- ptail
- new hive loader
- HIVE

Reduce complexity

- scuba
- stylus
- swift
- puma
Scribe Ecosystem with LogDevice

scribed

scribex router

ptail

new hive loader

Read-side migration

HIVE

scuba

stylus

swift

puma
Scribe Ecosystem with LogDevice

- scribed
- scribex router
- LogDevice
- ztail
- new hive loader
- Read-side migration
- scuba
- stylus
- swift
- puma
Scribe on LogDevice 2015-2018

Migration Plan

• Read-side constraints
  • Pipe interface
    • `ptail -f my_category | my_stream_app`
  • No one ever upgrades
Scribe on LogDevice 2015-2018
Migration Plan

- Read-side constraints
  - Pipe interface
    - `ptail -f my_category | my_stream_app`
  - No one ever upgrades
- Hide behind pipe interface
- Migrate using transparent, forced upgrades
  - `ptail-autoupgrader` (binary is still just `ptail`)
Scribe on LogDevice 2015-2018
A 6 month project you say?

- New components: scribex router, ztail, LogDevice, hive loader
  - Write code
- Migration Plan
  - Leverage our constraints
- Productionize
Scribe on LogDevice 2015-2018
Prototype implementations

scribex router
prototype complete
Scribe on LogDevice 2015-2018
Prototype implementations

Write rate at project start

Write Rate
- HDFS
- LogDevice
Scribe on LogDevice 2015-2018
Prototype implementations

ztail prototype complete
Scribe on LogDevice 2015-2018

How long could it take to make a plan?

Ready for production scale test

Write Rate
- HDFS
- LogDevice
Scribe on LogDevice 2015-2018

How long could it take to make a plan?

- The goalposts will move the longer a project goes on
Scribe on LogDevice 2015-2018

How long could it take to make a plan?

• The goalposts will move the longer a project goes on
• Replacing a mature system is hard
  • Feature specs are valuable but always incomplete
Scribe on LogDevice 2015-2018

How long could it take to make a plan?

- The goalposts will move the longer a project goes on
- Replacing a mature system is hard
  - Feature specs are valuable but always incomplete
- Put the new team on call for the legacy system
Scribe on LogDevice 2015-2018
Testing in prod

Write Rate
- HDFS
- LogDevice

Production scale tests complete
Scribe on LogDevice 2015-2018
Testing in prod

• End-to-end testing
  • Question both systems
Scribe on LogDevice 2015-2018
Testing in prod

• End-to-end testing
  • Question both systems

• Migration testing
  • How good is your test coverage?
  • Cover every tailer option permutation?
  • Cheat with empirical data
Scribe on LogDevice 2015-2018

Testing in prod

- End-to-end testing
  - Question both systems

- Migration testing
  - How good is your test coverage?
  - Cover every tailer option permutation?
  - Cheat with empirical data

- Cheat smart
  - Use Scuba to make the data manageable
  - Batch similar use cases together
Scribe on LogDevice 2015-2018
This journey is 15% complete
Scribe on LogDevice 2015-2018
This journey is 15% complete

• Migration automation
  • Ex: category owner communication, capacity checks, migration steps, etc
  • Make writing tooling easy with reusable components
  • Temporary is ok
  • Integration with our monitoring and other systems
Scribe on LogDevice 2015-2018
This journey is 15% complete

- Migration automation
  - Ex: category owner communication, capacity checks, migration steps, etc
  - Make writing tooling easy with reusable components
  - Temporary is ok
  - Integration with our monitoring and other systems
  - Hackable configuration (configerator)

```
{"date": "20171009",
"categories": [
  "pipe_finder",
  "pipe_indexer",
  "speedtrap_errors",
],}
```
Scribe on LogDevice 2015-2018
Now we’re rolling
Scribe on LogDevice 2015-2018

Holiday pause

Write Rate
- HDFS
- LogDevice
Scribe on LogDevice 2015-2018

Write Rate
- HDFS
- LogDevice

Done and done
Magic*
**Magic**

Layering is your friend

- Unix 101 – do one thing and do it well
Magic*
Layering is your friend

- Raw scribe – no structure, low dependency
  - `echo "foo" | scribe_cat test_category`
**Magic**

Layering is your friend

- Raw scribe – no structure, low dependency
  - scribe_cat
  - Ex: chef
**Magic**
Layering is your friend

- Raw scribe – no structure, low dependency
  - scribe_cat
  - Ex: chef -> json -> scribe_cat -> scribe -> scuba
  - No validation – good luck

```ruby
# chef handler
Mixlib::ShellOut.new(
  '/usr/local/bin/SCRIBE_CAT chef_stats',
  :input => Chef::JSONCompat.to_json(stats)
)
```
Magic*
Layering is your friend

- Our old friend syslog
- Imposed schema
Magic*
Layering is your friend

- Our old friend syslog
- Imposed schema, isolate dependencies
  - omscribe

```conf
# rsyslog.conf
action(type="omprog"
        binary="/usr/local/bin/omscribe"
)```

Our old friend syslog

Imposed schema, isolate dependencies
  • omscribe

Deal with it downstream
  • syslog -> omscribe -> scribe -> puma/stylus

Magic*
Layering is your friend

```bash
# rsyslog.conf
action(type="omprog"
  binary="/usr/local/bin/omscribe"
```
Magic*
Layering is your friend

- Facebook applications
  - ScribeClient libraries
- Custom integration
  - As much structure as the developer wants
  - Manage schema requirements yourself
Magic*
Layering is your friend

- Facebook applications
  - ScribeClient libraries
- Custom integration
  - As much structure as the developer wants
  - Manage schema requirements yourself
- Adding new destinations requires new schema management
- Thousands of engineers have to know what they’re doing
- System-wide optimization is hard
Magic*
Layering is your friend

- Full structured logging
  - logger - schema by config, destination(s) by config, automatic validation
  - configerator - schema distribution
**Magic**
Layering is your friend

- Full structured logging
  - logger - schema by config, destination(s) by config, automatic validation
  - configurator - schema distribution
- Magic
  - app -> logger -> JSON -> ScribeClient -> Scribe -> Scuba
**Magic**
Layering is your friend

- Full structured logging
  - logger - schema by config, destination(s) by config, automatic validation
  - configurator - schema distribution
- Magic
  - app -> logger -> JSON -> ScribeClient -> Scribe -> Scuba
  - Scuba JSON is inefficient, let’s change it!


Magic*
Layering is your friend

- Full structured logging
  - logger - schema by config, destination(s) by config, automatic validation
  - configurator - schema distribution
- Magic
  - app -> logger -> JSON -> ScribeClient -> Scribe -> Scuba
    - Scuba JSON is inefficient, let’s change it!
  - app -> logger -> thrift -> ScribeClient -> Scribe -> Scuba
    - Transparent for hundreds of apps
    - Transparent for Scribe
Conclusions

- Follow the Unix Philosophy
- Build complex features by layering simple components
- Your spec will be incomplete and take longer than you think
- Leverage your constraints
- Make tools easy to build to make them easy to throw away
- Sometimes a hack is good enough
facebook | Questions
Reference Links

- Facebook Thrift

- Ticketmaster vs. RMG Technologies

- Scribe Tech Talk 2/27/2009

- Open Source Scribe
  - https://github.com/facebookarchive/SCRIBE

- Hadoop Ecosystem
  - https://en.wikipedia.org/wiki/Apache_Hadoop
  - https://en.wikipedia.org/wiki/Apache_Hive

- The Log: What every software engineer should know about real-time data’s unifying abstraction (Jay Kreps)
Reference Links

• Realtime Data Processing at Facebook
  • https://research.fb.com/publications/realtime-data-processing-at-facebook/

• Open Source LogDevice
  • https://logdevice.io/

• Rsyslog omprog module

• Configerator – Holistic Configuration Management at Facebook
  • https://research.fb.com/publications/holistic-configuration-management-at-facebook/