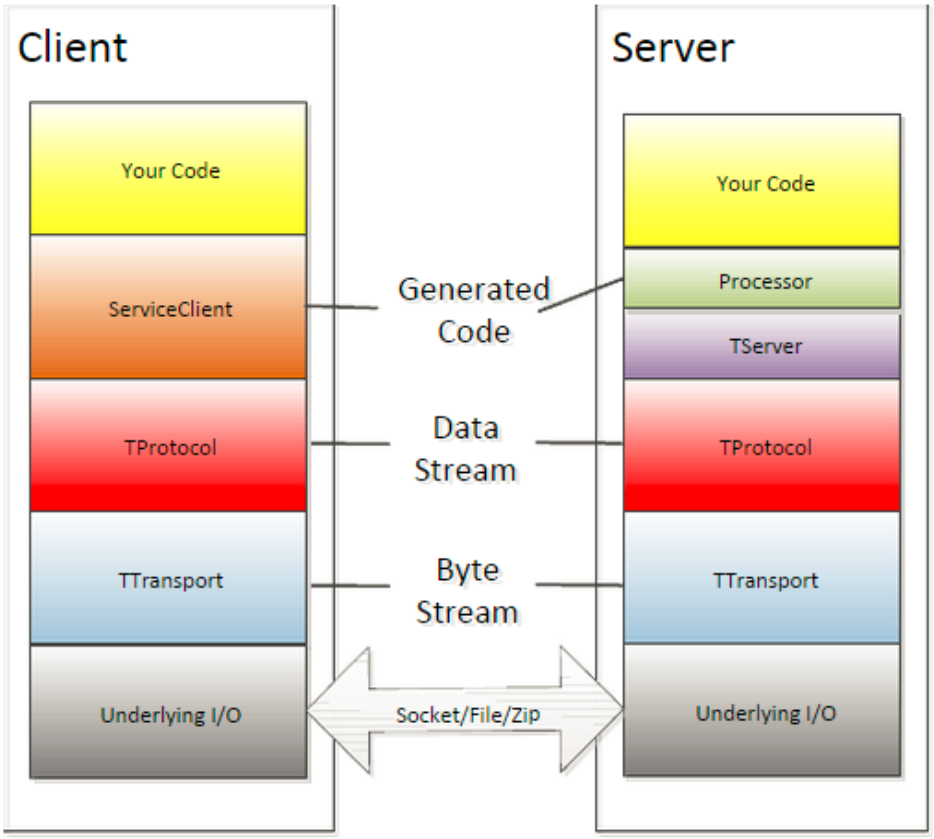
1）Thrift 框架研究

参考 :http://www.blogjava.net/ldwblog/archive/2014/12/03/421011.html

Thrift 是一种开源的跨语言的 RPC 服务框架。在多种不同的语言之间通信 thrift 可以作为二进制的高性能的通讯中间件，支持数据(对象)序列化和多种类型的 RPC 服务。Thrift 是 IDL(interface definition language)描述性语言的一个具体实现，Thrift 适用于程序对程序静态的数据交换，需要先确定好他的数据结构，他是完全静态化的，当数据结构发生变化时，必须重新编辑 IDL 文件，代码生成，再编译载入的流程，跟其他 IDL 工具相比较可以视为是 Thrift 的弱项，Thrift 适用于搭建大型数据交换及存储的通用工具，对于大型系统中的子系统间数据传输相对于 JSON 和 xml 无论在性能、传输大小上有明显的优势。



2）实践

本来觉得自己比较熟悉 java，然后我也不是很喜欢用虚拟机，所以决定在 windows 系统下安装 thrift。

http://wenku.baidu.com/link?url=9KP-k-39TWd_L4CP0LTvtWNy8ju2_GmS2JWpiKUTZuLQmpZuQUws1fBm_YjfxHTxGvbDEA7-Mbi8TuMQMccMS63pnEW5tF1-RztqTy0h_zi

参考的安装手册。

http://blog.csdn.net/jun55xiu/article/details/8988429

参考的使用 Eclipse 平台进行编写的文档

http://blog.csdn.net/jun55xiu/article/details/8985925

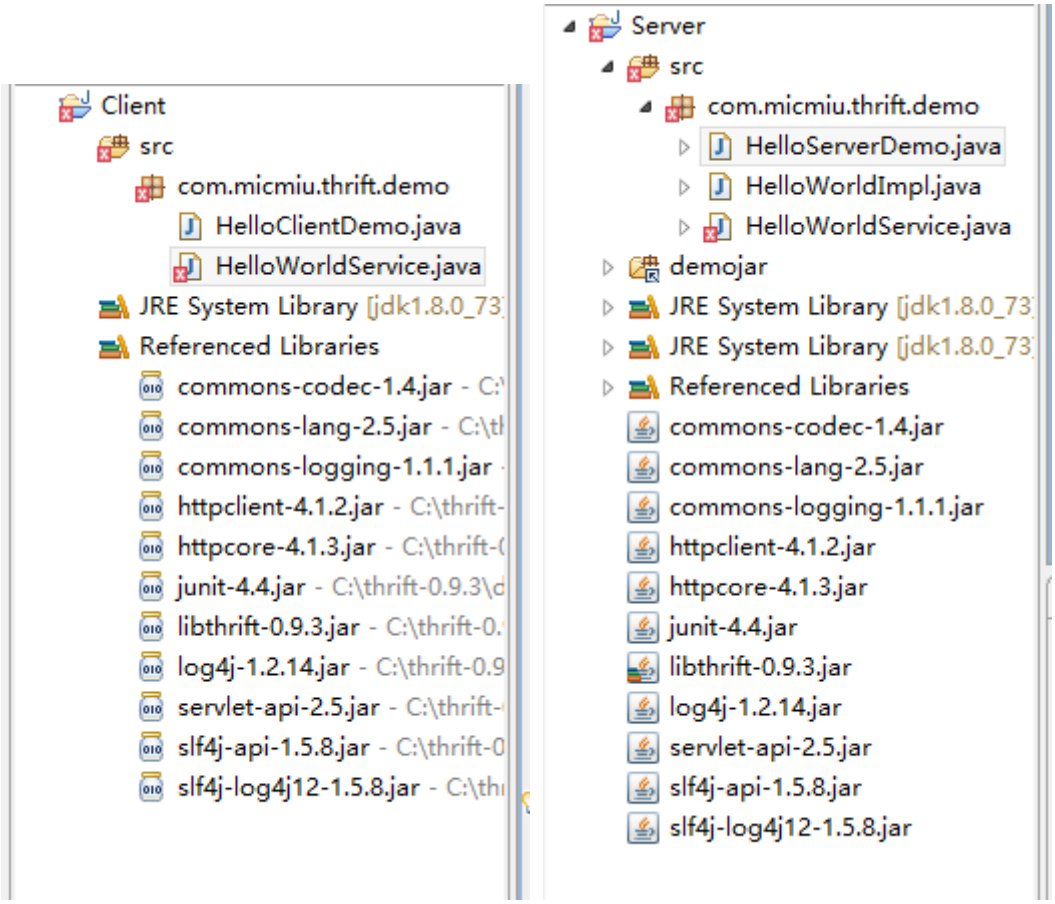
http://www.micmiu.com/soa/rpc/thrift-sample/

结果不是很理想……用 thrift.exe 生成的 demo.java 文件报错……

```
import java.util.EnumSet;
import java.util.Collections;
import java.util.BitSet;
import java.nio.ByteBuffer;
import java.util.Arrays;
import javax.annotation.Generated;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

@SuppressWarnings({"cast", "rawtypes", "serial", "unch
@Generated(value = "Autogenerated by Thrift Compiler (
public class HelloWorldService {

  public interface Iface {
```

也尝试过用 maven……

找了很久也没有找到合适的解决方法，不知道是不是 jre 版本和 thrift 版本之间有什么问题……用的是 1.8 和 0.9.3



折腾了几天实在不知道是哪里的问题……会继续尝试的，不过还是先写报告。

3）大概理解了一下 thrift 框架。

首先主要的功能是远程服务调用，这让我想起了基于 SOAP 的 web service

觉得 thrift 文件某种意义上和 xml 文件有些相似

Thrift 文件描述了方法，包括方法名，参数列表和返回类型。

然后通过下载的 thrift.exe 对这个文件进行编译，会得到一个*.java 文件。包含借口定义和通信细节。

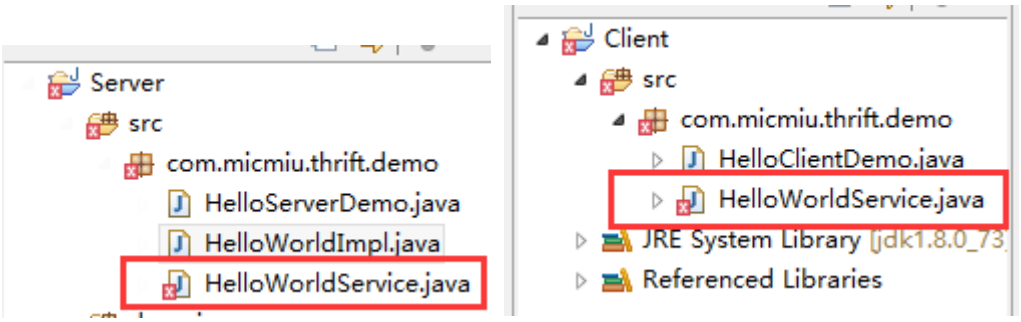
```
public interface Iface {

  public String sayHello(String username) throws org.apache.thrift.TException;

}

public interface AsyncIface {

  public void sayHello(String username, org.apache.thrift.async.AsyncMethodCallback resultHandler) throws org.apache.thrift.TException;

}
```

```
public class HelloWorldImpl implements HelloWorldService.Iface {
  public HelloWorldImpl(){

  }

  public String sayHello(String username)throws TException{
      return "Hi,"+ username + "welcome";
  }
}
```

对应的在 Server 端的实现。

```
      // TProtocol protocol = new TCompactProtocol(transport);
      // TProtocol protocol = new TJSONProtocol(transport);
      HelloWorldService.Client client = new HelloWorldService.Client(
              protocol);
      transport.open();
      String result = client.sayHello(userName);
      System.out.println("Thrify client result =: " + result);
  } catch (TTransportException e) {
      e.printStackTrace();
  } catch (TException e) {
```

对应在 Client 端的调用。



这两个就是通过 thrift.exe 生成的文件，通过它我们可以生成不同语言的文件，但是他们都对应相同的方法接口 ,这样我们就可以实现在服务器端和客户端跨语言的支持。

他支持的传输格式和数据传输方式也很多 ,就比如很多示例代码中给出的是这样的

```
// 简单的单线程服务模型，一般用于测试
TServerSocket serverTransport = new TServerSocket(SERVER_PORT);
TServer.Args tArgs = new TServer.Args(serverTransport);
tArgs.processor(tprocessor);
tArgs.protocolFactory(new TBinaryProtocol.Factory());
// tArgs.protocolFactory(new TCompactProtocol.Factory());
// tArgs.protocolFactory(new TJSONProtocol.Factory());
TServer server = new TSimpleServer(tArgs);
```

具体的 RPC 实现是封装的 0.0 不过我们可以通过结构图来看看。大致的结构图可以看我最上面贴出来的。具体的运作过程我觉得

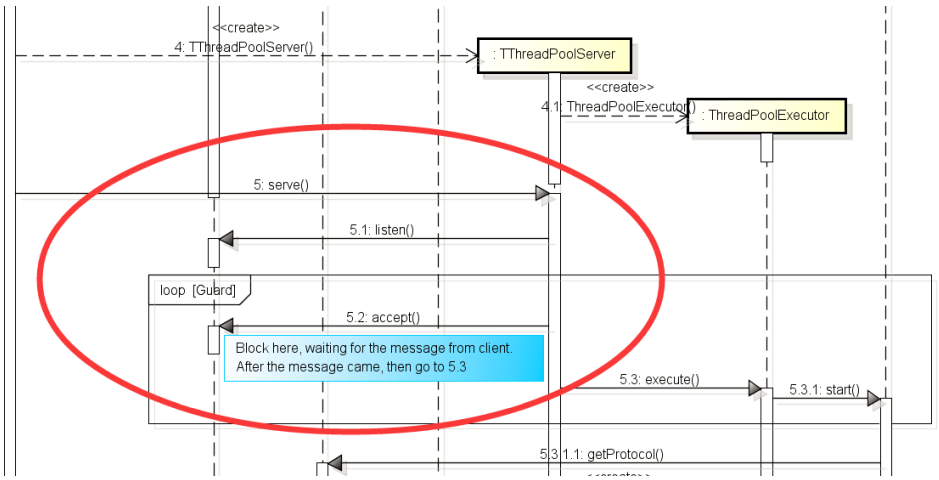
http://www.ibm.com/developerworks/cn/java/j-lo-apachethrift/

可以参考这个文档~

很巧的是大多数文档都很喜欢用 helloworld 作为例子讲解 ,方便了学习的过程。

总结一下 0.0 先运行服务器端时

【thrift 专递的是二进制文件，所以传输速度比较快】



程序调用了 TThreadPoolServer 的 serve 方法后 ,server 进入阻塞监听状态。当接收到来自客户端的消息后，服务器发起一个新线程处理这个消息请求，原线程再次进入阻塞状态。在新线程中，服务器通过 TBinaryProtocol 协议读取消息内容，调用 HelloServiceImpl 的 helloVoid 方法，并将结果写入 helloVoid_result 中传回客户端。

Thrift 也提供非阻塞的调用方式，可构建异步客户端。在这种方式中，Thrift 提供了新的类 TAsyncClientManager 用于管理客户端的请求，在一个线程上追踪请求和响应，同时通过接口 AsyncClient 传递标准的参数和 callback 对象，服务调用完成后，callback 提供了处理调用结果和异常的方法。

创建非阻塞服务器端实现代码，将 HelloServiceImpl 作为具体的处理器传递给异步 Thrift 服务器。

在文档中还提到了 null 的问题，虽然没有遇到还是马克一下

常见问题

NULL 问题

我们在对服务的某个方法调用时，有时会出现该方法返回 null 值的情况，在 Thrift 中，直接调用一个返回 null 值的方法会抛出 TApplicationException 异常。在清单 2 中，HelloServiceImpl 里实现了 helloNull 方法，返回 null 值，我们在 HelloServiceClient.java 中加入调用该方法的代码，出现如下图所示的异常：

图 4. TApplicationException 异常

```
org.apache.thrift.TApplicationException: helloNull failed: unknown result
        at service.demo.Hello$Client.recv_helloNull(Hello.java:257)
        at service.demo.Hello$Client.helloNull(Hello.java:228)
        at service.client.HelloServiceClient.main(HelloServiceClient.java:29)
```

为了处理返回 null 值情况，我们要捕获该异常，并进行相应的处理，具体客户端代码实现如下：

用c++的时候，service名为hello，不能有helloNull的接口，否则编译不过，原因是thrift会生成helloNull类
java不知道是否有此问题
thrift版本是0.9.1

由 hzw-up 于 2014年02月09日发布　　报告滥用

在一些评论里我也收获到很多有趣的信息，比如有人为了比较 thrift 和 web service 的传输速度做了一个小实验 0.0 有时间我也想自己尝试一下。虽然过程不太顺利，但是觉得这个学习的过程还是有点乐趣的 0.0