Transitioning from C# to Scala Using Apache Thrift

and

Twitter Finagle

Steven Skelton

September 19, 2013

Empathica

Empathica provides Customer Experience Management programs to more than 200 of the world's leading brands:

- 30 million customer surveys annually in 25 languages,
- serving 80,000 locations in over 50 countries.

1 year ago:

- Many separate .Net MVC web applications
- Some Java, but most developers strictly C#
- Never heard of Scala

Empathica Adopts Scala

Today:

- New web projects use play
- C# web projects still exist, too large to port
- Single Scala service providing data layer to all products
  - No repeated DB code
  - Single API project: versioned, well documented
  - Apache Thrift

Apache  Thrift

- Open sourced by Facebook in 2007
- Remote Procedure Call (RPC) library
- Supporting clients and servers in
  C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi
- HTML documentation generator
- Supports both binary and JSON protocols
- Mature, used by *a lot* of companies/libraries

Why not REST?

- Internal project
  - RPC is simpler,
  - No benefit in flexible transport
- Consistent across all languages
- Binary transport is more efficient
- No web server - binds to directly to port

Thrift IDL Files

Interface Definition Language:

- Defines all data structures transmitted between client / server
- Defines all methods exposed by server
- Is used only to generate classes
- Same file used for all languages
- Syntax looks a lot like C

.thrift IDL File

```
include “Global.thrift”

enum TweetType {
   TWEET,
   RETWEET = 2,
   DM = 0xa,
   REPLY
}

struct Tweet {
   1: required i32 userId;
   2: required string userName;
   3: required string text;
  18: optional TweetType tweetType = TweetType.TWEET;
}

exception TwitterUnavailable {
   1: Global.ErrorMessage message;
}

service Twitter {
   List<Tweet> last20Tweets(),
   /** Post a Tweet to Twitter */
   bool postTweet(1:Tweet tweet) throws (1:TwitterUnavailable unavailable)
}
```

Twitter made Thrift even better

Finagle

- Generic RPC network stack
- Written in Scala
- Adds Scala idioms to Thrift
  - Futures for Async calls,
  - Options for Optional fields,
  - No Java base types (String, Map, List, Set)

Twitter Scrooge

How?

The only way was to write another Thrift generator.

- Compatible with all Thrift protocols
- Generates .scala files
- Comes with SBT plugin
- Replaces Thrift's network layer with Netty

Finagle uses  Netty

Netty usually benchmarks faster than anything else out there – I'm referring to NodeJS.

Netty supports:

- Zero-copy buffers
- Streaming and pipelined requests
- SSL

Finagle/Netty also have implementations for:

- MySQL, Redis, Protobuf, Memcached, Kestrel

Connection Management

Finagle adds connection options:

- Connection pooling
- Connection timeouts
- Max request time
- Retries / Failover

Ostrich Statistics and Metrics

Tracks usage and performance metrics for all Thrift method calls:

- Time series graphs,
- Easy to add custom metrics
  - ie: time execution of specific blocks of code
- JSON dumps of JVM memory / thread info
- Ability to export (Graphite, gperftools)

Server Clusters

Clients can:

- Can specify a list of hosts, or
- Zookeeper server(s)

Clients will:

- load balance across all available servers,
- redirect requests if a server is down

Apache Zookeeper

- Creates a dynamic server set
  - Add new servers, clients use immediately
  - Crashed servers removed automatically

Zookeeper does much more

- Will find uses if you have lots of servers and/or distributed configuration
- Clients for most languages

Lessons Learned

- Twitter stack full of useful code
- Create separate services; thrift supports multiplexing over a single connection
- Generated classes get their own project, ours takes 2 minutes to compile
- Finagle is a must have, filling in the missing functionality for C# is awful
- Problems? Refer to the source, Twitter has probably already solved it.

Thanks!

Questions?

Steven Skelton

sskelton@empathica.com

http://stevenskelton.ca