

The logo features a large, circular network graph composed of numerous white nodes and connecting lines, set against a solid blue background. The graph is dense and irregular, forming a ring around the central text.

facebook

INFRASTRUCTURE

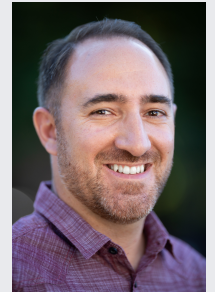
The History of Logging @ Facebook (Abridged)

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?

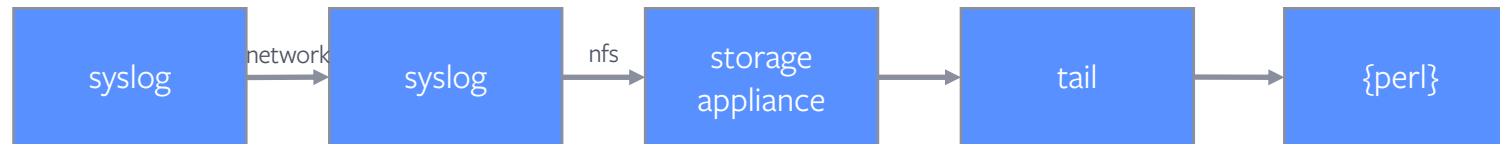
Anyone remember 2007?

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



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

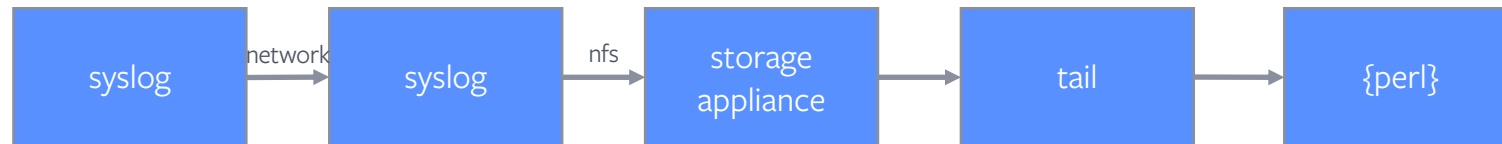
Scribe 2007-2010



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

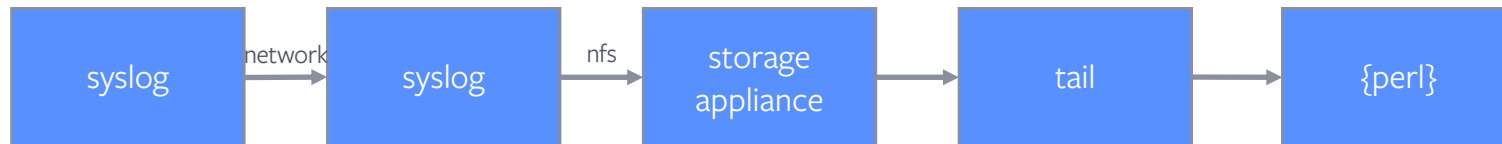
Scribe 2007-2010

Ticketmaster

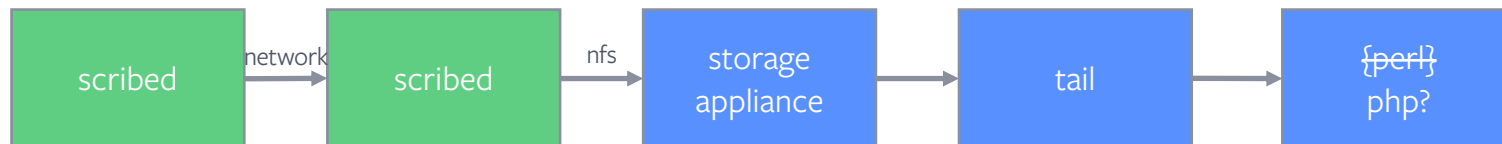


Scribe 2007-2010

Ticketmaster

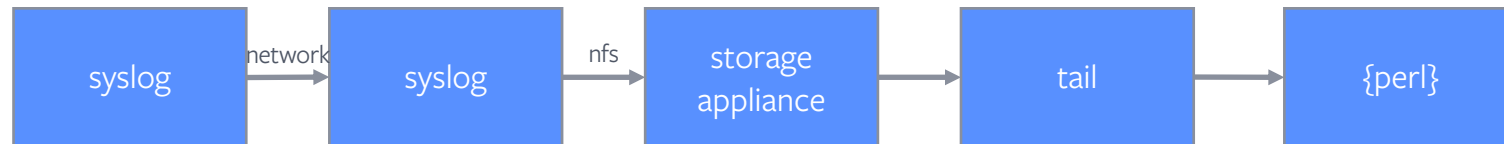


Facebook

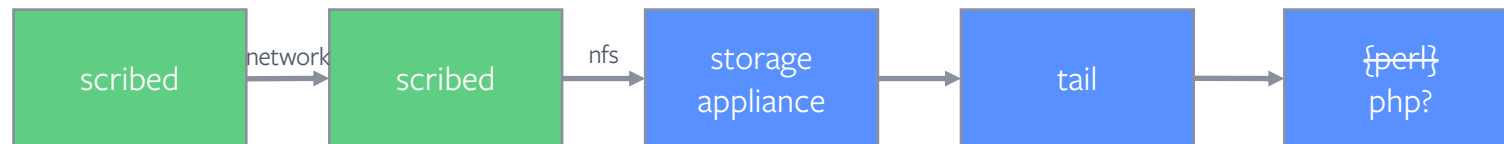


Scribe 2007-2010

Ticketmaster



Facebook



So what's different?

Data Model

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

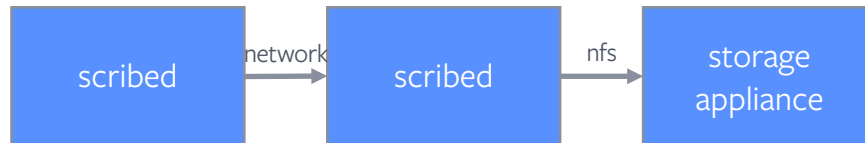
service scribe extends fb303.FacebookService
{
    ResultCode Log(1: list<LogEntry> messages);
}
```


Scribe 2007-2010

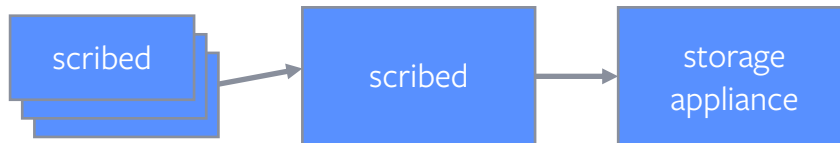
- Unix 101 - Do one thing and do it well
- Implications
 - Scribe is a transport layer
 - Never inspect or manipulate the payload
 - A log is a series of newline terminated strings, but that doesn't matter

2010-2015 Scribe on HDFS

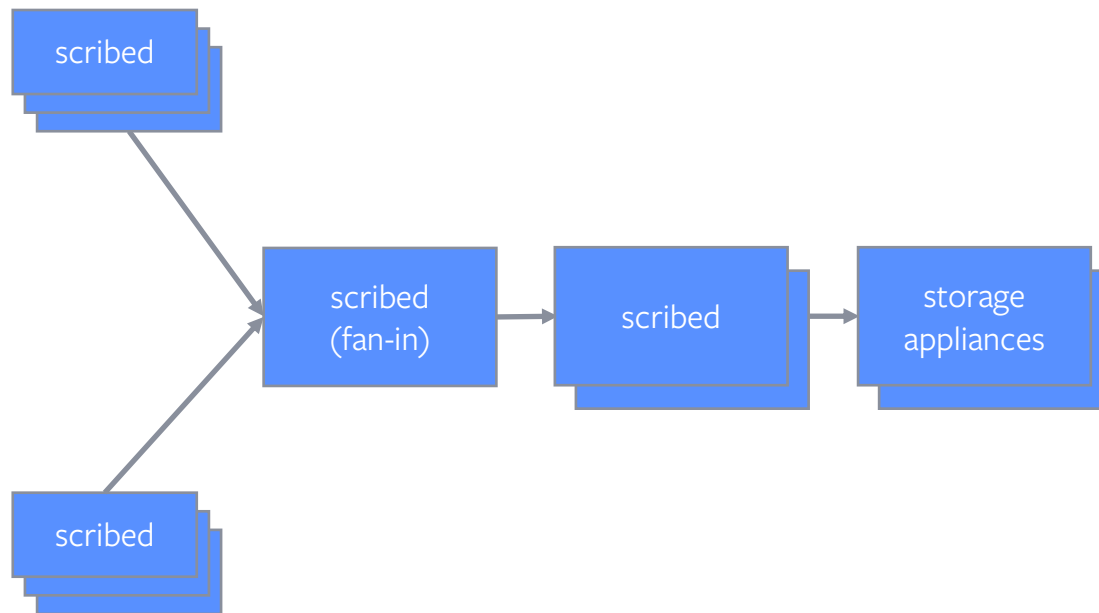
Scribe 2007-2015





Scribe 2007-2015



Scribe 2007-2015

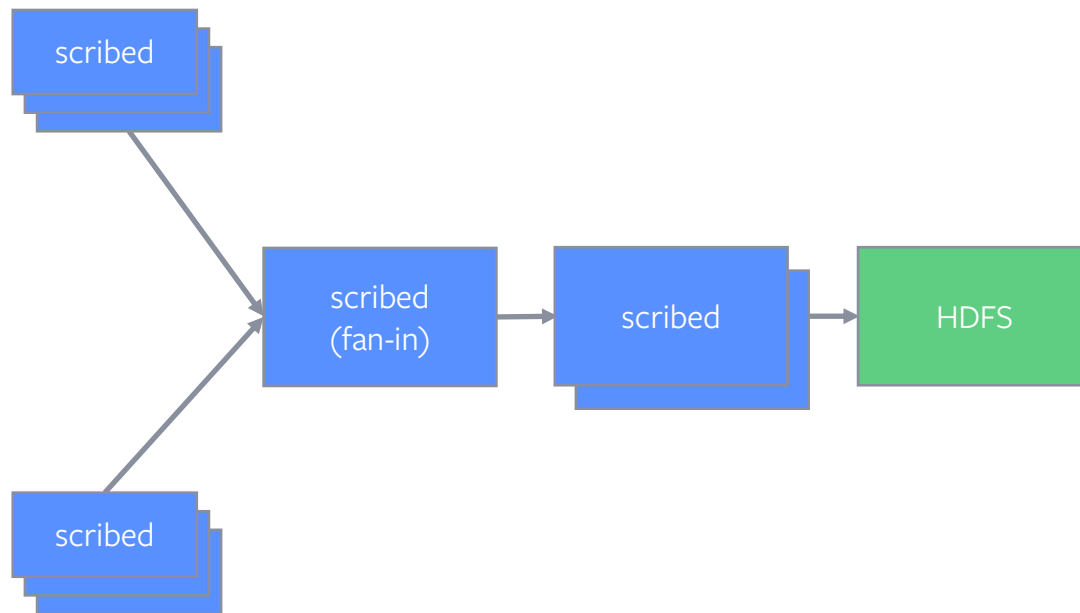


Meanwhile... ”big data”

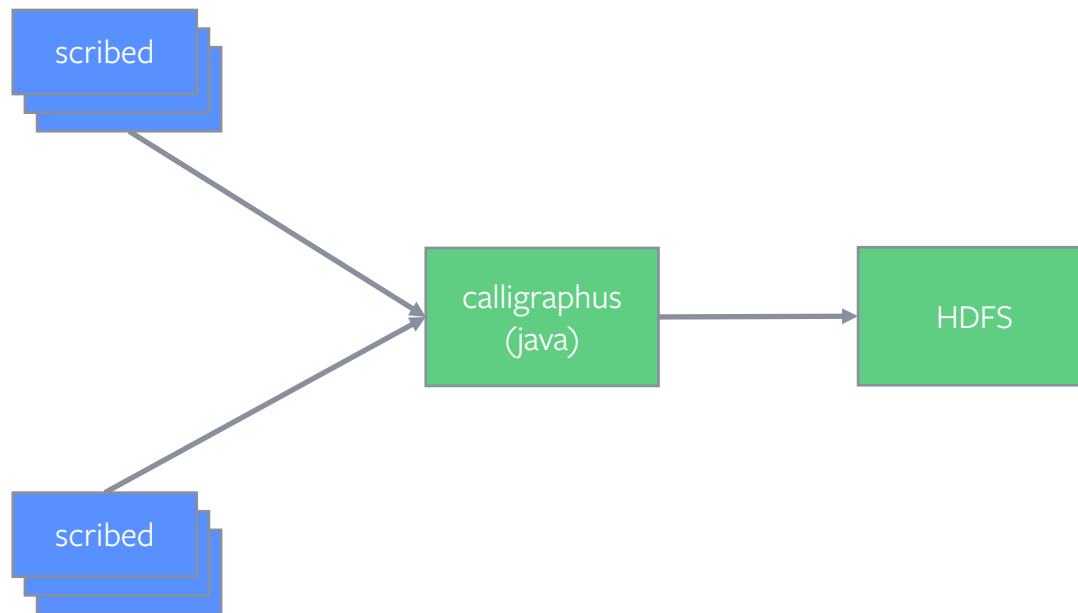
- Hadoop – big data ecosystem 
- HDFS – hadoop distributed filesystem
 - Patterned after Google File System paper
- Hive – SQL like queries on top of HDFS 
 - Originated at Facebook



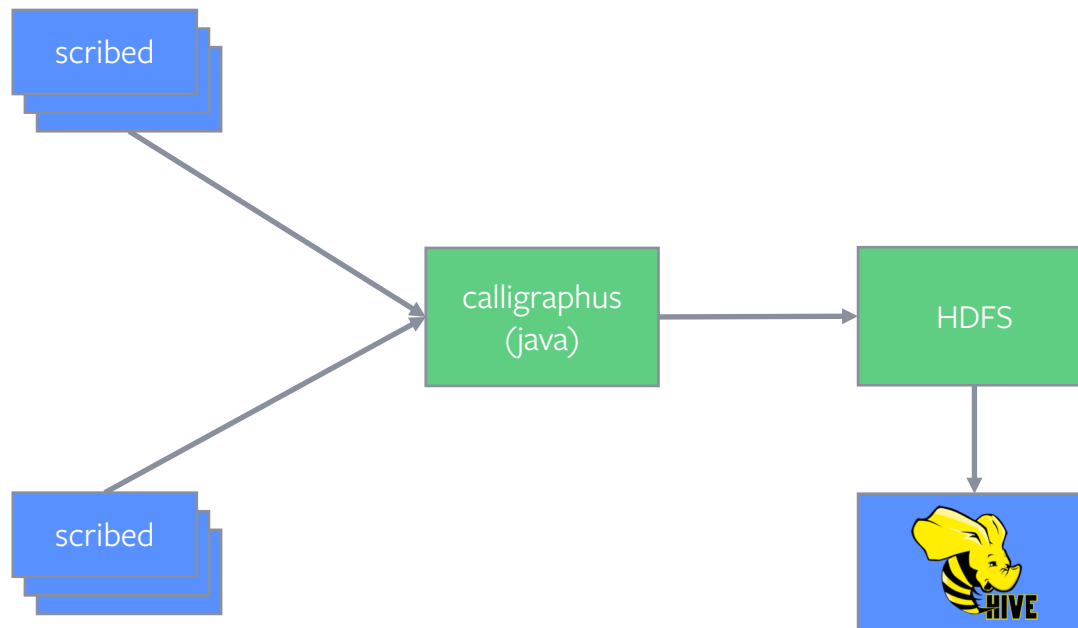
Scribe 2007-2015



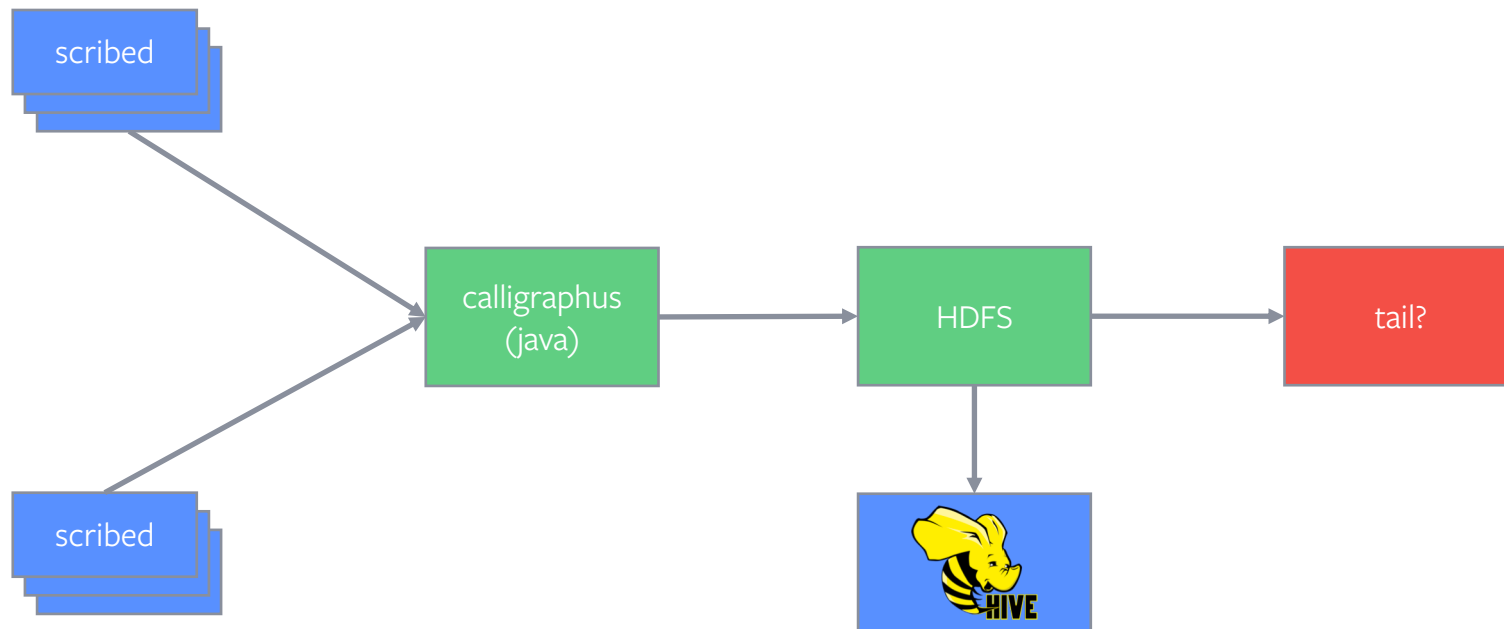
Scribe 2007-2015



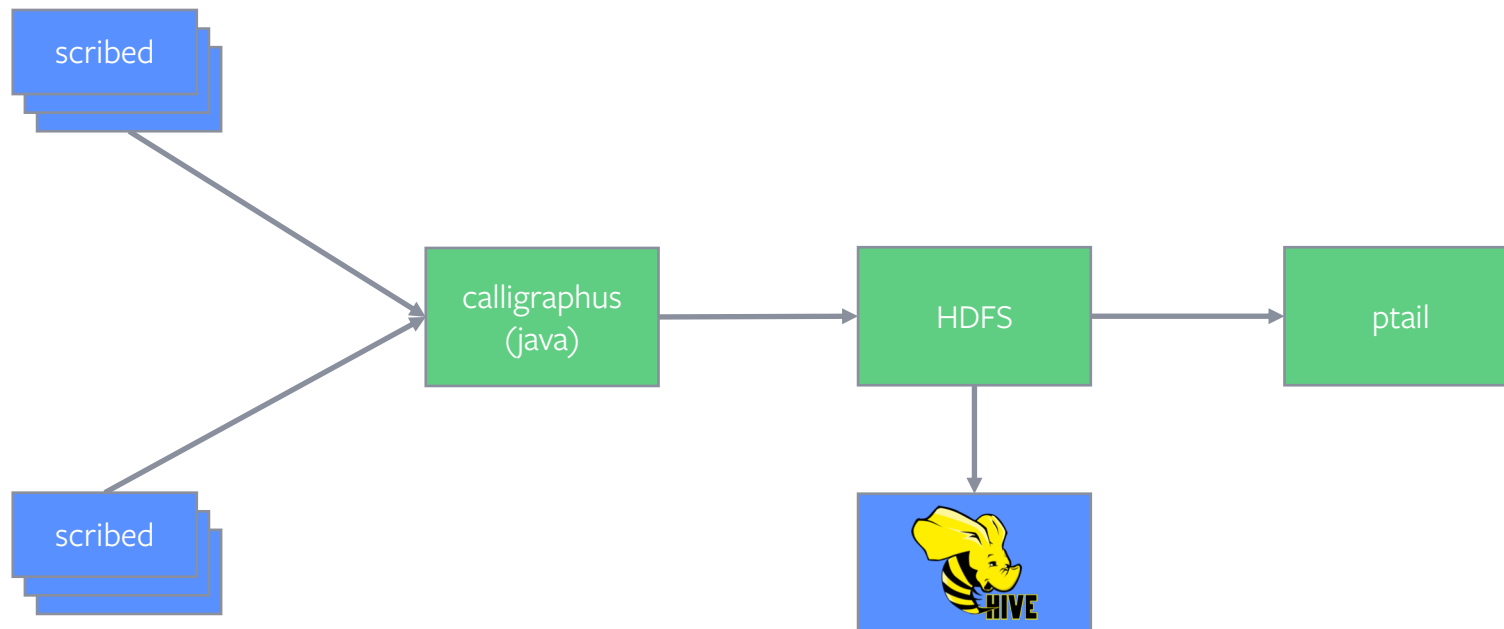
Scribe 2007-2015



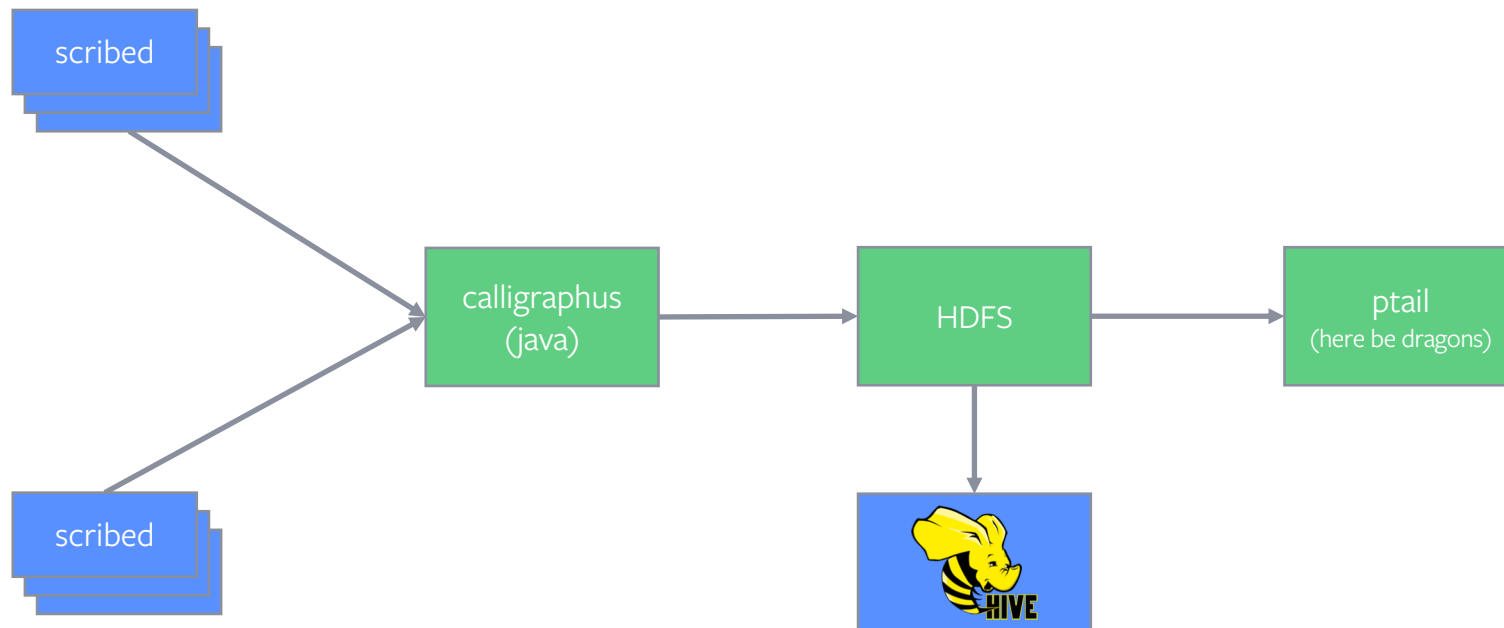
Scribe 2007-2015



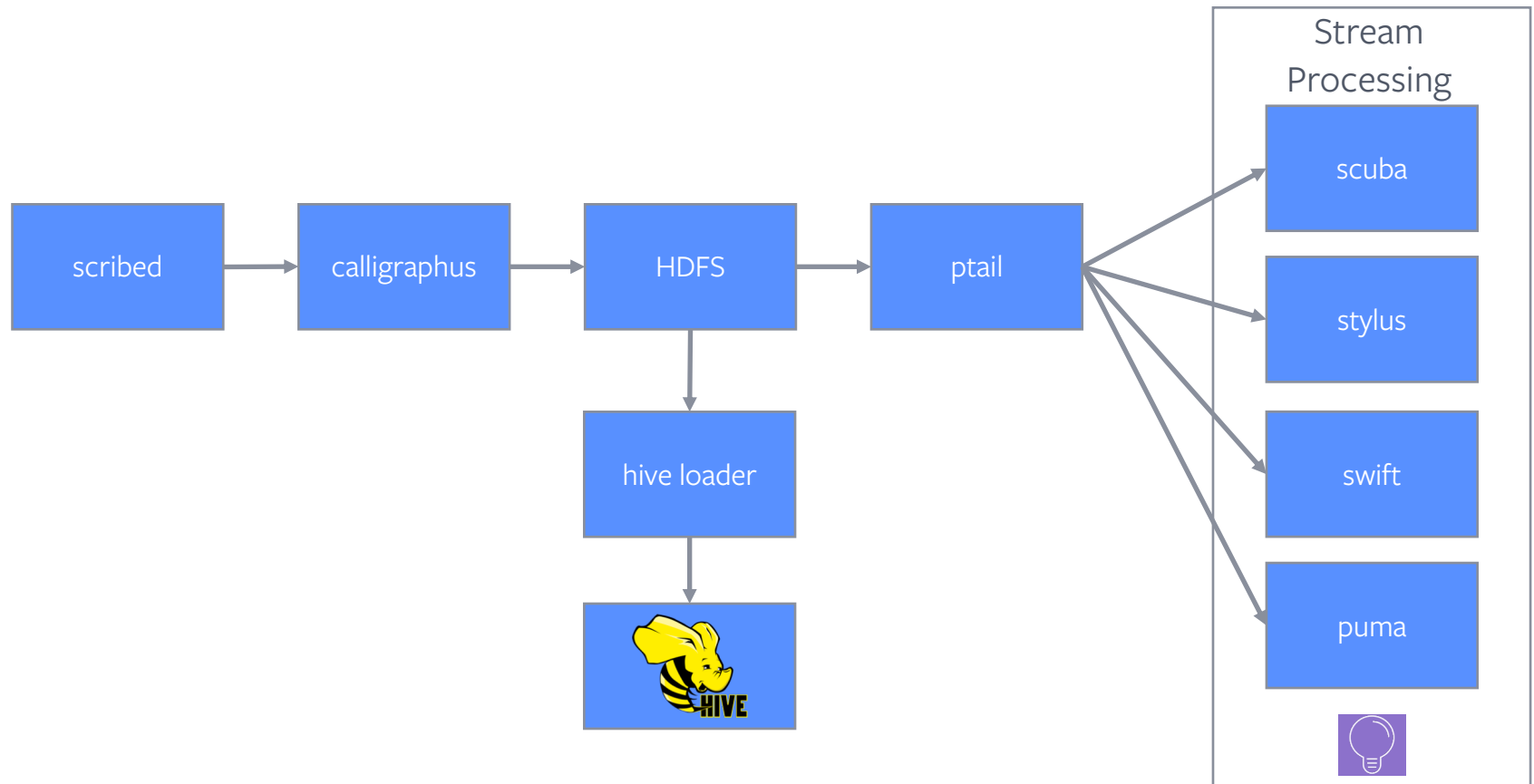
Scribe 2007-2015



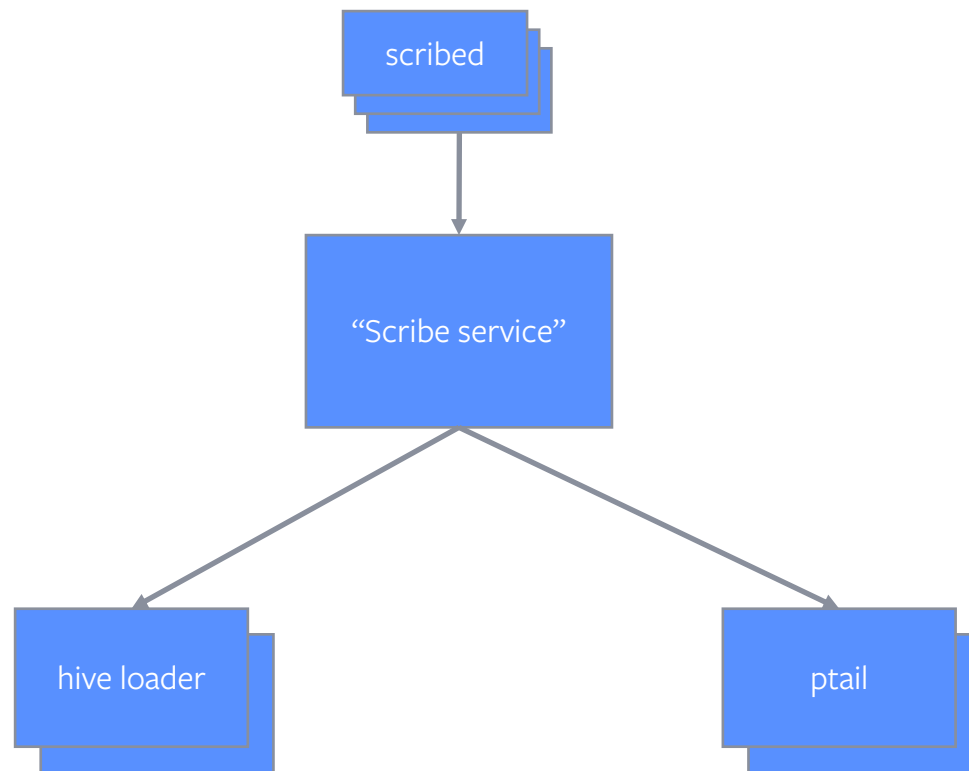
Scribe 2007-2015



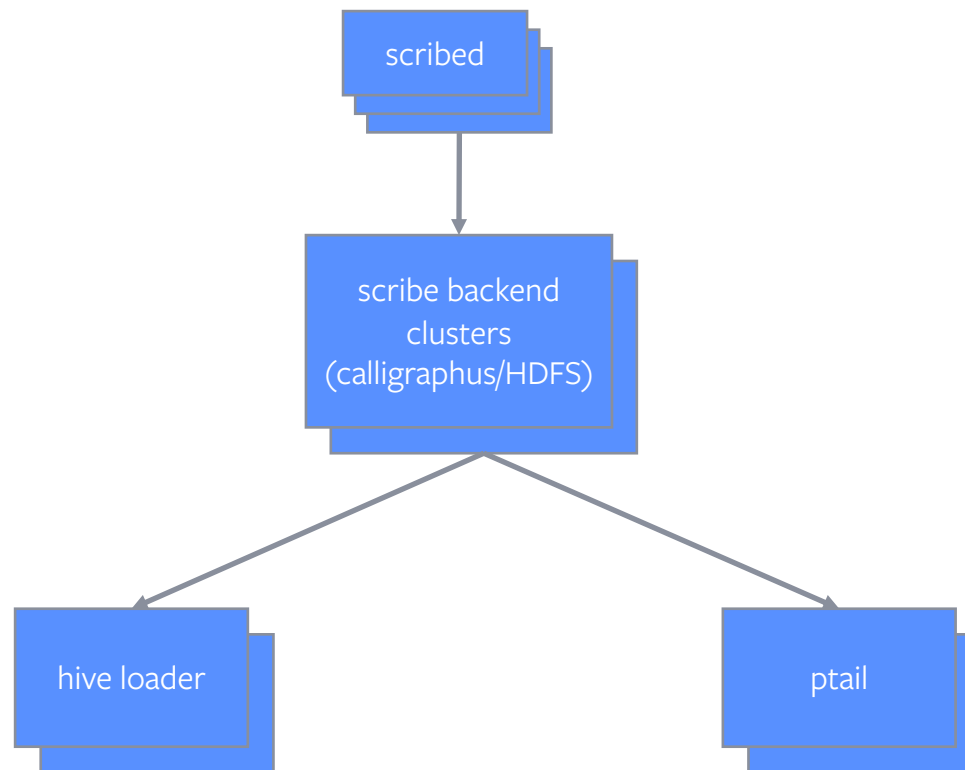
Scribe Ecosystem in 2015



Scribe Ecosystem in 2015



Scribe Ecosystem in 2015



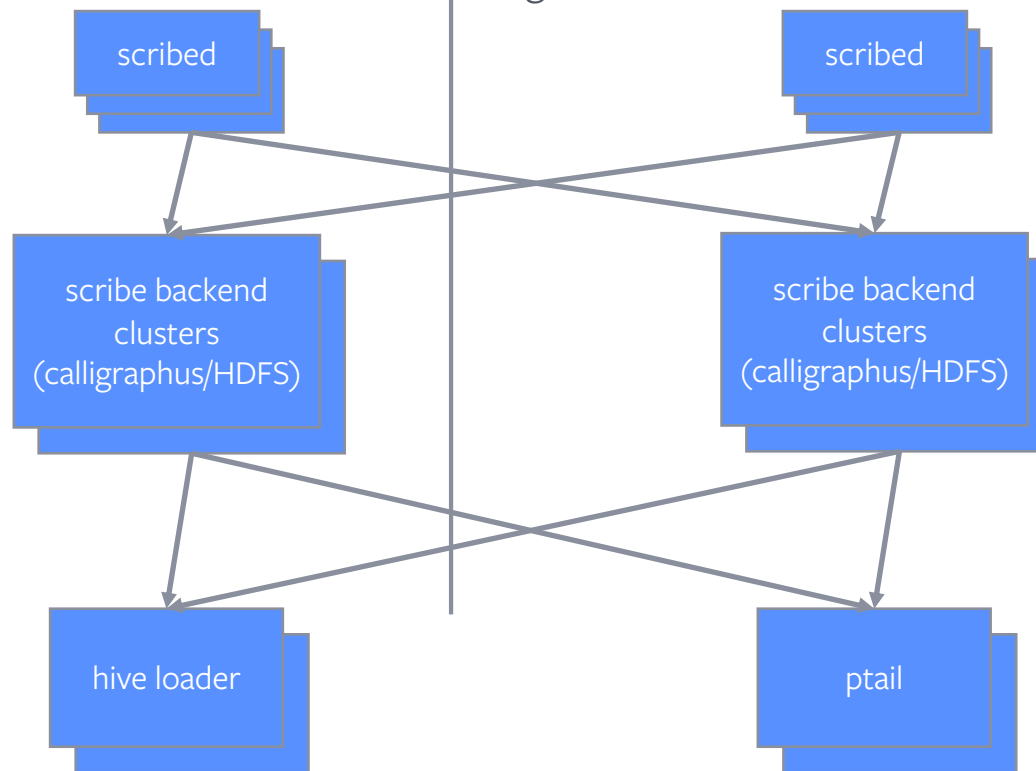
Not pictured:

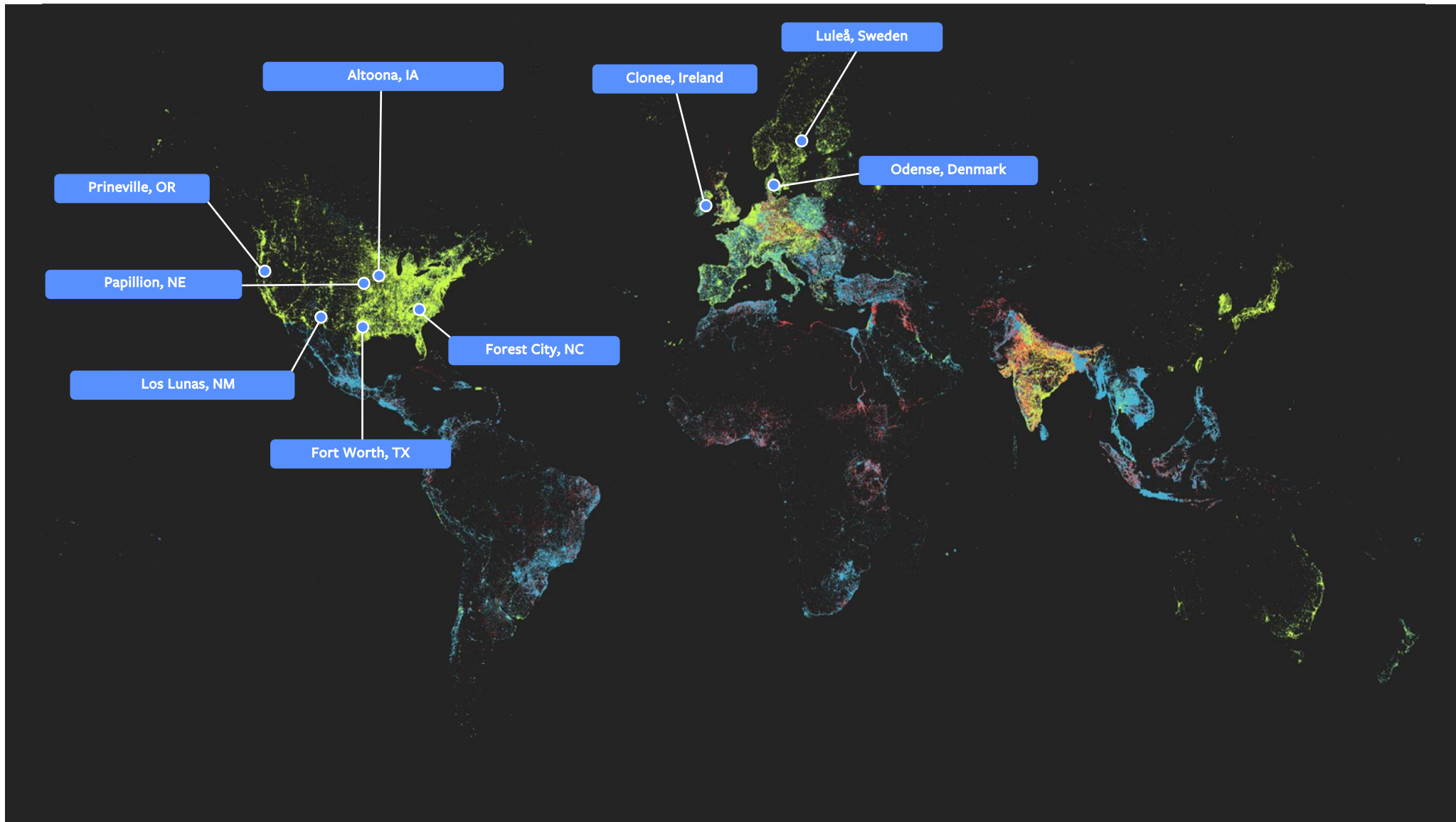
- Category registration
- Blacklisting
- Sampling
- more

Scribe Ecosystem in 2015

Region 1

Region 2





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...

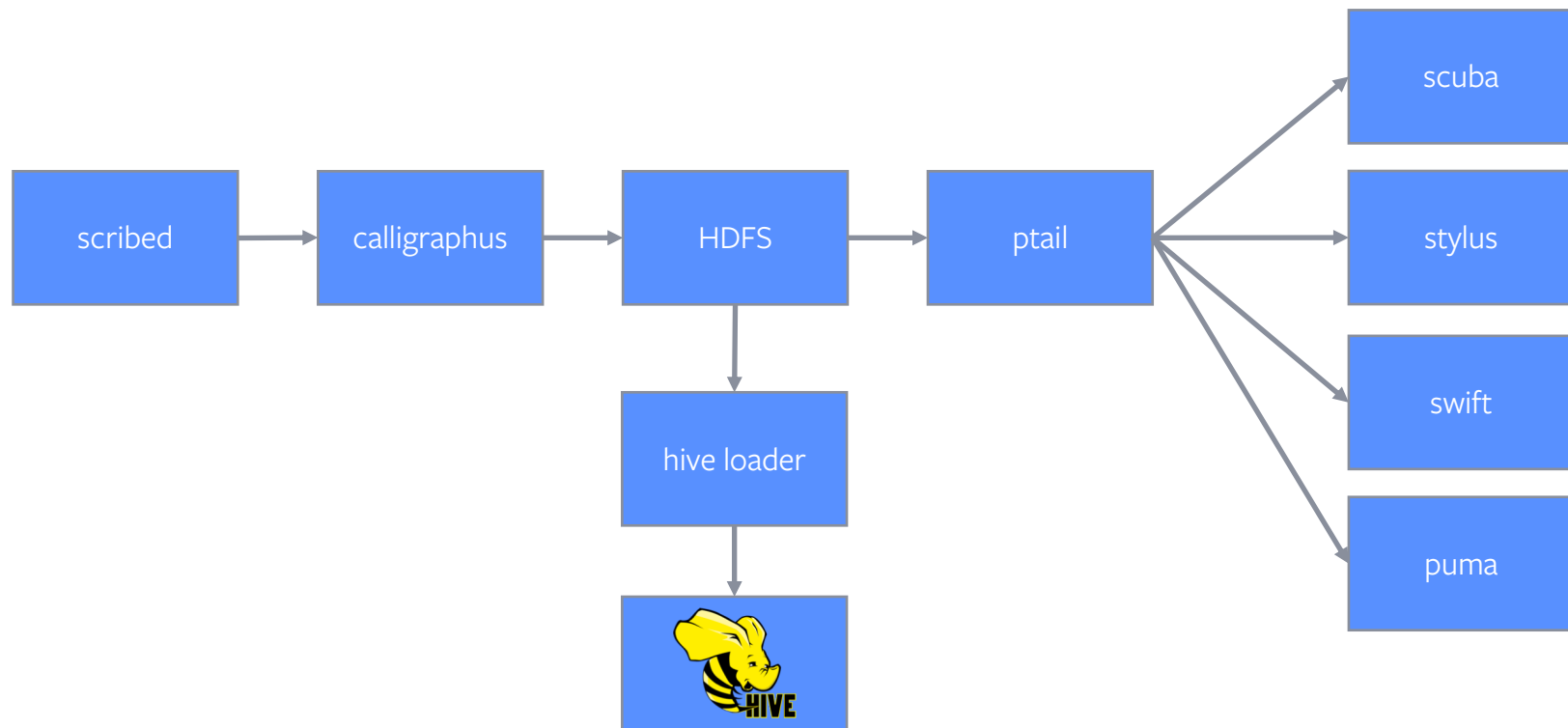
Enter LogDevice



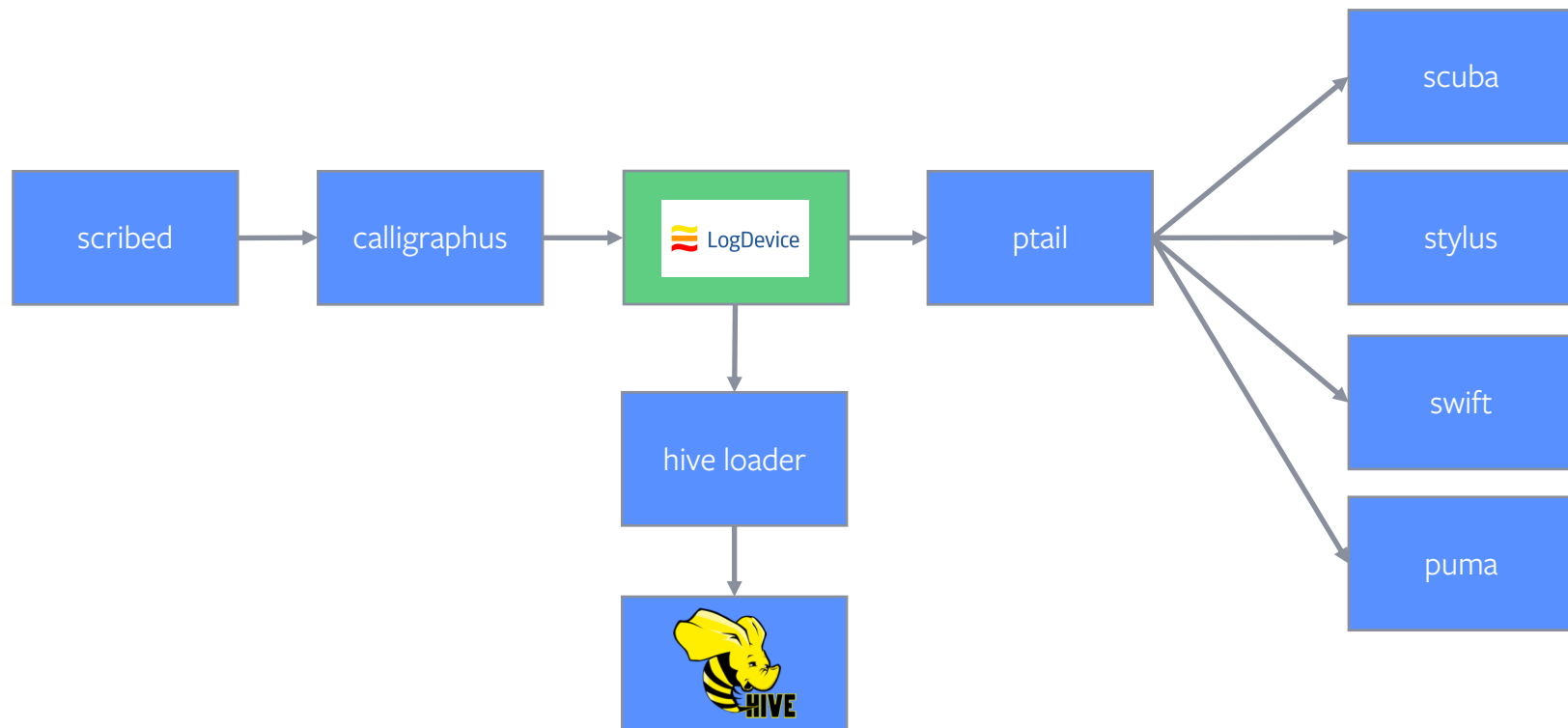
- Streaming is first class
- Built in trimming (by time or size)
- Built in transport encryption
- Supported by our own dev team, dedicated to our use case
- Distributed metadata (no Namenodes)

2015-2018 Scribe on LogDevice

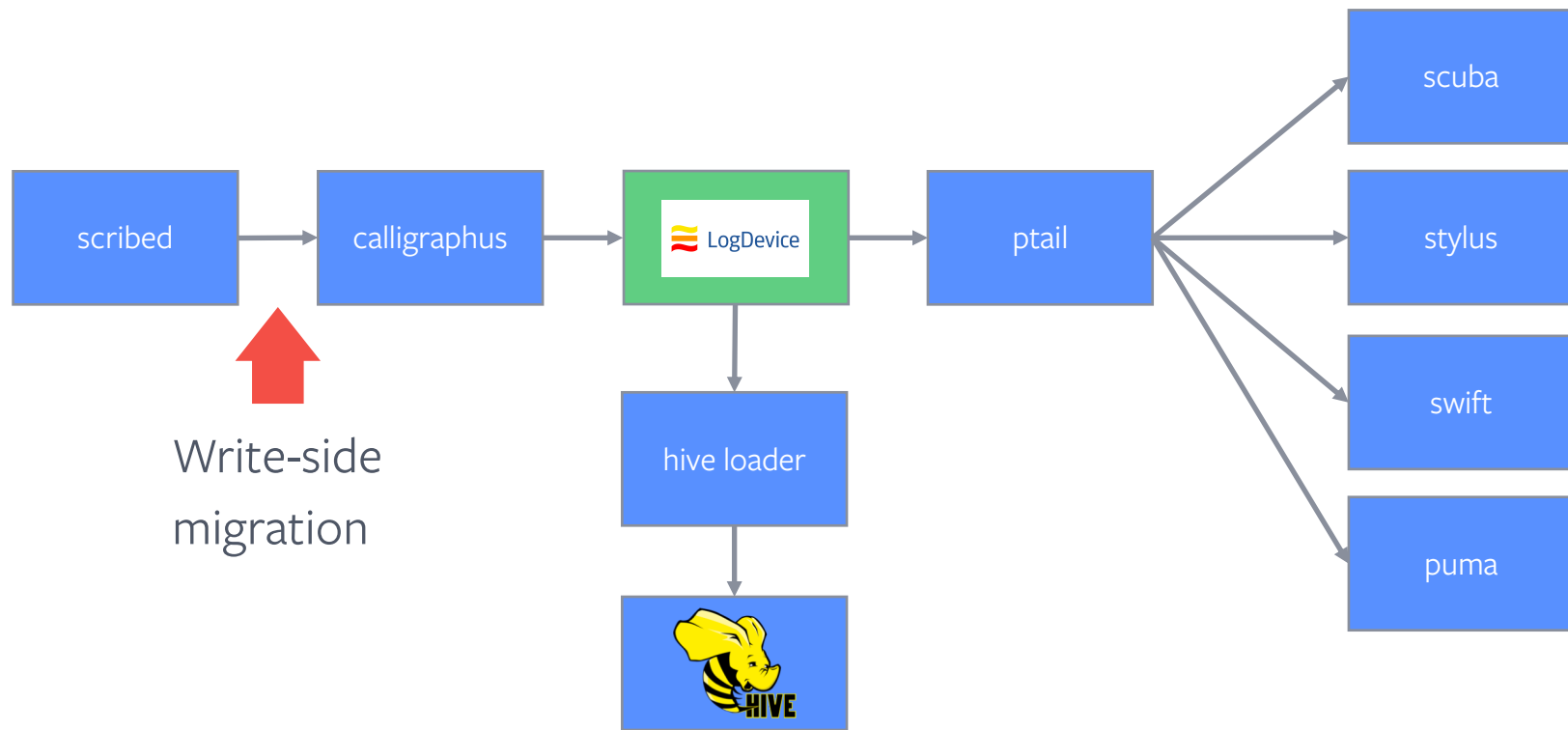
Scribe Ecosystem with HDFS



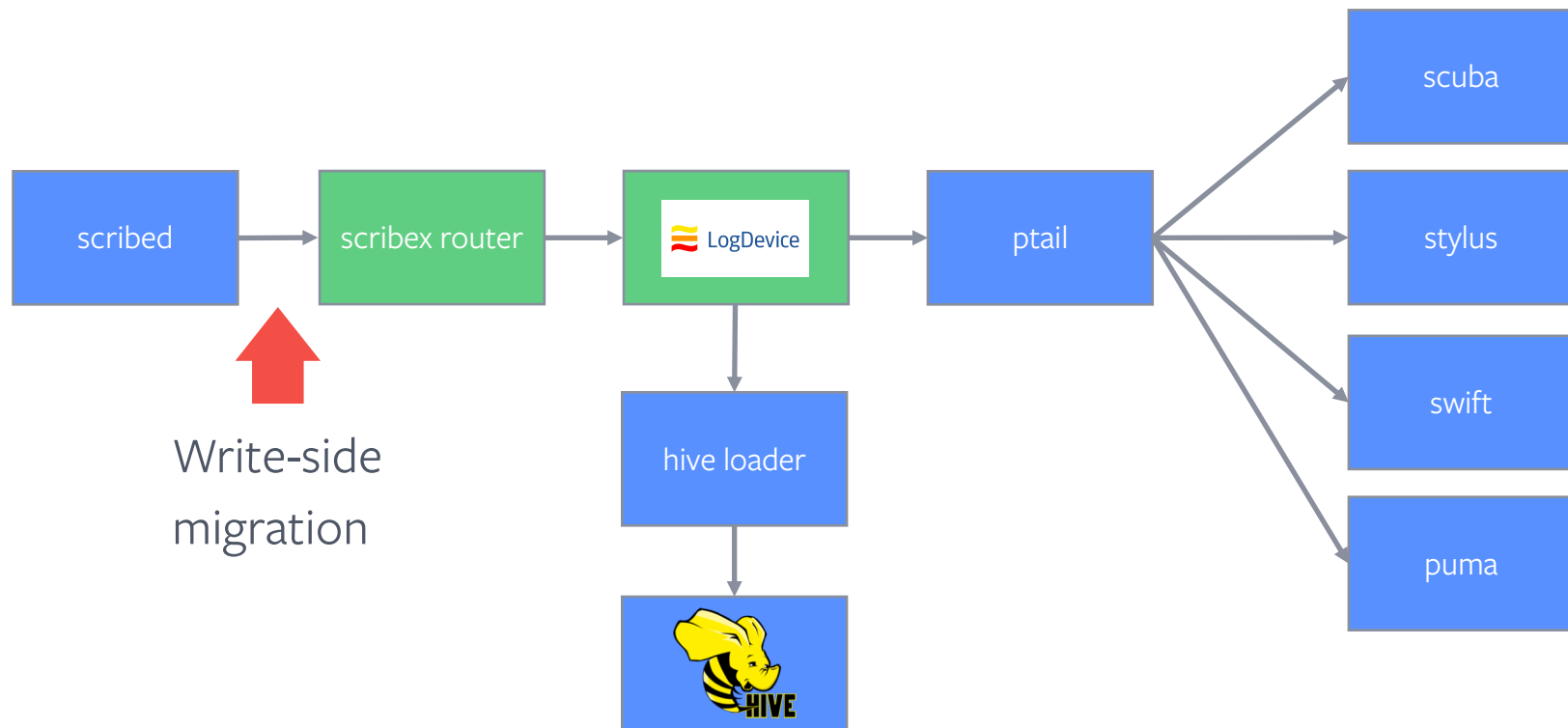
Scribe Ecosystem with LogDevice



Scribe Ecosystem with LogDevice



Scribe Ecosystem with LogDevice



Scribe on LogDevice 2015-2018

Migration Plan

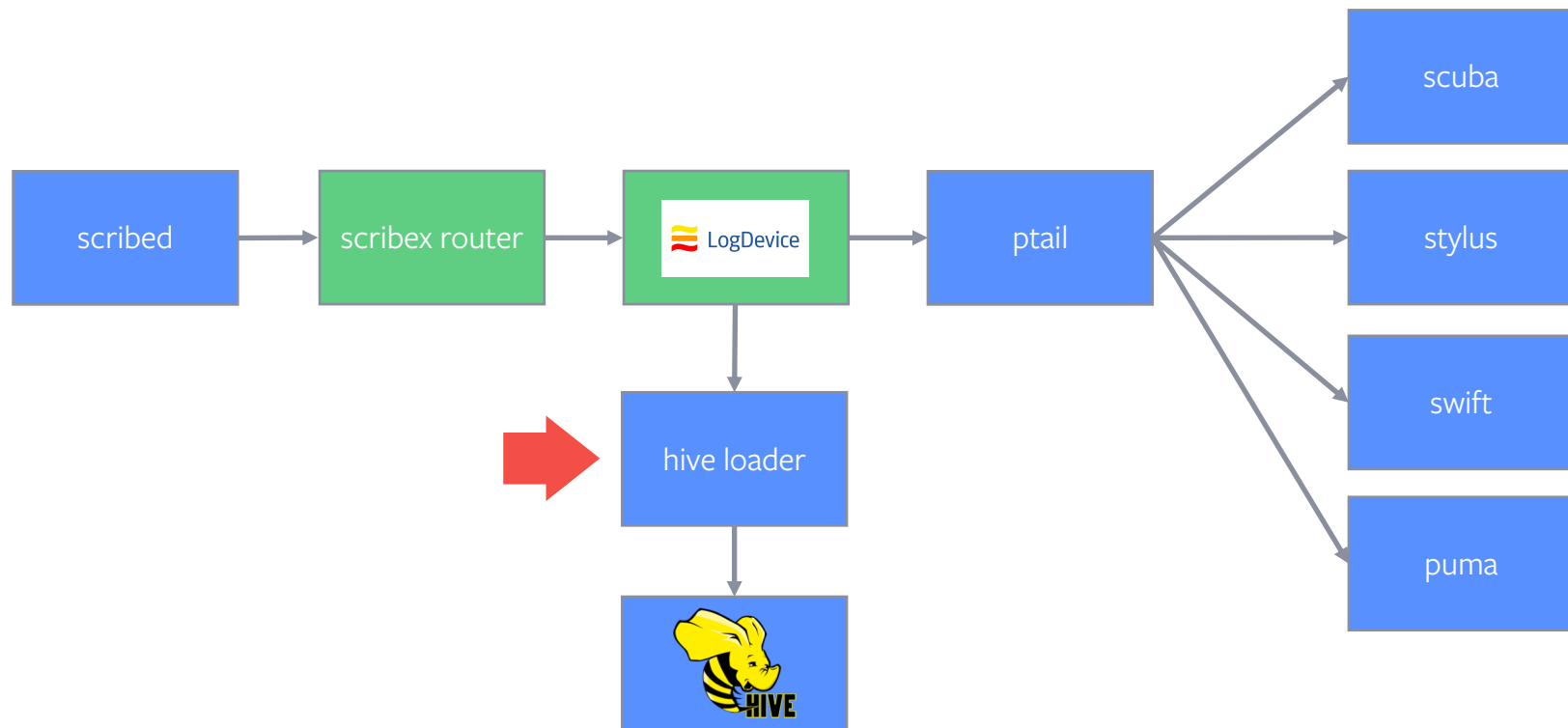
- Write-side constraints
 - Thrift
 - FB service discovery & routing
 - Scopes used to scale metadata

Scribe on LogDevice 2015-2018

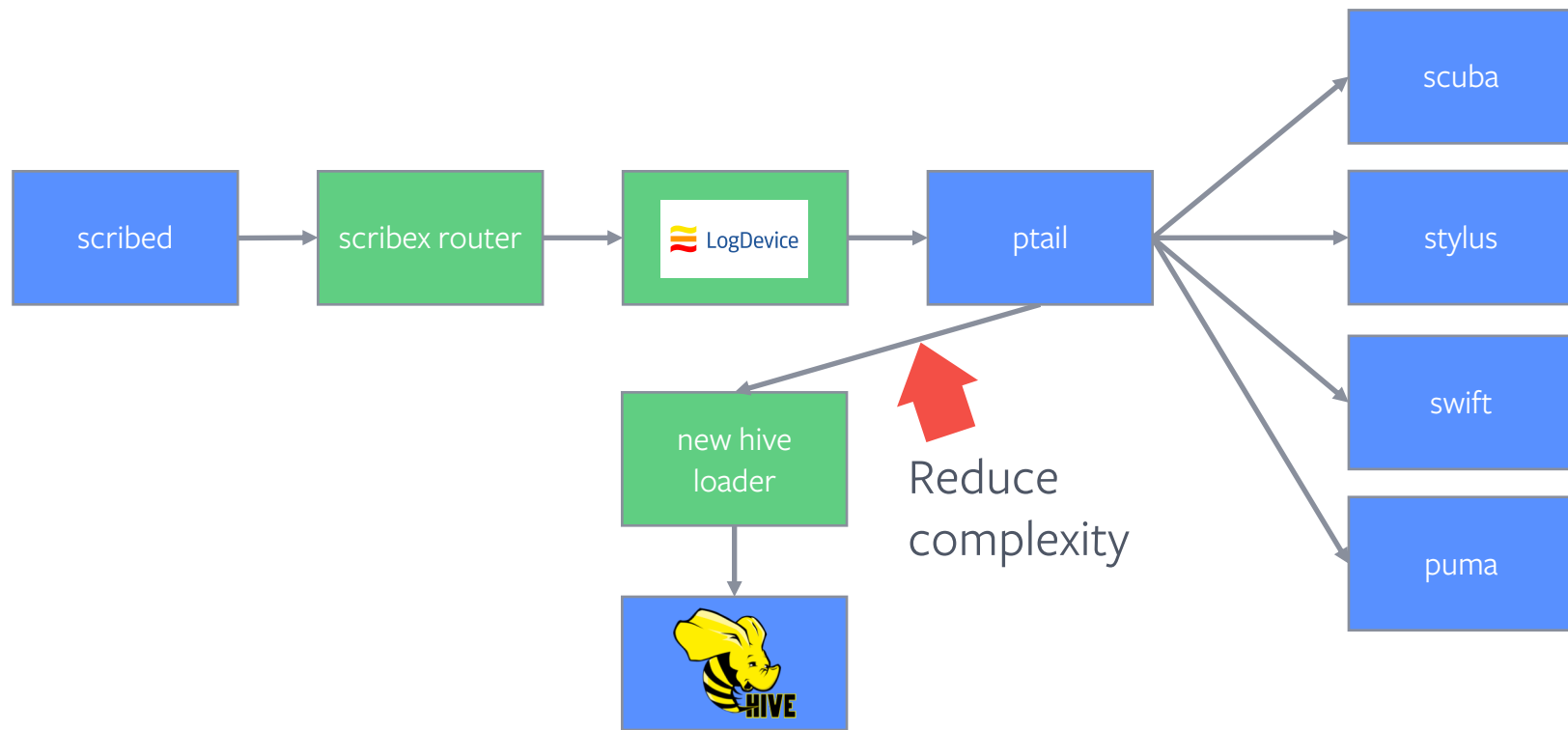
Migration Plan

- 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*

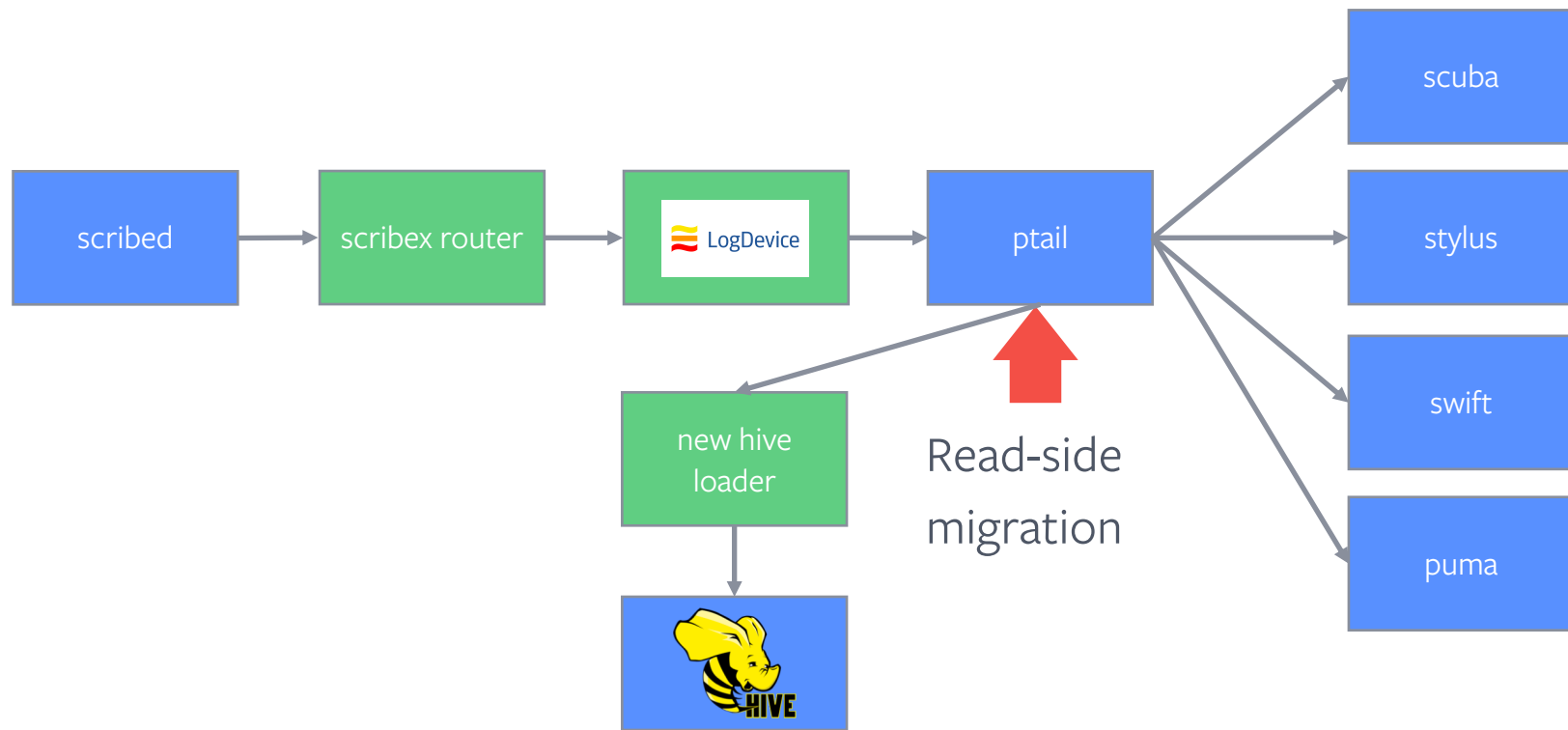
Scribe Ecosystem with LogDevice



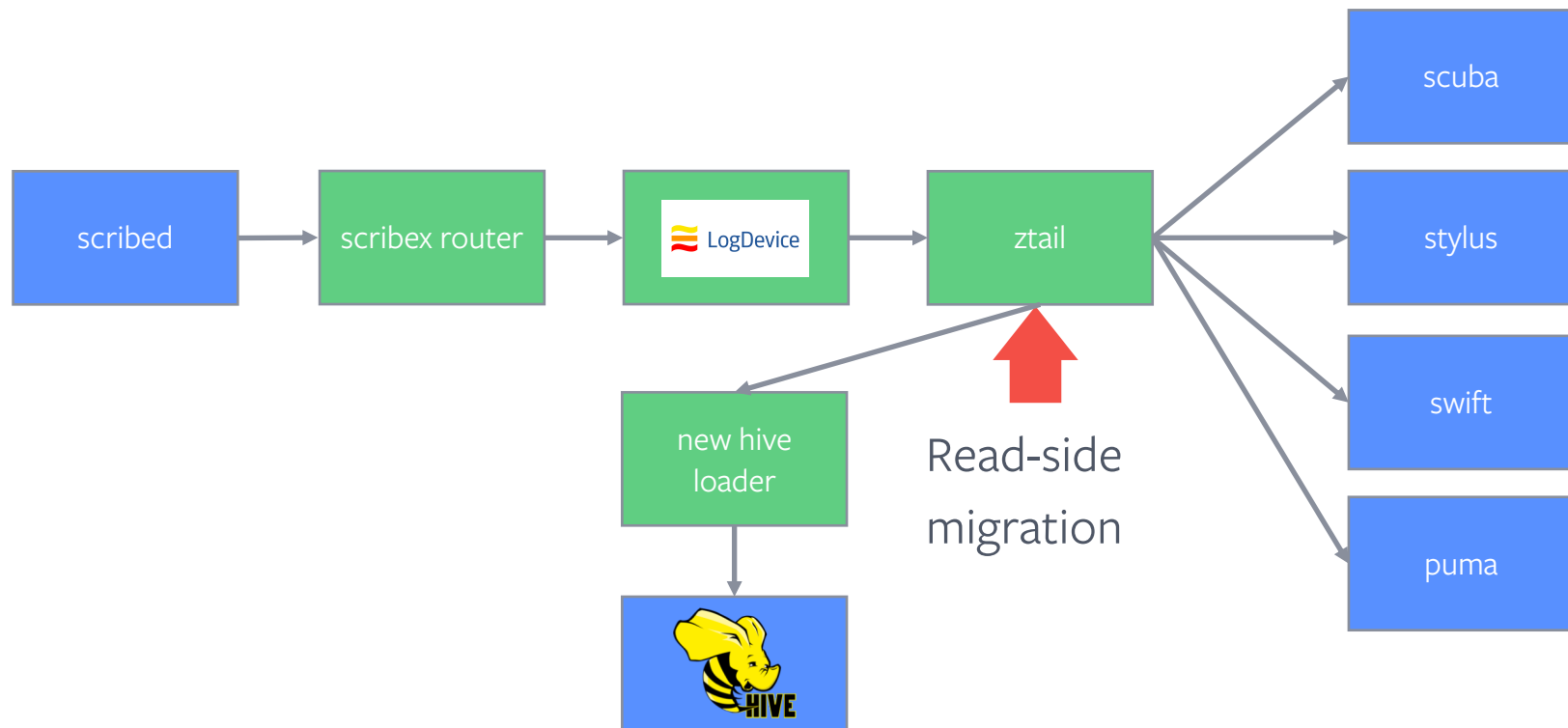
Scribe Ecosystem with LogDevice



Scribe Ecosystem with LogDevice



Scribe Ecosystem with LogDevice



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`)

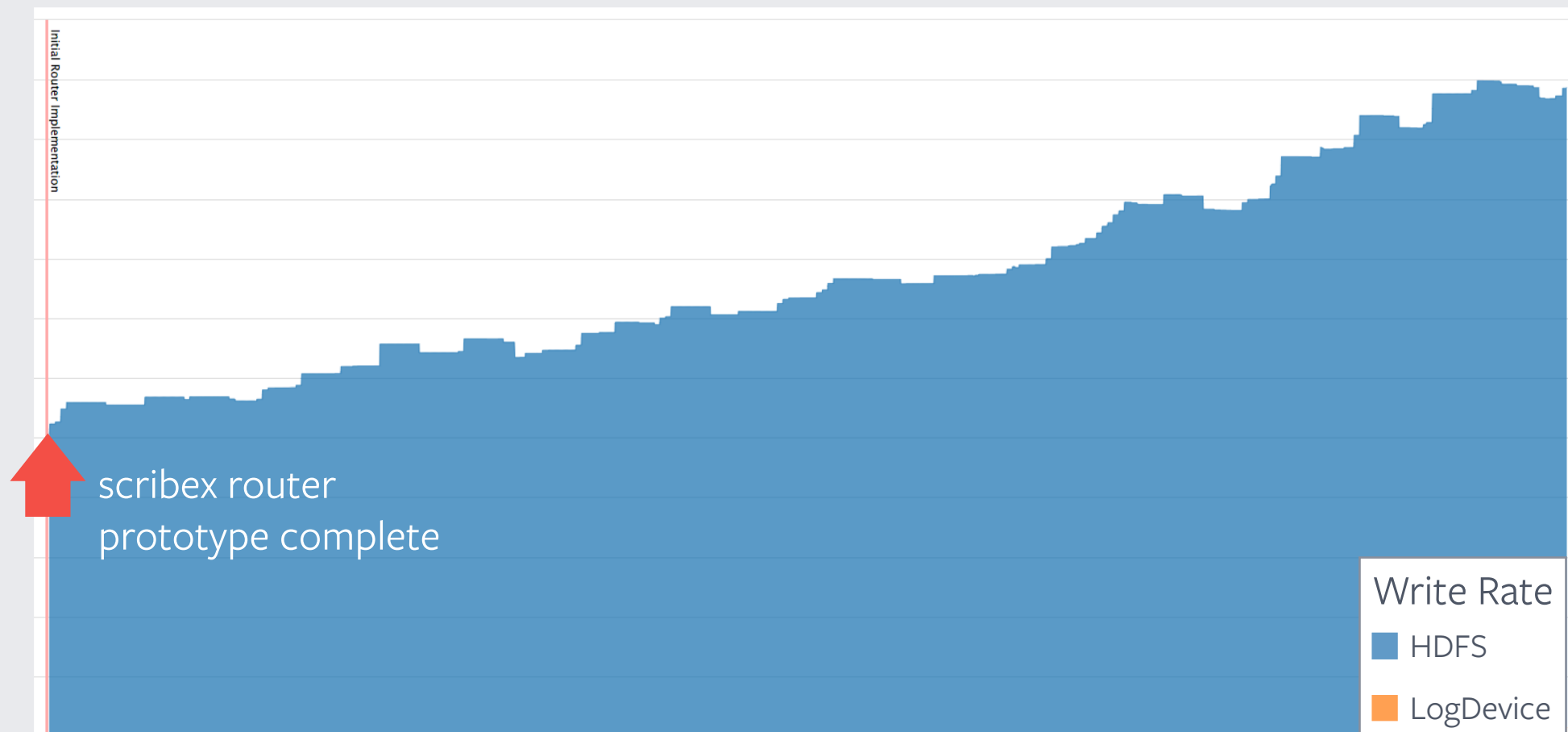
Scribe on LogDevice 2015-2018

A 6 month project you say?

- New components: scribex router, ztail, LogDevice, hive loader
 - Write code
- Migration Plan
 - Leverage our constraints
- **Productionize**

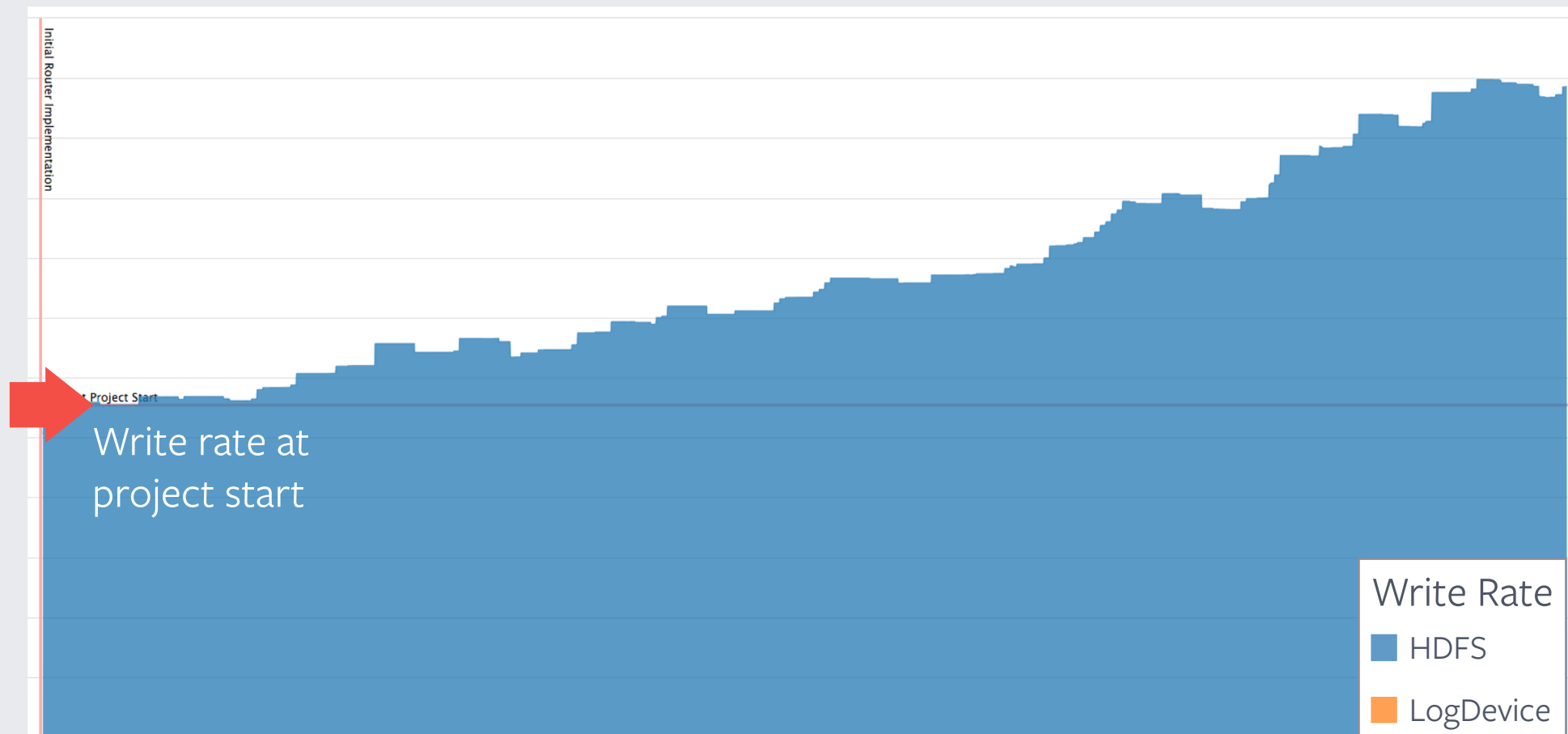
Scribe on LogDevice 2015-2018

Prototype implementations



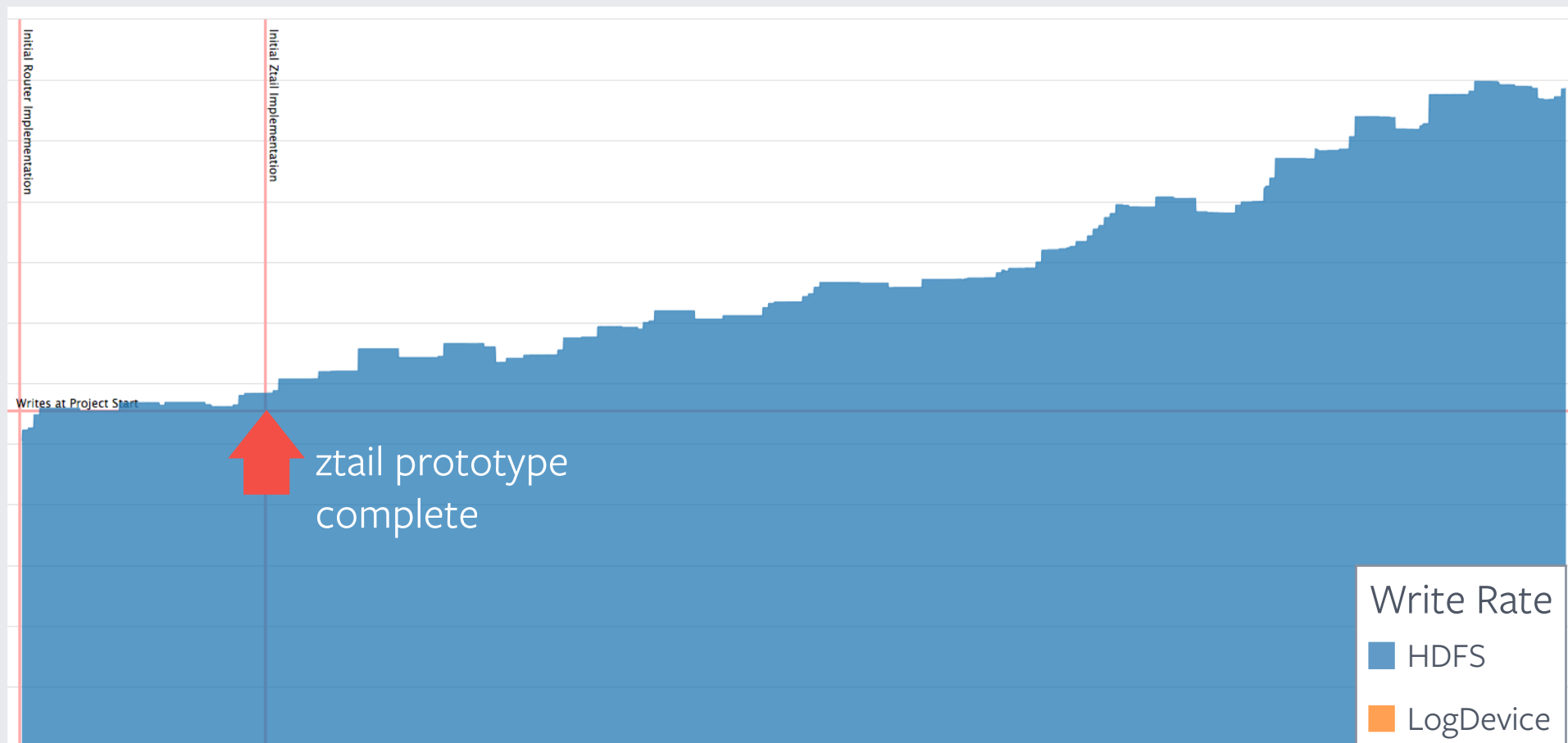
Scribe on LogDevice 2015-2018

Prototype implementations



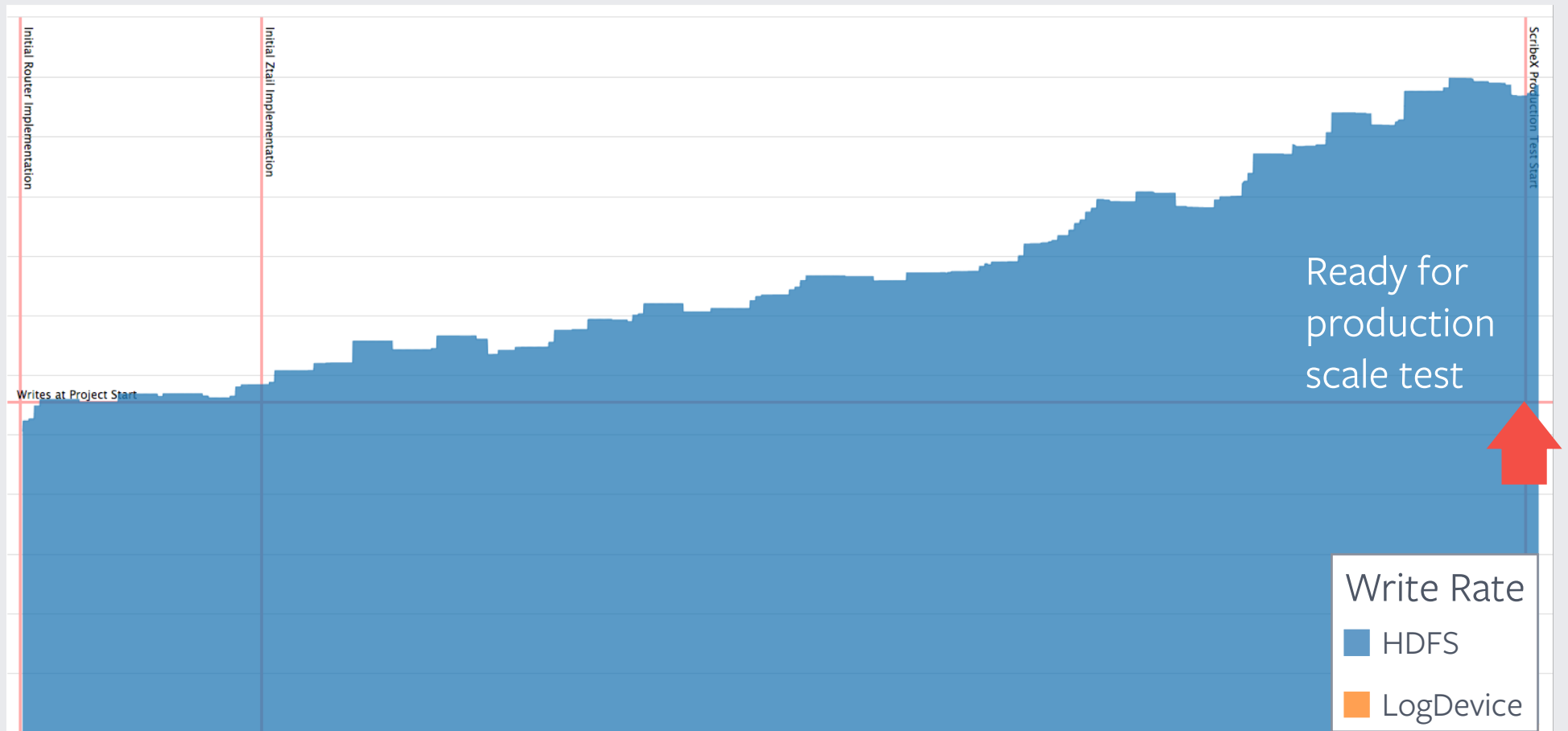
Scribe on LogDevice 2015-2018

Prototype implementations



Scribe on LogDevice 2015-2018

How long could it take to make a plan?



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

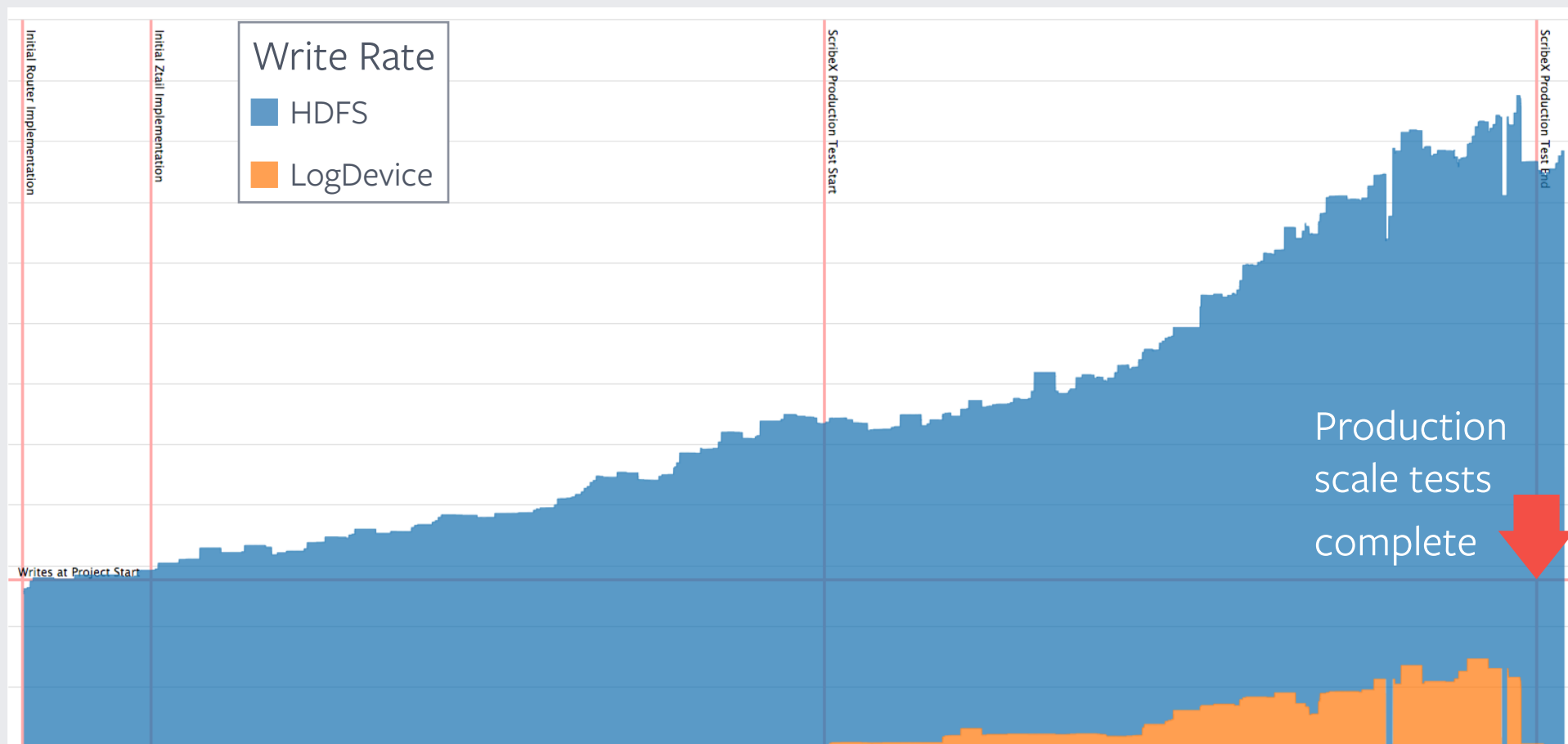
Scribe on LogDevice 2015-2018

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 oncall for the legacy system

Scribe on LogDevice 2015-2018

Testing in prod



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

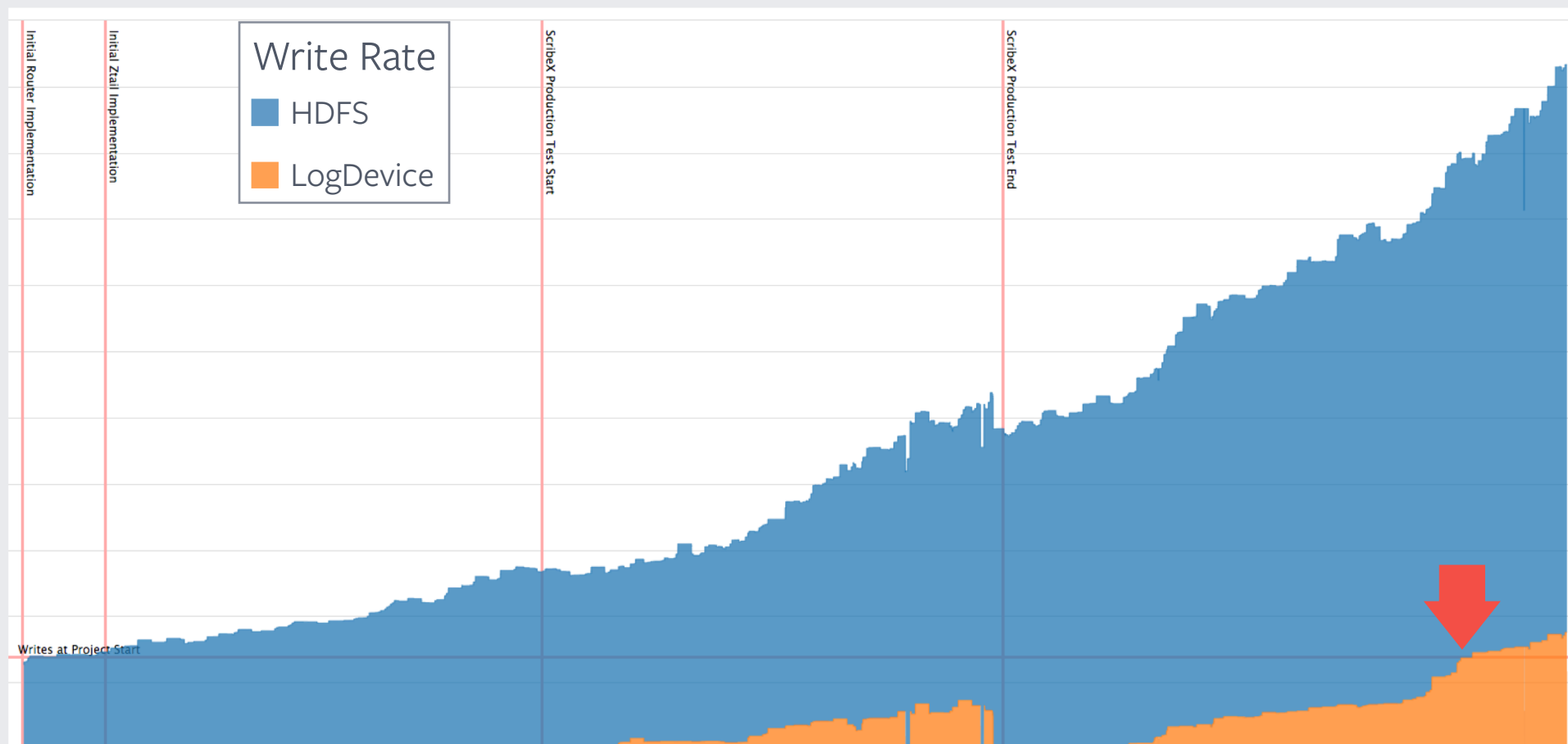
Scribe on LogDevice 2015-2018

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

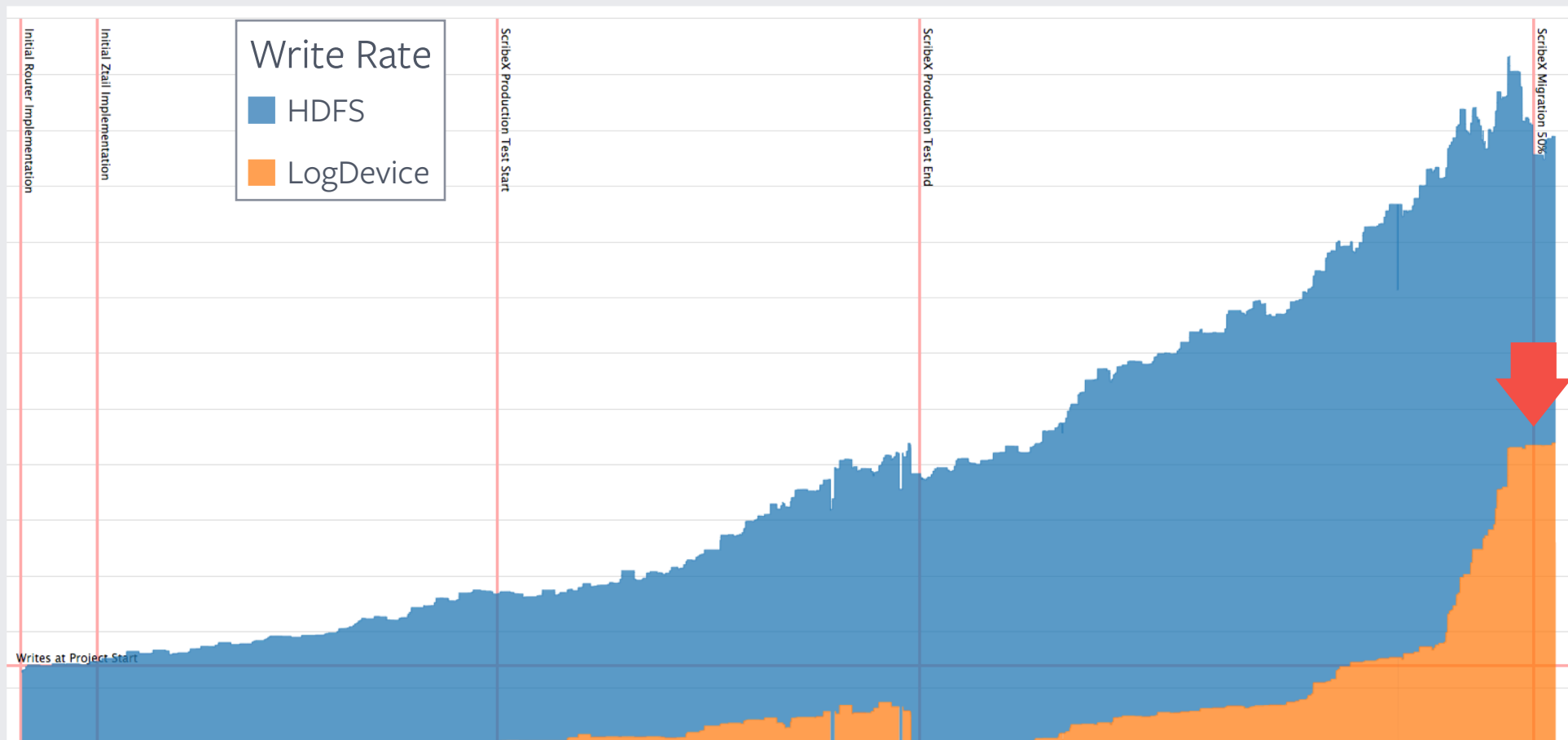
This journey is 15% complete

- 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 (configurator) 

```
{"date": "20171009",  
  "categories": [  
    "pipe_finder",  
    "pipe_indexer",  
    "speedtrap_errors",  
  ],}
```

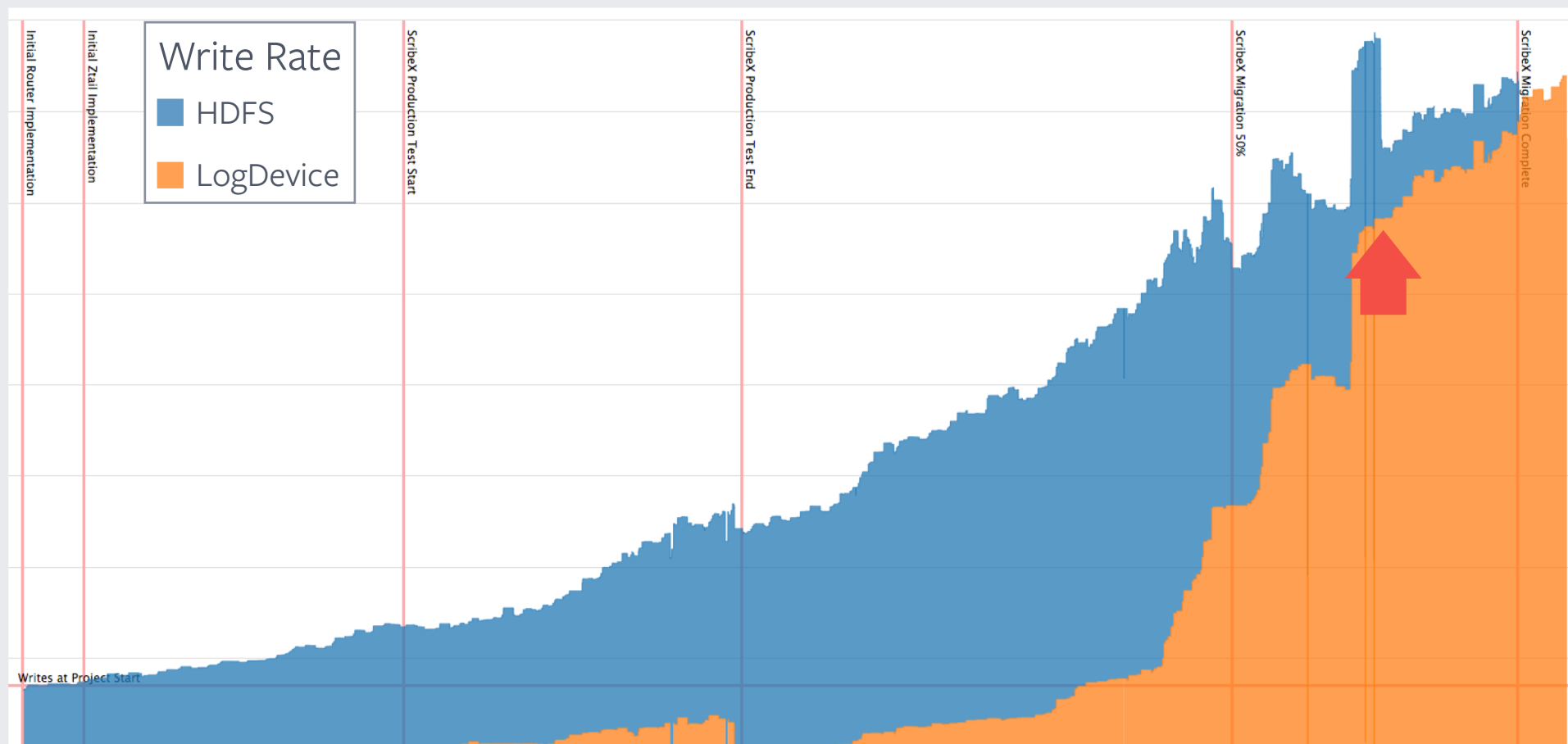

Scribe on LogDevice 2015-2018

Now we're rolling



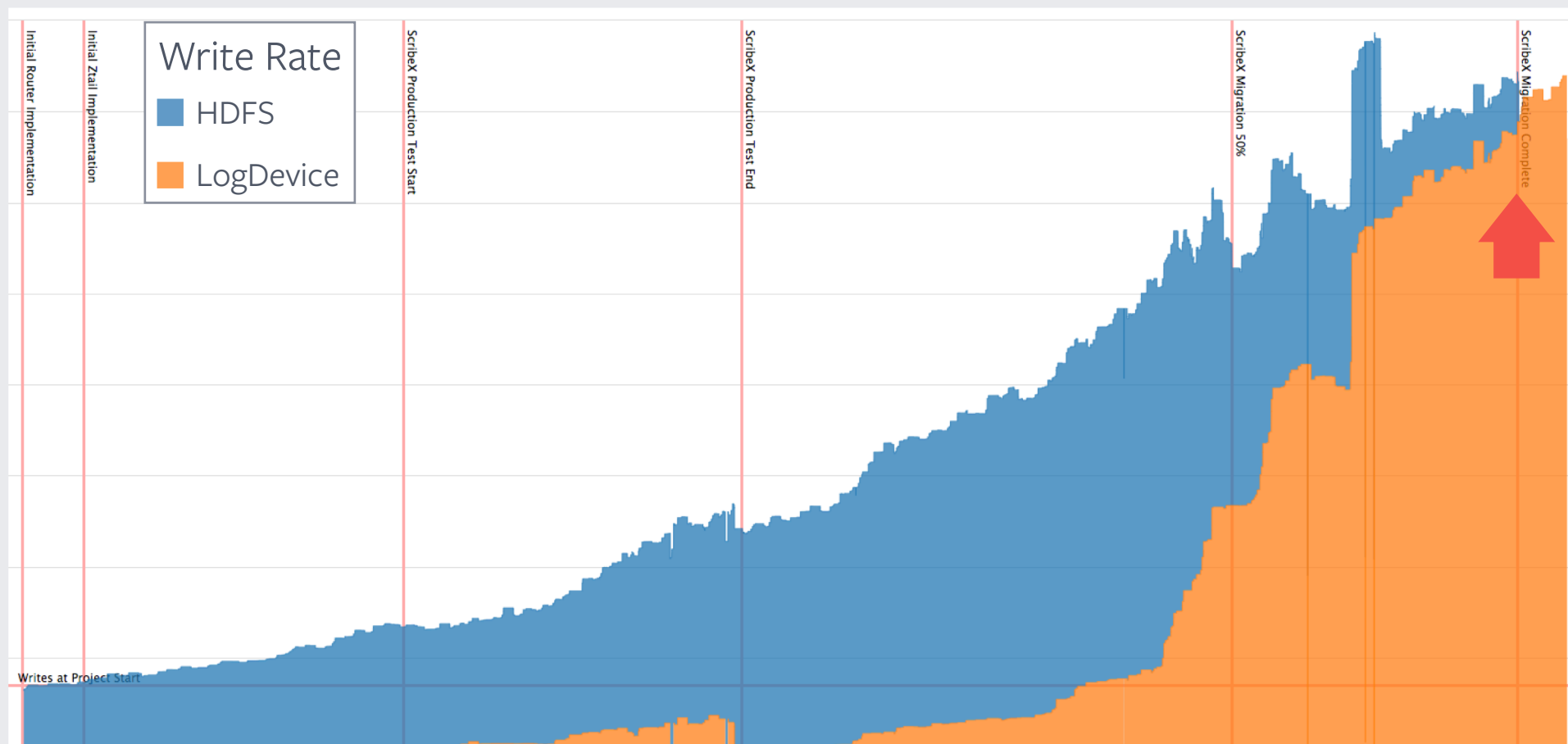
Scribe on LogDevice 2015-2018

Holiday pause



Scribe on LogDevice 2015-2018

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

```
# 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

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


Magic*

Layering is your friend

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

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

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

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
 - configerator - 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
 - configerator - 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
 - configerator - 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

Thank you

facebook

Questions



Reference Links

- Facebook Thrift
 - <https://code.fb.com/open-source/under-the-hood-building-and-open-sourcing-fbthrift/>
- Ticketmaster vs. RMG Technologies
 - https://en.wikipedia.org/wiki/Ticketmaster,_LLC_v._RMG_Technologies,_Inc.
- Scribe Tech Talk 2/27/2009
 - <https://www.facebook.com/Engineering/videos/650882334523/>
- 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)
 - <https://engineering.linkedin.com/distributed-systems/log-what-every-software-engineer-should-know-about-real-time-datas-unifying>



Reference Links

- Realtime Data Processing at Facebook
 - <https://research.fb.com/publications/realtime-data-processing-at-facebook/>
- Open Source LogDevice
 - <https://code.fb.com/core-data/logdevice-a-distributed-data-store-for-logs/>
 - <https://logdevice.io/>
- Rsyslog omprog module
 - <https://www.rsyslog.com/doc/v8-stable/configuration/modules/omprog.html>
- Configerator – Holistic Configuration Management at Facebook
 - <https://research.fb.com/publications/holistic-configuration-management-at-facebook/>

facebook