**Real-world examples of the usage of Apache Thrift**

**FBThrift in Facebook**

Facebook (http://www.facebook.com/) is a company that needs no introduction. With 1.49 billion active users every month, it is used by roughly 20% of the Earth's population. As you know from *Chapter 1*, *Introducing Apache Thrift*, Facebook engineers are the original authors of the framework. They developed it in 2006 and used it as an internal tool to provide application intercommunication. A year later, they decided to open source it and pass it to the Apache Foundation.

So, do they still develop and use it or has it become obsolete? The truth is that there has never been a better time for Thrift in Facebook. A recent article on the company's developer blog (https://code.facebook.com/posts/1468950976659943/) sheds some light on what's going on.

The framework is used in more than 100 production services implemented mostly in C++, Java, PHP, and Python. It is being actively developed and tweaked by their engineers. One of the most remarkable examples is the Messenger mobile application, which uses Thrift to transmit messages between the server and client.

Technically, in Facebook, it is not Apache Thrift. They started their own fork (separate version) of the software and called it **FBThrift**. It is open source too and is available as a repository on GitHub at https://github.com/facebook/fbthrift. This code is independently maintained, but both its authors and Apache Thrift contributors hope to merge it with the original branch. You may want to consider using this version if your needs are similar to those of Facebook.

FBThrift's idea is to upgrade Apache Thrift to provide higher performance and support for complex features. Over time, Facebook's services evolved from the simplest ones to being increasingly complex, with some services calling other services and its latency or memory footprint started to be an issue.

They worked to improve performance, mainly by implementing parallel processing. They also updated the code to use Folly (https://github.com/facebook/folly), Facebook's open source C++ library of components designed keeping performance in mind.

The other need was to provide more complex features needed by their services. The C++ compiler is completely rewritten and available as the cpp2 generator in the FBThrift codebase. Tests indicate that the services moved to the code generated by it showed significantly better performance: latency decreased by up to 50% and there's a much smaller memory footprint.

As FBThrift is the foundation of Facebook's systems architecture, we may expect lots of further improvements in the future.

If you are interested in trying out FBThrift, check out its GitHub page. You will find detailed installation instructions for several popular platforms (CentOS, Ubuntu 12.04, and 14.04).

**Apache Thrift in Evernote**

Evernote (http://www.evernote.com/) is a popular cross-platform service, designed to take notes and organize information. Its applications are present on all major operating systems and mobile devices. Since its start in 2008, it's gained significant popularity with over 11 million users.

The leading feature of the service is its ability to work on one's notes stored in the cloud using different devices or web applications. In the company CTO's article on Evernote Tech Blog (https://blog.evernote.com/tech/2011/05/26/evernote-and-thrift/), we can read some information about the internal process behind choosing Apache Thrift as the best solution for their needs. It is a great insight into what to take into consideration when looking for the most suitable framework for your service.

Evernote's engineers wanted to design a universal API that will serve both thin clients (web browsers, which only serve fragments of the dataset as needed) and thick ones (applications that synchronize the whole user's database at once). They had some requirements that weren't easy to match. First of all, they wanted their API to be cross-platform (they had code in a few different languages, such as Java, C++, and Objective-C), and they wanted to have native bindings for each of them. Another important thing was binary efficiency. As the notes might contain binary attachment, there shouldn't be much overhead (that is, when binary data is encoded using the Base64 encoding, the overhead is 33%, so a 20 MB file is transmitted as a 27 MB payload). Moreover, another important issue was to offer backward compatibility. So, even users using the old version of the software will be able to connect and work with the API. The last two requirements were that the framework should be open source and not too big in terms of code complexity.

They spent a significant amount of time reviewing and testing different options, including some of those that we discussed. When they tested Apache Thrift, they noticed that it fulfills all of their requirements, especially those that are important when building an external API for applications installed on end users' machines: backward compatibility and efficiency in transferring binary data.

As a result, the Evernote Service API (https://dev.evernote.com/doc/) was developed using Apache Thrift. It is an open API that allows developers to build their own applications and there are lots of them. You can check them at https:// appcenter.evernote.com/. The documentation provides good tutorials and code examples, so you can try working with their APIs on your own.

What makes Evernote's usage of Apache Thrift different from Facebook's (and other popular use cases), is that they use it to expose not internal, but public services. This poses more challenges in terms of security, performance, and compatibility. You learned in the previous section of this chapter how to deal with such issues.

**Apache Thrift in Twitter**

Twitter (http://www.twitter.com/) is another extremely popular company that uses Apache Thrift to power many of its internal services. What is interesting and useful for developers is that this company releases lots of the tools that they create as open source, for everyone to use. Here, I will highlight some of the most interesting ones.

Finagle (http://twitter.github.io/finagle/) is one of the most interesting of Twitter's projects. It is a protocol-agnostic, asynchronous RPC system for the JVM (so it can be used for Java, Scala, Clojure, Groovy, and others). It may be considered as an extra level of abstraction above Apache Thrift services because it integrates services using different protocols (not only Thrift, but also MySQL, Mux, HTTP, and so on). You may consider researching this solution if you have some services already in place and would like to integrate them with your new Apache Thrift services. Finagle is actively developed and used in production not only by Twitter itself, but also by well-known companies, such as Foursquare, ING Bank, The New York Times, Pinterest, and Tumblr. You may read about some case studies on the Adopters page at https://github.com/twitter/finagle/blob/master/ ADOPTERS.md.

Scrooge (https://github.com/twitter/scrooge) is a replacement for the original Apache Thrift code generator for Scala and Java, which is recommended for use with Finagle.

Diffy (https://github.com/twitter/diffy) is another great tool from Twitter, useful for everyone writing Apache Thrift and HTTP-based services. The purpose of this tool is to test different versions of the same service, find the differences in them, evaluate the problems that may occur, or help fix bugs. The idea is simple: the old stable (primary) and new (candidate) services are run side by side, and every request sent to Diffy, which acts as a proxy, is passed on to them. Then, the results are compared and potential differences are detected and reported in the form of a nice report. Diffy uses a nice trick to filter out some nondeterministic noises (such as some random results, timestamps, and so on) by sending the payload to the third instance (secondary) for comparison. This tool allows performing tests and development a lot quicker compared to writing traditional integration tests. You will need the **Scala Build Tool** (**SBT**), which you can download from http://www. scala-sbt.org/, to run Diffy.

There are some more Apache Thrift-related tools that you can check out on Twitter. If you wish to start using their toolbox, I suggest browsing their GitHub profile (https://github.com/twitter) for yourself and taking your pick.

**Apache Thrift in other companies**

Apache Thrift is a popular solution among top companies. However, as it is used mainly internally, information on this is rarely public. However, many companies, Twitter and Facebook being notable examples, share their knowledge and the tools that they develop internally so that we can learn a little bit more about their internals and Apache Thrift usage.

Pinterest (https://www.pinterest.com/) is one such enterprise. It runs a website and mobile applications that let users upload, organize, and share photos. It uses Apache Thrift internally, though not much information on this is available. However, it does share useful tools; one of them is the thrift-tools application and library, available at https://github.com/pinterest/thrift-tools. It is a great debugging appliance as it allows you to observe the requests coming to your service and the resulting responses. Let's look at this example from the documentation:

**$ sudo thrift-tool --iface eth0 --port 9091 dump --show-all -pretty**

**[00:39:42:850848] 10.1.8.7:49858 -> 10.1.2.20:3636: method=dosomething, type=call, seqid=1120**

**header: ()**

**fields: [ ( 'struct',**

**1,**

**[ ('string', 1, 'something to do'),**

**('i32', 3, 0),**

**( 'struct',**

**9,**

**[ ('i32', 3, 2),**

**('i32', 14, 0),**

**('i32', 16, 0),**

**('i32', 18, 25)])])]**

**------>[00:39:42:856204] 10.1.2.20:3636 -> 10.1.8.7:49858: method=dosomething, type=reply, seqid=1120**

**header: ()**

**fields: [ ( 'struct',**

**0, [ ('string', 1, 'did something'),**

**('string', 2, 'did something else'),**

**('string', 3, 'did some other thing'),**

**('string', 4, 'did the last thing'),**

**('i32', 6, 3),**

**('i32', 7, 11),**

**('i32', 8, 0),**

**('i32', 9, 0),**

**('list', 10, [0]),**

Note that it requires elevated privileges and uses low-level network tools, so it may or may not work in your environment.

Another interesting tool from Pinterest is the quasar-thrift library (https:// github.com/pinterest/quasar-thrift), which integrates Apache Thrift with Quasar (http://docs.paralleluniverse.co/quasar/), the library that provides high-performance, lightweight threads for JVM languages. This library provides a high-performance server that's fully compatible with Apache Thrift. You may want to consider it if you need highly-performing services in the JVM environment.

Flipkart (http://www.flipkart.com/) is one of the biggest e-commerce businesses in India with 8 million orders monthly and 10 million page views per day. According to a presentation by Siddhartha Reddy, Tech Lead is in charge of search and browse experience at Flipkart (http://www.slideshare.net/sids/how-flipkart-scales-php). It uses the Apache Thrift binary services to support scalability of its PHP applications.

Phantom (https://github.com/Flipkart/phantom) is a tool similar to Twitter's Finagle; it is a high-performance proxy for the accessing of distributed services. It supports not only Apache Thrift but also HTTP, MySQL, and Avro and lets you write your own protocol proxies. You may want to consider it when integrating your Apache Thrift services in an environment where other services are already present.

There are many more well-known companies using Apache Thrift, including Uber, Siemens, Last.fm, and reCAPTCHA. You may check the official lists at https:// thrift.apache.org/about and https://wiki.apache.org/thrift/PoweredBy. I hope that after reading this book, you will be able to use Apache Thrift in your application on a significant scale; don't forget to add in your company's name too!