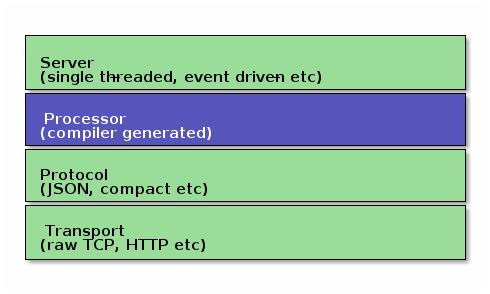
[thrift使用小记](http://blog.chinaunix.net/uid-20357359-id-2876170.html) 2011-09-19 21:28:13

分类： C/C++

    Thrift是一个跨语言的服务部署框架，最初由Facebook于2007年开发，2008年进入Apache开源项目。Thrift通过一个中间语言(IDL, 接口定义语言)来定义RPC的接口和数据类型，然后通过一个编译器生成不同语言的代码（目前支持C++,Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk和OCaml）,并由生成的代码负责RPC协议层和传输层的实现。

    Thrift实际上是实现了C/S模式，通过代码生成工具将接口定义文件生成服务器端和客户端代码（可以为不同语言），从而实现服务端和客户端跨语言的支持。用户在Thirft描述文件中声明自己的服务，这些服务经过编译后会生成相应语言的代码文件，然后用户实现服务（客户端调用服务，服务器端提服务）便可以了。其中protocol（协议层, 定义数据传输格式，可以为二进制或者XML等）和transport（传输层，定义数据传输方式，可以为TCP/IP传输，内存共享或者文件共享等）被用作运行时库。

**基本概念**  
Thrift中的几个概念：  
Server 服务模型  
Handler 数据处理接口  
Processor 数据处理对象  
Protocol 数据传输协议  
Transport 数据传输方式

[](http://blog.chinaunix.net/attachment/201109/20/20357359_1316531363wZjq.jpg)

Handler为抽象接口，需要在编译后的代码上自行实现。Processor调用Handler中的代码，编译自动生成，不用关心。

（1）支持的传输格式  
TBinaryProtocol – 二进制格式.  
TCompactProtocol – 压缩格式  
TJSONProtocol – JSON格式  
TSimpleJSONProtocol –提供JSON只写协议, 生成的文件很容易通过脚本语言解析。  
TDebugProtocol – 使用易懂的可读的文本格式，以便于debug

（2） 支持的数据传输方式  
TFileTransport：文件（日志）传输类，允许client将文件传给server，允许server将收到的数据写到文件中。  
THttpTransport：采用Http传输协议进行数据传输  
TSocket：采用TCP Socket进行数据传输  
TZlibTransport：压缩后对数据进行传输，或者将收到的数据解压

下面几个类主要是对上面几个类地装饰（采用了装饰模式），以提高传输效率。  
TBufferedTransport：对某个Transport对象操作的数据进行buffer，即从buffer中读取数据进行传输，或者将数据直接写入buffer  
TFramedTransport：以frame为单位进行传输，非阻塞式服务中使用。同TBufferedTransport类似，也会对相关数据进行buffer，同时，它支持定长数据发送和接收。  
TMemoryBuffer：从一个缓冲区中读写数据

（3）支持的服务模型  
TSimpleServer – 简单的单线程服务模型，常用于测试  
TThreadedServer - 多线程服务模型，使用阻塞式IO，每个请求创建一个线程。  
TThreadPoolServer – 线程池服务模型，使用标准的阻塞式IO，预先创建一组线程处理请求。  
TNonblockingServer – 多线程服务模型，使用非阻塞式IO（需使用TFramedTransport数据传输方式）

处理大量更新的话，主要是在TThreadedServer和TNonblockingServer中进行选择。TNonblockingServer能够使用少量线程处理大量并发连接，但是延迟较高；TThreadedServer的延迟较低。实际中，TThreadedServer的吞吐量可能会比TNonblockingServer高，但是TThreadedServer的CPU占用要比TNonblockingServer高很多。

**安装与使用**

thrift的安装需要注意实现安装一些库。  
thrift的C++编译器使用了boost中的shared\_ptr，如果需要配合c++使用的话，需要先安装boost。  
如果要使用NonBlockingServer的话，需要安装libevent。

thrift的使用的时候，首先定义一个idl文件(interface description language)，然后使用thrift编译出相应的代码。

thrift --gen cpp XYZ.thrift

我们需要记住的是: Thrift帮你生成了给定Service的服务器端和客户端代码.Thrift这里的命名规则是对于Service XYZ, 它对应的服务器端代码(具体这个Service的执行)在类XYZHandler中,客户端代码(负责marshall, execute RPC)在类XYZClient中. 所以你需要用这个服务, 你只需要直接修改或者继承这些类.

服务器编写的一般步骤：  
1. 创建Handler  
2. 基于Handler创建Processor  
3. 创建Transport  
4. 创建Protocol方式  
5. 基于Processor, Transport和Protocol创建Server  
6. 运行Server

客户端编写的一般步骤：  
1. 创建Transport  
2. 创建Protocol方式  
3. 基于Transport和Protocol创建Client  
4. 运行Client的方法

创建Transport的时候，一般都需要创建相应的Socket。

**示例代码**

附上一份周末写的测试代码，用thrift将leveldb封装了一个网络服务。其中包含各种服务模型。

[kv.rar](http://blog.chinaunix.net/attachment/attach/20/35/73/59203573592988c7e09779f0e118830888244af188.rar)

**需要注意的问题**

1. Thrift生成的server端是thread safe的. 但是client端不是thread safe. 所以需要多个thread和server端通信,则每个thread需要initiate一个自己的client实例.

2. 如果服务器采用TNonblockingServer的话，客户端必须采用TFramedTransport。程序链接的时候需要thriftnb。  
3. 默认TServerSocket和TSocket都设置了NoDelay为1，使得报文尽快发送出去，如果客户端和服务器间传输数据量较大，通过可以设置NoDelay为0来开启Nagel算法，缓存一段数据后再进行发送，减少报文数量。  
TSocket默认开启了Linger，并设置linger time为0，这样close会丢弃socket发送缓冲区中的数据，并向对端发送一个RST报文，close不会被阻塞，立即返回。  
TServerSocket默认关闭了Linger，close不会被阻塞，立即返回。  
4. fb303作为handler的基类，里面预置了一些rpc方法，用于监控，包括系统状态，请求次数等状态信息。  
thrift文件中需要include "fb303.thrift"这样来将service导入目标thrift文件中。thrift编译后的代码只需要相应的Handler多重继承facebook::fb303::FacebookBase就好了。

1. class scribeHandler : virtual public scribe::thrift::scribeIf,
2. public facebook::fb303::FacebookBase {

5. 可以将ZeroMQ等作为transport使用其zeromq来进行通讯。  
 可以参考thrift-0.7.0/contrib/zeromq中的代码。【未进行测试】  
<https://issues.apache.org/jira/browse/THRIFT-812>  
6. thrift支持完全async，生成代码的时候需要使用  
 thrift --gen cpp:cob\_style xxx.thrift  
这样的话，生成的代码需要TEventServer.h，但是async目录下没有，只有TEvhttpServer.h  
<https://github.com/klickverbot/thrift/commit/5ddabb8e3f63a15874e436c9a650dc17f7dd7028#diff-2>

【注意】async有些问题，编译自动生成的代码需要TEventServer.h(0.7.0和svn trunk都是如此)，但是thrift中没有这个文件。contrib/async中有一个http的异步测试代码，大致看了一下实现，使用std::tr1::function和std::tr1::bind实现完成回调函数，实现Processor的异步处理，增加了TAsyncProcessor，其process函数返回的时候，真正的逻辑可能没有完成，依赖完成回调函数处理请求完成的部分，对于服务器端来讲主要是将response发送给客户端。一般RPC业务同步的TProcessor就可以了，只有类似proxy这种中间服务需要异步处理，不过当前thrift中只有TEvhttpServer可用;-)

**参考**  
<http://dongxicheng.org/search-engine/thrift-framework-intro/>  
<http://dongxicheng.org/search-engine/thrift-guide/>  
<http://dongxicheng.org/search-engine/thrift-rpc/>

<http://diwakergupta.github.com/thrift-missing-guide/>