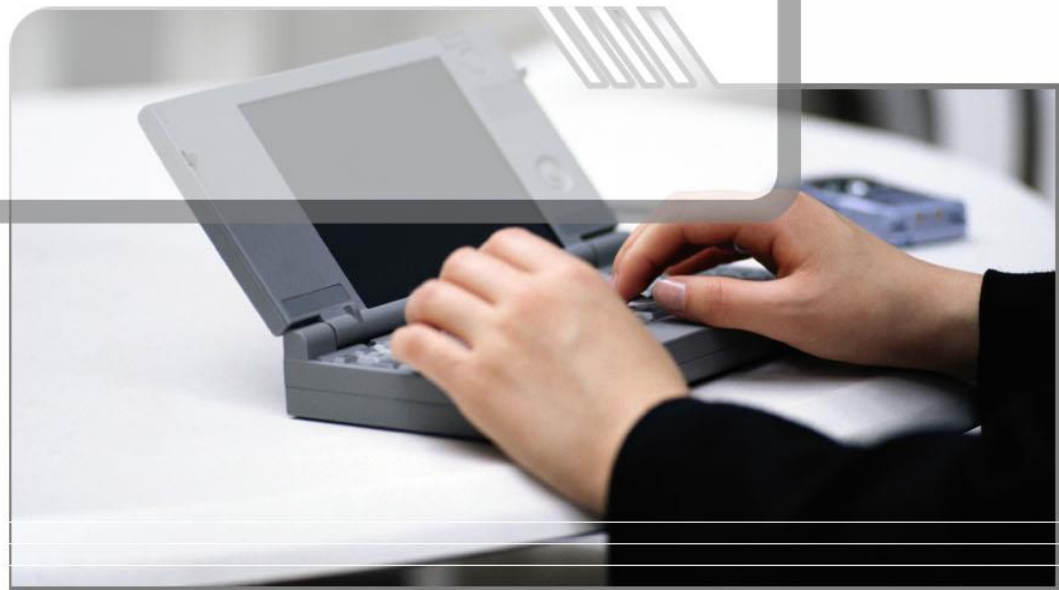
RPC了解一下



一、RPC是啥？

二、听说thrift不错？

三、看看例子怎么样？

四、总结一下

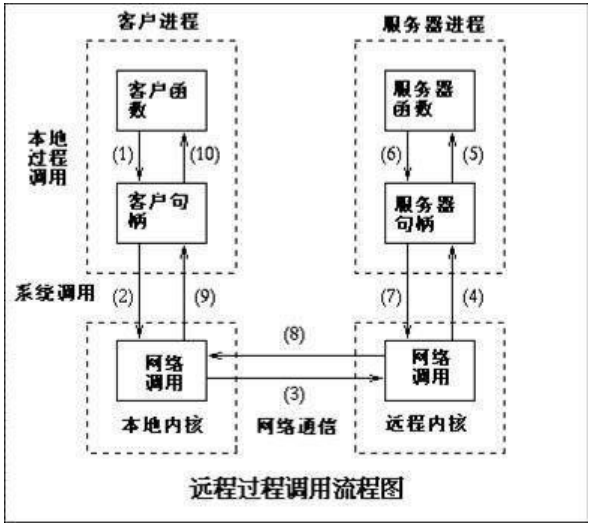
RPC是啥？

在了解RPC之前，我们先来看看互联网架构发展

RPC是啥？

RPC(Remote Procedure Call，远程过程调用)最早可以追溯到快跨越半个世纪的1974年时发布的RFC 674草案，这个草案并不是很出众的一个，但是确确实实的拉开了RPC的序幕。

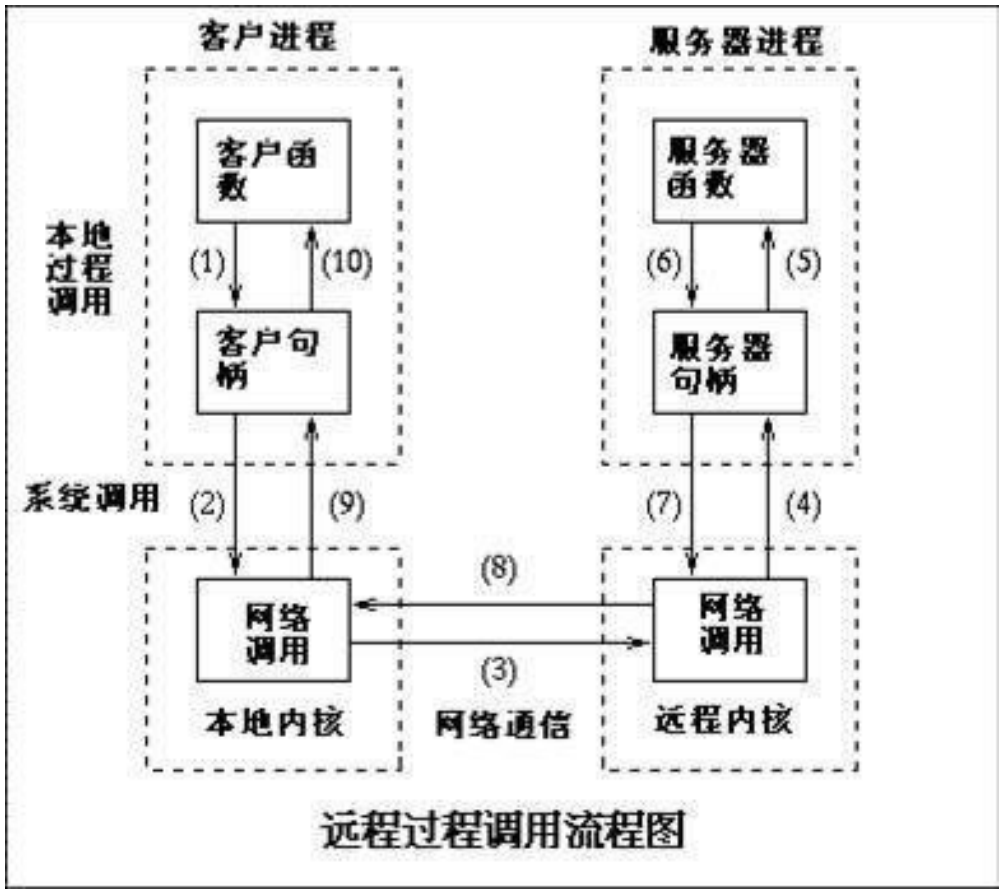
在分布式计算中，RPC是指计算机程序在不同的计算机之间调用、传递信息时，就好像在本地调用、传递信息一样。[1]



远程过程调用流程图

注：[1] Remote procedure call - Wikipedia

RPC是啥？



远程过程调用流程图

1.调用客户端句柄；执行传送参数

2.调用本地系统内核发送网络消息

3.消息传送到远程主机

4.服务器句柄得到消息并取得参数

5.执行远程过程

6.执行的过程将结果返回服务器句柄

7.服务器句柄返回结果，调用远程系统内核

8.消息传回本地主机

9.客户句柄由内核接收消息

10.客户接收句柄返回的数据

RPC框架实现的几个核心技术点：

1. 服务定义

远程提供者需要以某种形式提供服务调用相关的信息，包括但不限于服务接口定义、数据结构、或者中间态的服务定义文件。

例如Facebook的 Thrift的IDL文件，Web service的WSDL文件；Google的ProtoBuf定义文件。

2. RPC通信

通信方法

RPC肯定会涉及通信的问题，不同的RPC框架所用的通讯方法也是不尽相同的。常用的通信方法有：TCP、UDP、HTTP、HTTP/2。

但是，需要说明的是直接使用TCP和HTTP的在性能上面并没有明确的差异。

RPC是啥？

RPC序列化

序列化是之将一个对象以二进制或者字节的方式呈现，反序列化则是逆过程。RPC之间的数据在传输的时候则需要将要发送的内容进行序列化，服务端则执行反序列化执行数据。
因为是远程通信，需要将对象转化成二进制流进行传输。不同的RPC框架应用的场景不同，在序列化上也会采取不同的技术（跨语言）

常见的序列化协议规范：

- COM （微软出品，Windows专享）
- CORBA （早期的跨平台，跨语言协议）
- RMI（Java改进版CORBA）
- ProtoBuf（谷歌出品，必属精品）
- TProtocol （Thrift内置）

RPC是啥？

RPC风险思考

- 意外中断
- 中间人攻击
- 信息伪造

RPC是啥？

RPC应用例子

知乎：

知乎在一开始做的时候就直接大规模的使用SOA的微服务理念，使得其中任何一个模块在更换语言或者出现异常时都可以做到外部无感知。比如知乎社区、评论、问答都是分离的部分，但是当其模块之间需要通信的时候，就会使用RPC和HTTP两种方式。官方说在社区因为QPS很高的地方，只使用了RPC。

注：参考 https://zhuanlan.zhihu.com/p/48039838

RPC延伸一下

据说RESTful好像也可以做通信？

在微服务架构中，同步的有两种主流的候选协议，分别是RPC和RESTful。

	RPC	RESTful
载体	TCP/UDP协议	HTTP
语言支持	很多语言	几乎全部
带宽占用	0.2～0.5（Binary）	1
性能	更高	高
简洁性	简洁	更简洁
面向对象	服务	资源
规范	自定义规范	有明确规范
特点	框架简单	可以无框架

注：参考 https://zhuanlan.zhihu.com/p/48039838

RPC延伸一下

异步的不可以吗？

在微服务架构中，异步有AMQP、STOMP、RabbitMQ、Apache Kafka

	同步	异步
复杂程度	复杂	更复杂
语言支持	很多语言	很多语言
特点	高耦合度，异常处理较少，业务应用简单	低耦合度，高并发场景应用，业务处理复杂

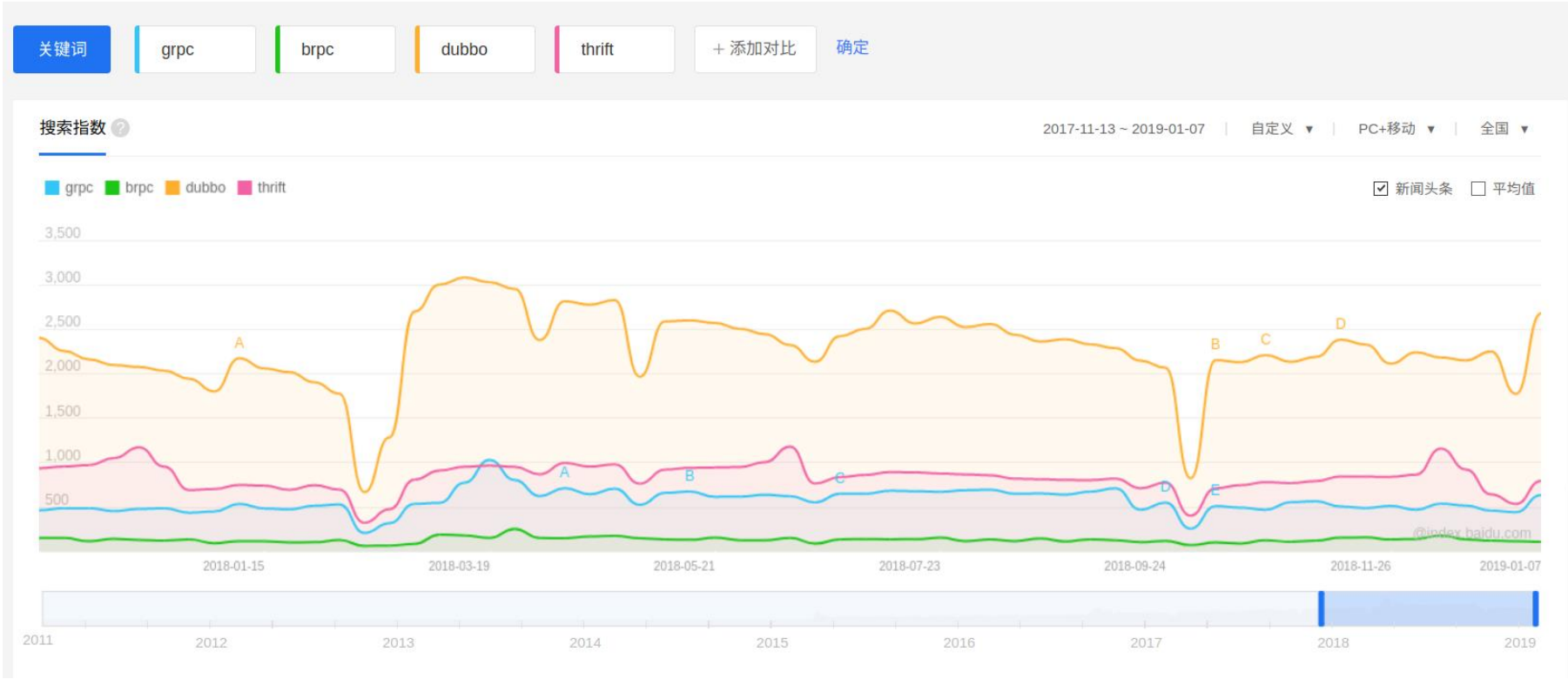
注：参考 https://zhuanlan.zhihu.com/p/48039838

RPC是啥？

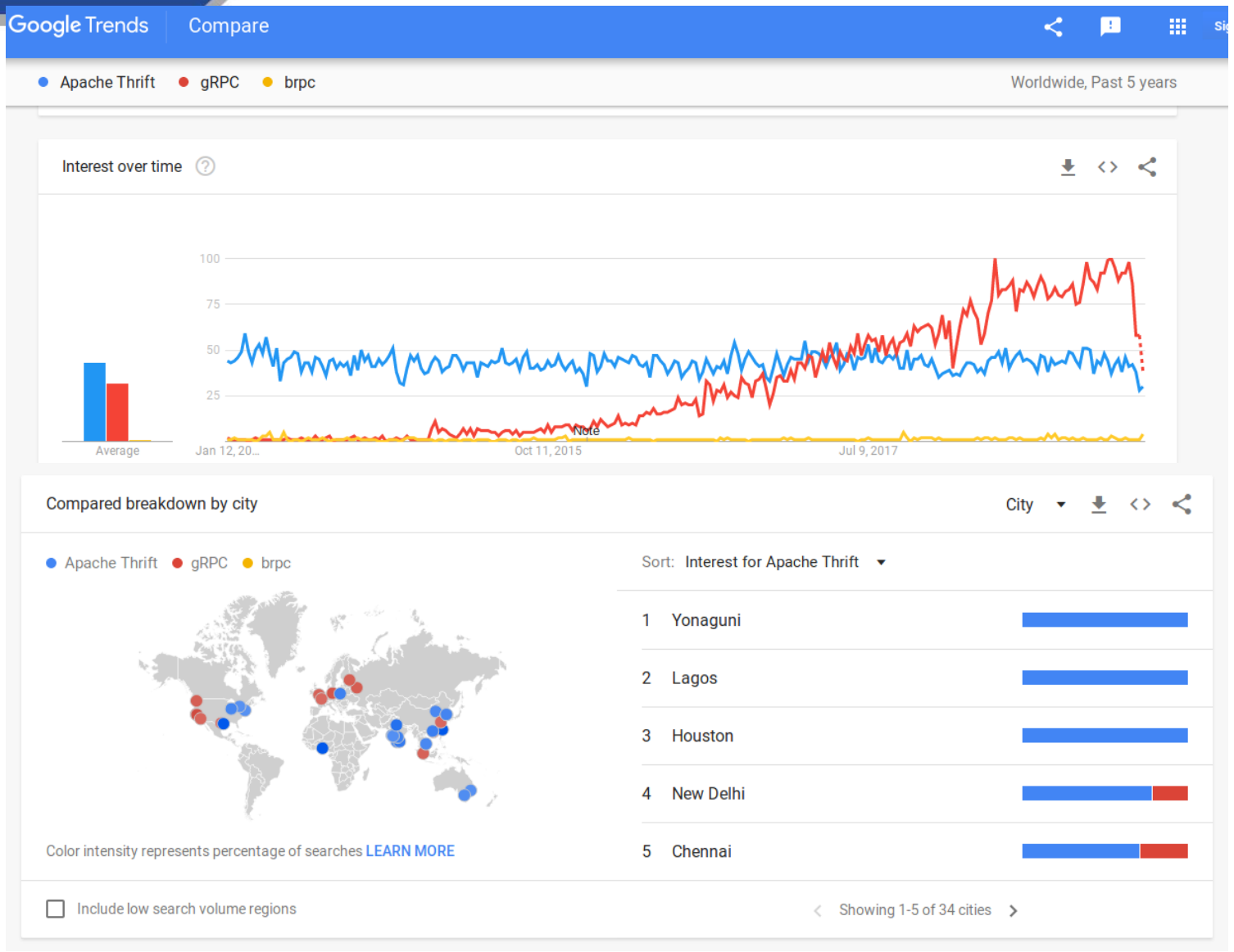
RPC框架有那些？

- COM+ ：Microsoft出品，仅用于Windows平台
- RMI： Java内置
- .NET Remoting：Net专享
- XML-RPC，SOAP，WebService 性能都是硬伤
- gRPC：Google开发，基于ProtoBuf，HTTP/2协议，多语言支持
- Thrift：Facebook开发，基于TProtocol协议等，多语言支持
- Dubbo：taobao开发，多协议支持，主要面向spring，语言支持较少
- Hessian：caucho开发，基于HTTP协议修改，多语言支持
- brpc：百度开发，基于ProtoBuf，多协议支持，仅开源C++版本

RPC是啥？


关键词
grpc
brpc
dubbo
thrift
＋添加对比
确定
搜索指数
2017-11-13 ~ 2019-01-07
自定义
PC+移动
全国
grpc brpc dubbo thrift
新闻头条
平均值
3,500
3,000
2,500
2,000
1,500
1,000
500
2018-01-15
2018-03-19
2018-05-21
2018-07-23
2018-09-24
2018-11-26
2019-01-07
2011
2012
2013
2014
2015
2016
2017
2018
2019

RPC是啥？


Google Trends
Compare
Apache Thrift
gRPC
brpc
Worldwide, Past 5 years
Interest over time
100
75
50
25
Note
Average
Jan 12, 20...
Oct 11, 2015
Jul 9, 2017
Compared breakdown by city
City
Apache Thrift
gRPC
brpc
Sort: Interest for Apache Thrift
1 Yonaguni
2 Lagos
3 Houston
4 New Delhi
5 Chennai
Color intensity represents percentage of searches LEARN MORE
Include low search volume regions
Showing 1-5 of 34 cities

RPC是啥？

总上所述，我们对前两个RPC框架进行了一次性能的测试，测试环境：

OS X(8G内存，Intel Core i5)
Python 3.6
gRPC 1.17.1
Thrift 0.11.0

框架	服务端语言	客户端语言	10000次	平均时间
gRPC	Python	Python	61.808	0.006
Thrift	Python	Python	49.563	0.004
Thrift	C++	Python	28.254	0.002

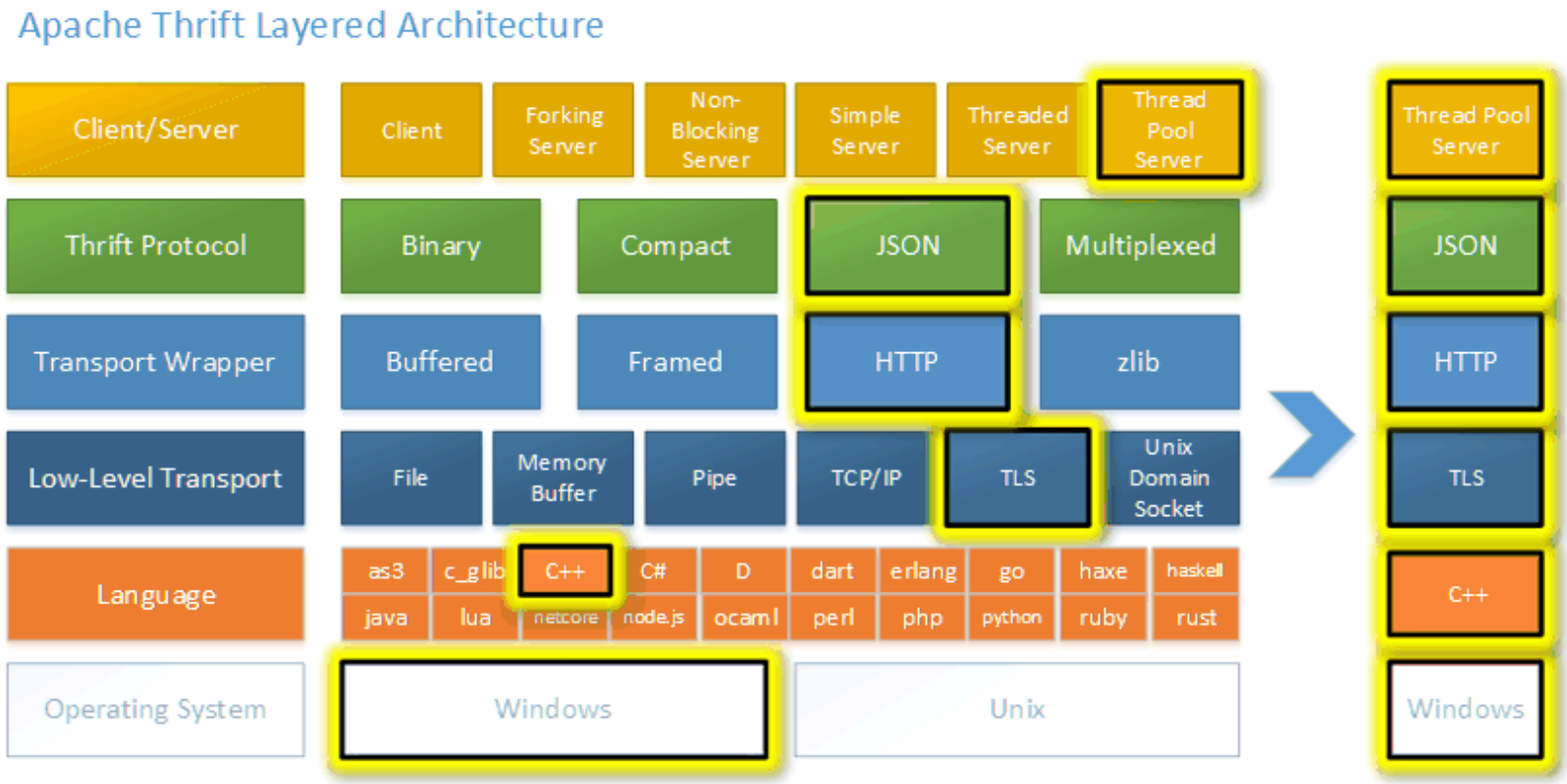
实测结果，我们在下面将详细讲解Thrift框架。

注：所有测试均为单进程锁测试，代码仓库见https://github.com/ctudoudou/demonstration-RPC

听说thrift不错？

Apache Thrift 是 Facebook 实现的一种高效的、支持多种编程语言的远程服务调用的框架。Thrift是由Facebook开发的，并在2008年捐给了Apache基金会，成为了一个孵化器项目。

官方网站：thrift.apache.org



听说thrift不错？

支持的类型

标签	类型	说明
bool	布尔	布尔类型，占一个字节
byte	字节	有符号字节
i16	整型	16位有符号整型
i32	整型	32位有符号整型
i64	整型	64位有符号整型
double	浮点型	64位浮点数
string	字节	未知编码或者二进制的字符
list	容器	有序列表
set	容器	无序列表
map	容器	键值对存储

传输的协议

- TBinaryProtocol
  是Thrift的默认协议，使用二进制编码格式进行数据传输，基本上直接发送原始数据

- TCompactProtocol
  压缩的、密集的数据传输协议，基于Variable-length quantity的zigzag 编码格式

- TJSONProtocol
  以JSON (JavaScript Object Notation)数据编码协议进行数据传输

- TDebugProtocol
  常常用以编码人员测试，以文本的形式展现方便阅读

看看例子怎么样？

定义thrift文件

```
struct FileData {       // 定义文件数据结构
    1:string name,      // 文件名
    2:binary buff       // 文件数据流
}

service Filebuff {
    string predict(1:FileData file)  // 使用文件结构
}
```

看看例子怎么样？

这是本地调用的方法

```
boxs = [(5, 1, 17, 21), (17, 1, 29, 21), (29, 1, 41, 21), (41, 1, 53, 21)]
app = load_model('../model/net.h5')
while True:
    name = input("filename: ")
    img = Image.open(name).convert('L').convert('1')
    name = ''
    for x in range(len(boxs)):
        aaa = []
        roi = img.crop(boxs[x])
        roi = np.array(roi)
        aaa.append([roi])
        aaa = np.array(aaa)
        aaa = app.predict(aaa)
        name += dic_[np.argmax(aaa)]
    print(name)
```

注：代码参考https://github.com/ctudoudou/Captcha-Recognition2

看看例子怎么样？

```
# Make socket
transport = TSocket.TSocket('127.0.0.1', 8000)

# Buffering is critical. Raw sockets are very slow
transport = TTransport.TBufferedTransport(transport)

# Wrap in a protocol
protocol = TBinaryProtocol.TBinaryProtocol(transport)

# Create a client to use the protocol encoder
client = Filebuff.Client(protocol)

# Connect!
transport.open()

a = FileData('0.gif', open('0.gif', 'rb').read())
a = client.predict(a)
print(a)
```

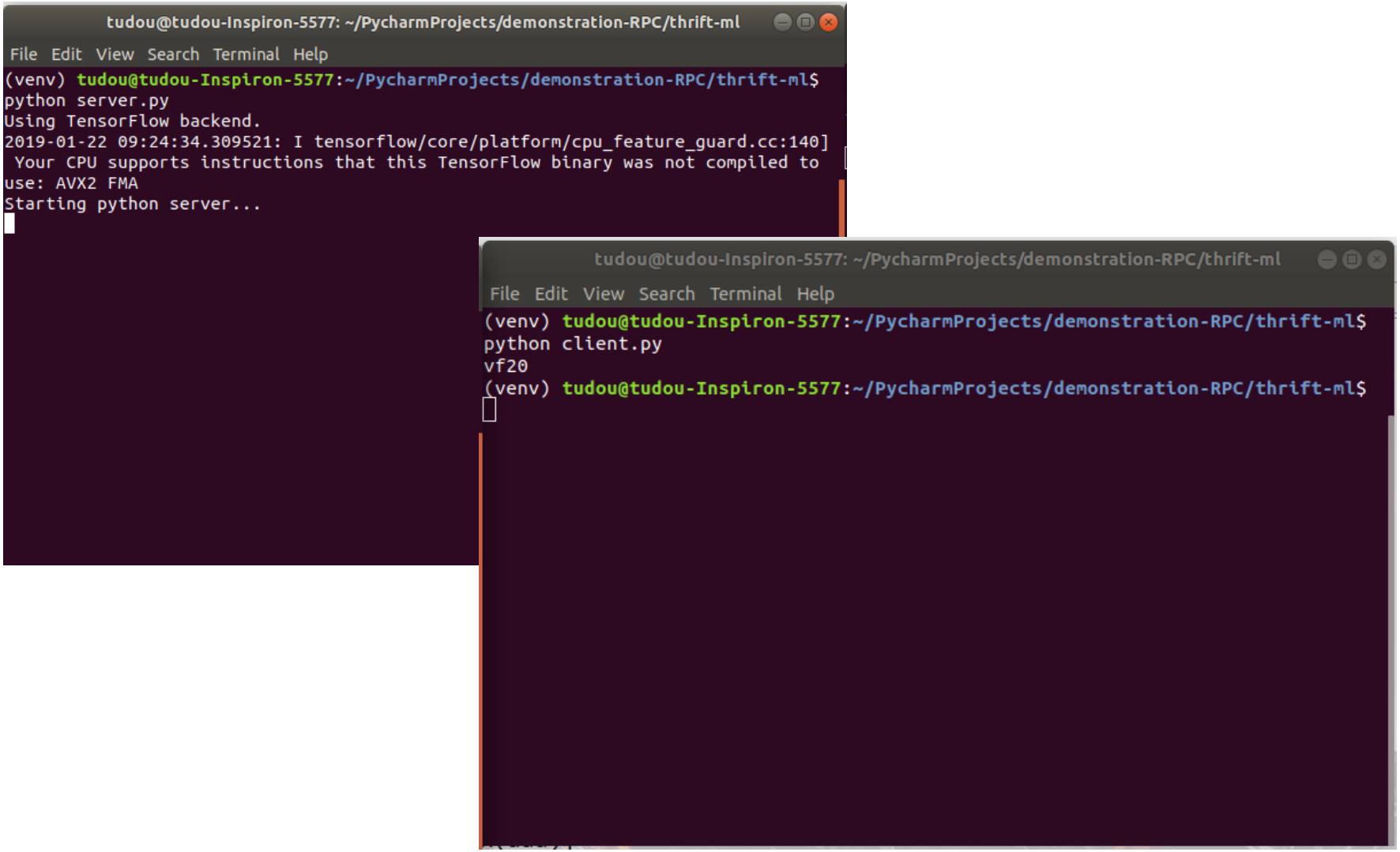
看看例子怎么样？

```python
class FilebuffHandler:
    def __init__(self):
        self.log = {}

    def predict(self, data):
        open('./tmp/{}'.format(data.name), 'wb+').write(data.buff)
        img = Image.open('./tmp/{}'.format(data.name)).convert('L').convert('1')
        name = ''
        for x in range(len(boxs)):
            aaa = []
            roi = img.crop(boxs[x])
            roi = np.array(roi)
            aaa.append([roi])
            aaa = np.array(aaa)
            aaa = app.predict(aaa)
            name += dic_[np.argmax(aaa)]
        return name
```

注：代码参考https://github.com/ctudoudou/demonstration-RPC

看看例子怎么样？


tudou@tudou-Inspiron-5577: ~/PycharmProjects/demonstration-RPC/thrift-ml
File Edit View Search Terminal Help
(venv) tudou@tudou-Inspiron-5577:~/PycharmProjects/demonstration-RPC/thrift-ml$
python server.py
Using TensorFlow backend.
2019-01-22 09:24:34.309521: I tensorflow/core/platform/cpu_feature_guard.cc:140]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
Starting python server...
tudou@tudou-Inspiron-5577: ~/PycharmProjects/demonstration-RPC/thrift-ml
File Edit View Search Terminal Help
(venv) tudou@tudou-Inspiron-5577:~/PycharmProjects/demonstration-RPC/thrift-ml$
python client.py
vf20
(venv) tudou@tudou-Inspiron-5577:~/PycharmProjects/demonstration-RPC/thrift-ml$

总结一下

1. RPC 即远程过程调用
2. RPC 的技术核心有：服务定义，通信，序列化
3. RPC 的框架常见的框架有：gRPC，Dubbo，Thrift
4. 广义 RPC 包含 HTTP 通信
5. RPC 设计规范可以参照 RESTful 设计风格
6. RPC 本身存在安全问题
7. RPC 本身不涉及负载均衡，相比来说 HTTP 的 RPC 会好很多

感谢观看