

# Thrift

serviços para comunicação inter-linguagens

Bruno Atrib Zanchet  
Yahoo! Brasil  
UFPel

# Remote Procedure Call

# Comunicação inter-processos

## Sintaxe “familiar”

“Distributed objects era of the 90s”

(91) Corba

(93) Microsoft COM

(97) Java RMI

(98) XML-RPC (depois SOAP)

(99) EJB

**overhead**



# REST

Roy Fielding, 2000

“estado” abstraído para “recurso”

sintaxe universal para links

operações e content-types bem definidos

stateless, layered, cacheable

# Thrift

Facebook, 2007

# RPC

(de novo)

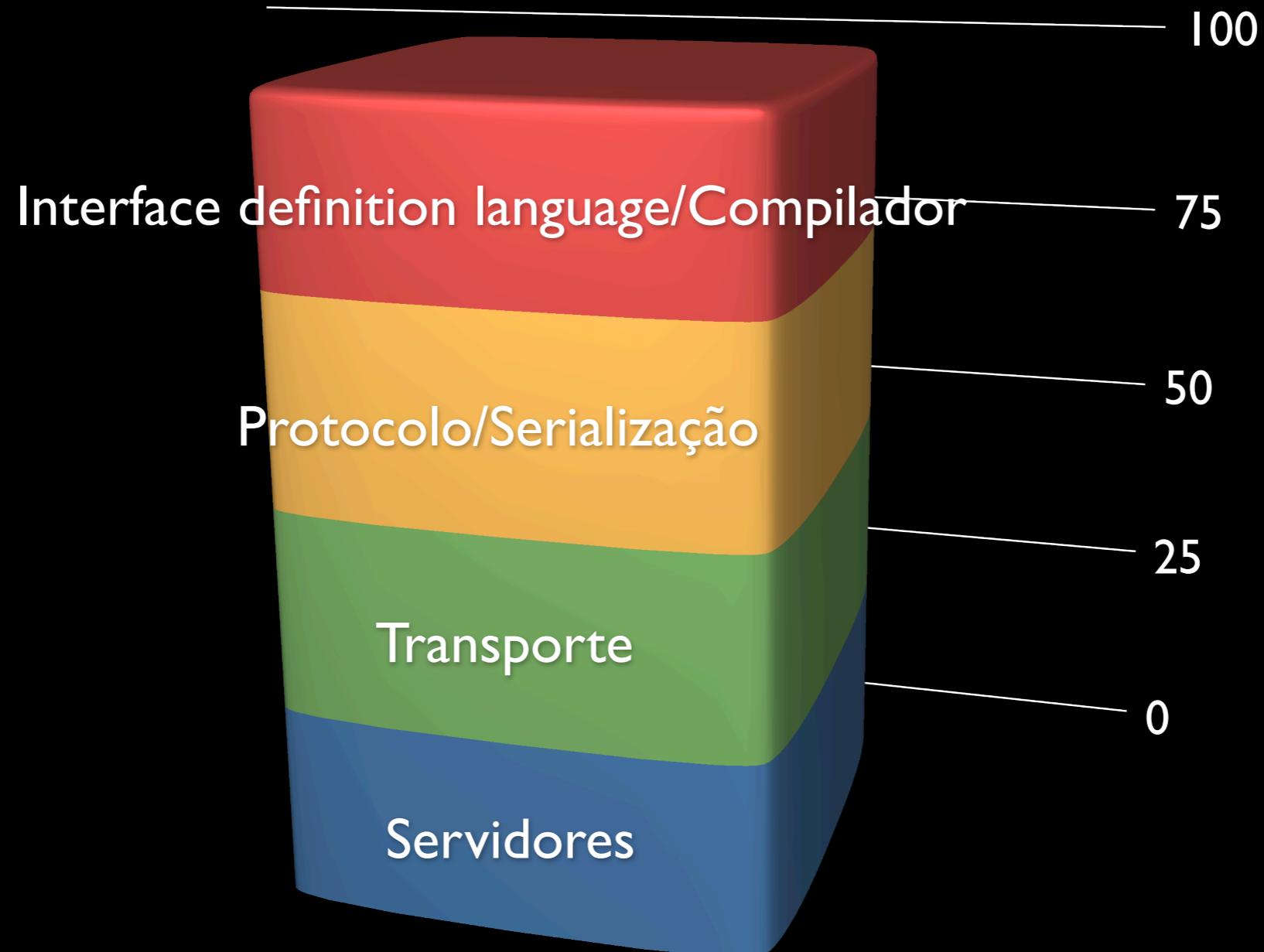
Adotado pelo Apache Incubator (2008)

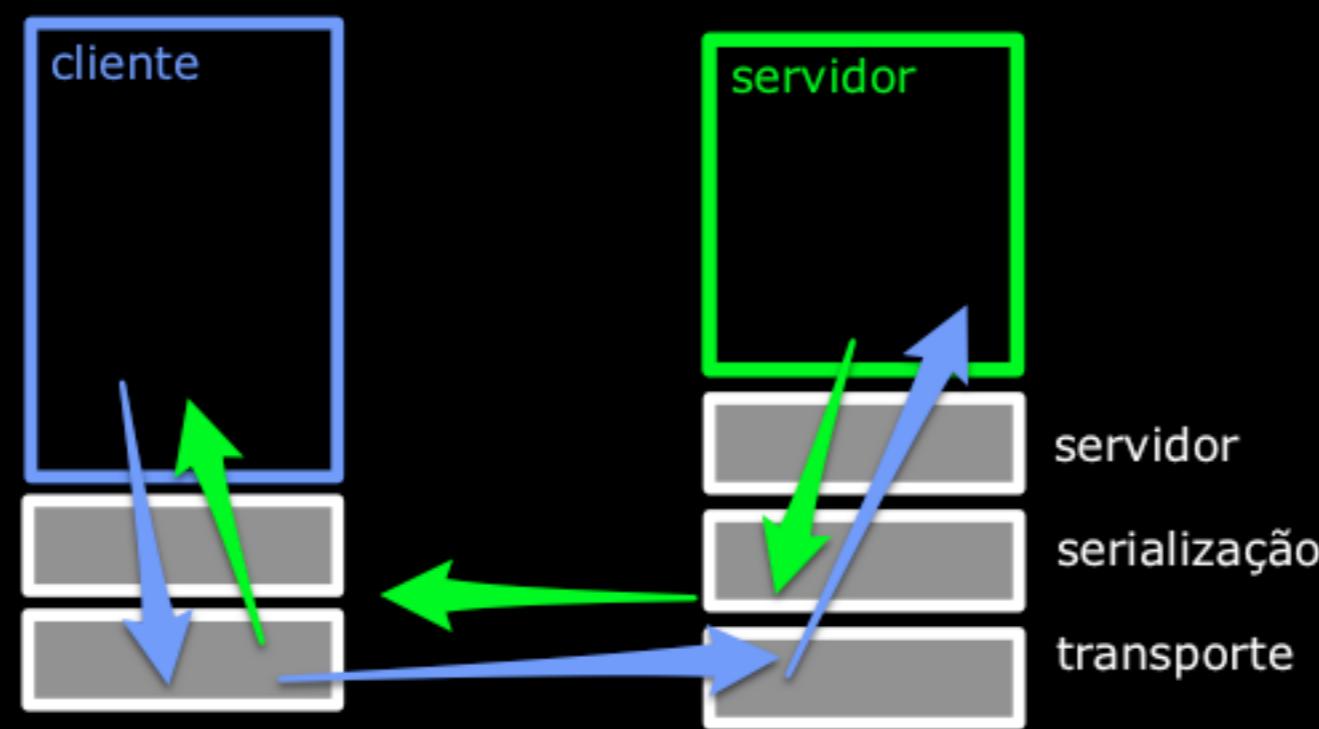
C++, Java, Python, PHP, Ruby, Erlang, Perl,  
Haskell, C#, Cocoa, Smalltalk, OCaml, ...

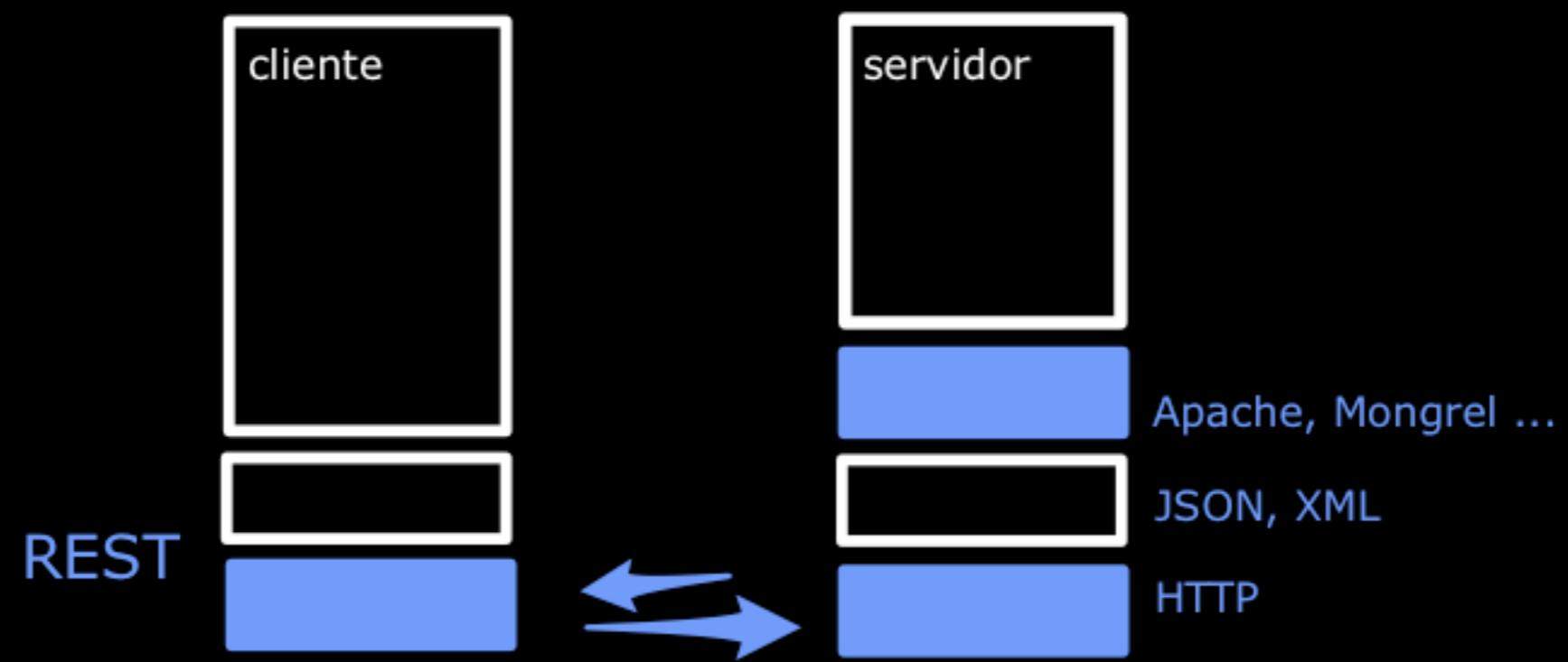
“Rápido, realmente rápido”

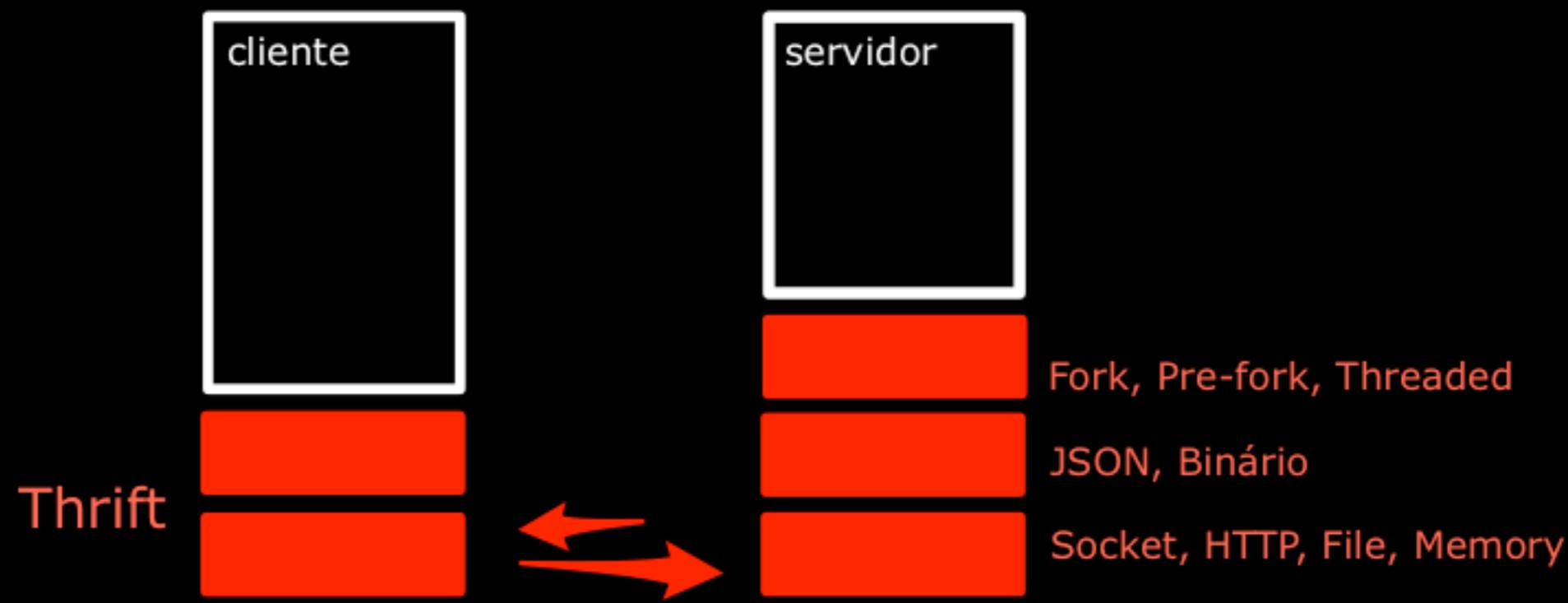
hein?

# Thrift









# Interface Definition Language

compilador/gerador de código

# Tipos

Traduzidos em tipos “nativos”

Sem tipos especiais ou wrappers

# Básicos

bool

byte

i16, i32, i64

double

string

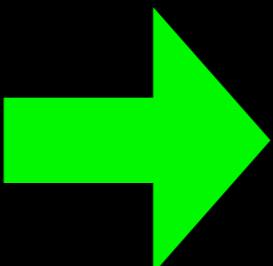
# Containers

`list<type>`

`set<type>`

`map<type1, type2>`

`list<type>`



`[ ]`

Ruby/Python

`array( )`

PHP

`STL vector`

C++

`ArrayList`

Java

# Structs

```
1 struct Example {  
2     1:i32      number=10,  
3     2:i64      big_number,  
4     3:double   decimal,  
5     4:string   name="thrifty"  
6 }  
7
```

```
# examples.thrift
```

## Ruby

```
1 class Example
2   include ::Thrift::Struct
3
4   ::Thrift::Struct.field_accessor self, :number, :big_number, :decimal, :name
5
6   ...
7 end
```

## Python

```
1 class Example(object):
2
3     def __init__(self, number=10, big_number=None, decimal=None, name="thrifty",):
4         self.number = number
5         self.big_number = big_number
6         self.decimal = decimal
7         self.name = name
8
9     ...
```

## PHP

```
1 <?php
2 class Example {
3     public $number = 10;
4     public $big_number = null;
5     public $decimal = null;
6     public $name = "thrifty";
7
8     public function __construct($vals=null) {
9         if (is_array($vals)) {
10             if (isset($vals['number'])) {
11                 $this->number = $vals['number'];
12             }
13             if (isset($vals['big_number'])) {
14                 $this->big_number = $vals['big_number'];
15             }
16             if (isset($vals['decimal'])) {
17                 $this->decimal = $vals['decimal'];
18             }
19             if (isset($vals['name'])) {
20                 $this->name = $vals['name'];
21             }
22         }
23     }
24 }
25 ?>
```

## Java

```
1 public class Example implements TBase, java.io.Serializable, Cloneable {
2     public int number;
3     public long big_number;
4     public double decimal;
5     public String name;
6
7     public Example() {
8         this.number = 10;
9         this.name = "thrifty";
10    }
11
12    public Example(int number, long big_number, double decimal, String name)
13    {
14        this();
15        this.number = number;
16        this.big_number = big_number;
17        this.decimal = decimal;
18        this.name = name;
19    }
20
21    ...
22 }
```

# Exceções

```
8 exception ExampleException {  
9   1:i32      number=10,  
10  2:i64      big_number,  
11  3:double   decimal,  
12  4:string   name="thrifty"  
13 }  
14
```

```
# examples.thrift
```

**Python**

```
class ExampleException(Exception):
```

**Java**

```
public class ExampleException extends Exception {
```

**PHP**

```
class ExampleException extends TException {
```

**Ruby**

```
class ExampleException < ::Thrift::Exception
```

# Serviços

```
15 service RemoteHashMap {
16     void             set(1:i32 key, 2:string value),
17     string          get(1:i32 key) throws (1: KeyNotFound knf),
18     async void      delete(1:i32 key)
19 }
20

# examples.thrift
```

# Protocolo

Métodos para leitura e escrita

Encoding dos tipos básicos, structs e  
containers

## TProtocol

`readMessageBegin()`

`writeMessageBegin(name, type, seq)`

`readStructBegin()`

`writeStructBegin(name)`

`readI16()`

`writel16()`

`readI32()`

`...`

**TBinaryProtocol**

**TCompactProtocol**

**TJSONProtocol**

...

# Transporte

Transferência de dados

Duas interfaces

TTransport

open

close

isOpen

read

write

flush

TServerTransport

open

listen

accept

close

**TSocket**

**TFileTransport**

**TMemoryBuffer**

**THttpClient**

...

# Servidores

**TThreadedServer**

**TThreadPoolServer**

**TForkingServer**

...

Fim.

WS-\*, alguém?

SOAP (formerly known as Simple Object Access Protocol)	P3P
SOAP Message Transmission Optimization Mechanism	WS-ReliableMessaging
WS-Notification	WS-Reliability
WS-BaseNotification	WS-RM Policy Assertion
WS-Topics	Web Services Resource Framework
WS-BrokeredNotification	WS-BaseFaults
WS-SapOverUDP	WS-ServiceGroup
WS-Addressing	WS-ResourceProperties
WS-Transfer	WS-ResourceLifetime
WS-Eventing	WS-Transfer
WS-Enumeration	Resource Representation SOAP Header Block
WS-MakeConnection	WS-I Basic Profile
WS-Policy	WS-I Basic Security Profile
WS-PolicyAssertions	Simple Soap Binding Profile
WS-PolicyAttachment	WS-BPEL
WS-Discovery	WS-CDL
WS-Inspection	Web Services Choreography Interface
WS-MetadataExchange	WS-Choreography
Universal Description, Discovery, and Integration (UDDI)	XML Process Definition Language
WSDL 2.0 Core	WS-BusinessActivity
WSDL 2.0 SOAP Binding	WS-AtomicTransaction
Web Services Semantics (WSDL-S)	WS-Coordination
WS-Resource Framework (WSRF)	WS-CAF
WS-Security	WS-Transaction
XML Signature	WS-Context
XML Encryption	WS-CF
XML Key Management (XKMS)	WS-TXM
WS-SecureConversation	WS-Management
WS-SecurityPolicy	WS-Management Catalog
WS-Trust	WS-ResourceTransfer
WS-Federation	WSDM
WS-Federation Active Requestor Profile	Web Services for Remote Portlets
WS-Federation Passive Requestor Profile	WS-Provisioning
Web Services Security Kerberos Binding	Devices Profile for Web Services (DPWS)
Web Single Sign-On Interoperability Profile	ebXML
Web Single Sign-On Metadata Exchange Protocol	ISO/IEC 19784-2:2007 Information technology
Security Assertion Markup Language (SAML)	ISO 19133:2005 Geographic information
XACML	ISO/IEC 20000-1:2005 Information technology
ISO/IEC 20000-2:2005 Information technology	ISO/IEC 25437:2006 Information technology

# Na prática

# I. Definir as estruturas

```
3
4 enum MartialArt {
5   AIKIDO    = 1,
6   KARATE    = 2
7 }
8
9 struct UserProfile {
10  1: i32          uid,
11  2: string       name,
12  3: MartialArt style
13 }
14
15 service UserStorage {
16   void          store(1: UserProfile user),
17   UserProfile retrieve(1: i32 uid)
18 }
19
```

# service.thrift

# I. Definir as estruturas

```
3
4 enum MartialArt {
5   AIKIDO    = 1,
6   KARATE    = 2
7 }
8
9 struct UserProfile {
10  1: i32          uid,
11  2: string       name,
12  3: MartialArt style
13 }
14
15 service UserStorage {
16   void          store(1: UserProfile user),
17   UserProfile retrieve(1: i32 uid)
18 }
19
```

Enum!

## 2. Gerar código “stub”

```
$ thrift --gen rb --gen java service.thrift
```

## 2. Gerar código “stub”

```
$ thrift --gen php --gen py:new_style service.thrift
```

# 3. Implementar lógica do serviço

```
16
17 class UserStorageHandler:
18     def __init__(self):
19         pass
20
21     def store(self, user):
22         print "stored " + str(user)
23
24     def retrieve(self, id):
25         print "retrieved " + str(id)
26         return UserProfile(
27             uid=id,
28             name="Ralph Waldo Emerson",
29             style=MartialArt.KARATE
30         )
31
# server.py
```

# 3. Implementar lógica do serviço

```
16  
17 class UserStorageHandler:  
18     def __init__(self):  
19         pass  
20  
21     def store(self, user):  
22         print "stored " + str(user)  
23  
24     def retrieve(self, id):  
25         print "retrieved " + str(id)  
26         return UserProfile(  
27             uid=id,  
28             name="Ralph Waldo Emerson",  
29             style=MartialArt.KARATE  
30         )  
31  
  
# server.py
```

código gerado!

# 4. Escolher o servidor

```
32  
33 handler = UserStorageHandler()  
34 processor = example.UserStorage.Processor(handler)  
35 transport = TSocket.TServerSocket(9090)  
36 tfactory = TTransport.TBufferedTransportFactory()  
37 pfactory = TBinaryProtocol.TBinaryProtocolFactory()  
38  
39 server = TServer.TThreadedServer(processor, transport, tfact...  
40  
41 print 'Starting the server...'  
42 server.serve()  
43  
  
# server.py
```

# 5. Implementar o cliente

```
17 $socket = new TSocket('localhost', 9090);
18 $transport = new TBufferedTransport($socket, 1024, 1024);
19 $protocol = new TBinaryProtocol($transport);
20 $client = new UserStorageClient($protocol);
21
22 $transport->open();
23
24 $user = $client->retrieve(1);
25 var_dump($user);
26
27 $new_user = new example_UserProfile(array(
28     "uid" => '123',
29     "name" => "Ralph Waldo Emerson",
30     "style" => example_MartialArt::KARATE
31 ));
32 $client->store($new_user);
33
34 $transport->close();

# client.php
```

# 5. Implementar o cliente

```
17 $socket = new TSocket('localhost', 9090);
18 $transport = new TBufferedTransport($socket, 1024, 1024);
19 $protocol = new TBinaryProtocol($transport);
20 $client = new UserStorageClient($protocol);
21
22 $transport->open();
23
24 $user = $client->retrieve(1);
25 var_dump($user);
26
27 $new_user = new example_UserProfile(array(
28     "uid" => '123',
29     "name" => "Ralph Waldo Emerson",
30     "style" => example_MartialArt::KARATE
31 ));
32 $client->store($new_user);
33
34 $transport->close();
```

código gerado!

# 5. Implementar o cliente

```
17 $socket = new TSocket('localhost', 9090);
18 $transport = new TBufferedTransport($socket, 1024, 1024);
19 $protocol = new TBinaryProtocol($transport);
20 $client = new UserStorageClient($protocol);
21
22 $transport->open();
23
24 $user = $client->retrieve(1);
25 var_dump($user);
26
27 $new_user = new example_UserProfile(array(
28     "uid" => '123',
29     "name" => "Ralph Waldo Emerson",
30     "style" => example_MartialArt::KARATE
31 ));
32 $client->store($new_user);
33
34 $transport->close();
```

# client.php

código gerado!

# 6. Deploy!

```
$ python server.py
```

?!

## invocação remota

```
# client.php
```

```
22 $transport->open();
23
24 $user = $client->retrieve(1);
25 var_dump($user);
26
27 $new_user = new example_UserProfile(array(
28     "uid" => '123',
29     "name" => "Ralph Waldo Emerson",
30     "style" => example_MartialArt::KARATE
31 ));
32 $client->store($new_user);
33
34 $transport->close();
```

```
# server.py
```

```
17 class UserStorageHandler:
18     def __init__(self):
19         pass
20
21     def store(self, user):
22         print "stored " + str(user)
23
24     def retrieve(self, id):
25         print "retrieved " + str(id)
26         return UserProfile(
27             uid=id,
28             name="Ralph Waldo Emerson",
29             style=MartialArt.KARATE
30         )
31
```

mágica!

```
# client.php                                     # server.py

22 $transport->open();                           17 class UserStorageHandler:
23 $user = $client->retrieve(1);                   18     def __init__(self):
24 var_dump($user);                               19         pass
25
26 $new_user = new example_UserProfile(array(      20
27     "uid" => '123',
28     "name" => "Ralph Waldo Emerson",
29     "style" => example_MartialArt::KARATE
30 ));                                           21     def store(self, user):
31 $client->store($new_user);                     22         print "stored " + str(user)
32
33 $transport->close();                           23
34
```

```
# client.php                                         # server.py

22 $transport->open();                            17 class UserStorageHandler:
23 $user = $client->retrieve(1);                   18     def __init__(self):
24 var_dump($user);                                19         pass
25
26 $new_user = new example_UserProfile(array(      20
27     "uid" => '123',
28     "name" => "Ralph Waldo Emerson",
29     "style" => example_MartialArt::KARATE
30 ));                                              21     def store(self, user):
31 $client->store($new_user);                     22         print "stored " + str(user)
32
33 $transport->close();                           23
34
```

... e caímos aqui

```
# client.php
```

```
22 $transport->open();  
23  
24 $user = $client->retrieve(1);  
25 var_dump($user);  
26  
27 $new_user = new example_UserProfile(array(  
28     "uid" => '123',  
29     "name" => "Ralph Waldo Emerson",  
30     "style" => example_MartialArt::KARATE  
31 ));  
32 $client->store($new_user);  
33  
34 $transport->close();
```

```
# server.py
```

```
17 class UserStorageHandler:  
18     def __init__(self):  
19         pass  
20  
21     def store(self, user):  
22         print "stored " + str(user)  
23  
24     def retrieve(self, id):  
25         print "retrieved " + str(id)  
26         return UserProfile(  
27             uid=id,  
28             name="Ralph Waldo Emerson",  
29             style=MartialArt.KARATE  
30         )  
31
```

return

mágica!

```
# client.php                                     # server.py

22 $transport->open();                           17 class UserStorageHandler:
23 $user = $client->retrieve(1);                   18     def __init__(self):
24 var_dump($user);                               19         pass
25
26 $new_user = new example_UserProfile(array(      20
27     "uid" => '123',
28     "name" => "Ralph Waldo Emerson",
29     "style" => example_MartialArt::KARATE
30 ));                                           21     def store(self, user):
31 $client->store($new_user);                     22         print "stored " + str(user)
32
33 $transport->close();                           23
34
```

de volta!

```
# client.php                                         # server.py

22 $transport->open();                           17 class UserStorageHandler:
23 $user = $client->retrieve(1);                   18     def __init__(self):
24 var_dump($user);                               19         pass
25
26 $new_user = new example_UserProfile(array(      20
27     "uid" => '123',
28     "name" => "Ralph Waldo Emerson",
29     "style" => example_MartialArt::KARATE
30 ));                                           21     def store(self, user):
31 $client->store($new_user);                     22         print "stored " + str(user)
32
33 $transport->close();                           23
34
```

```
# client.php                                         # server.py

22 $transport->open();                            17 class UserStorageHandler:
23 $user = $client->retrieve(1);                   18     def __init__(self):
24 var_dump($user);                                19         pass
25
26 $new_user = new example_UserProfile(array(      20
27     "uid" => '123',
28     "name" => "Ralph Waldo Emerson",
29     "style" => example_MartialArt::KARATE
30 ));                                              21     def store(self, user):
31 $client->store($new_user);                     22         print "stored " + str(user)
32
33 $transport->close();                           23
34
```

invocação remota



```
# client.php
```

```
22 $transport->open();
23
24 $user = $client->retrieve(1);
25 var_dump($user);
26
27 $new_user = new example_UserProfile(array(
28     "uid" => '123',
29     "name" => "Ralph Waldo Emerson",
30     "style" => example_MartialArt::KARATE
31));
32 $client->store($new_user);
33
34 $transport->close();
```

```
# server.py
```

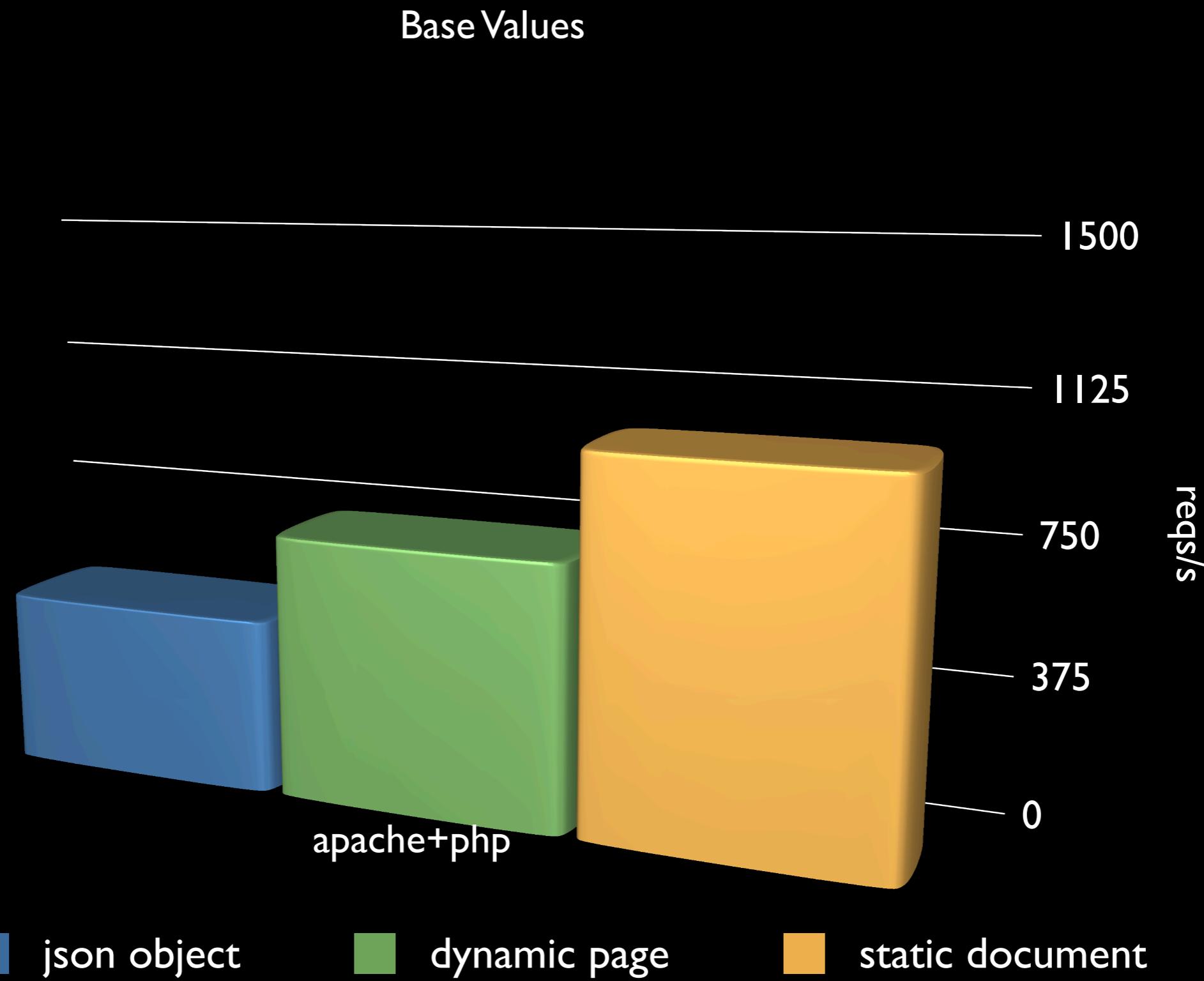
```
17 class UserStorageHandler:
18     def __init__(self):
19         pass
20
21     def store(self, user):
22         print "stored " + str(user)
23
24     def retrieve(self, id):
25         print "retrieved " + str(id)
26         return UserProfile(
27             uid=id,
28             name="Ralph Waldo Emerson",
29             style=MartialArt.KARATE
30         )
31
```

bingo!

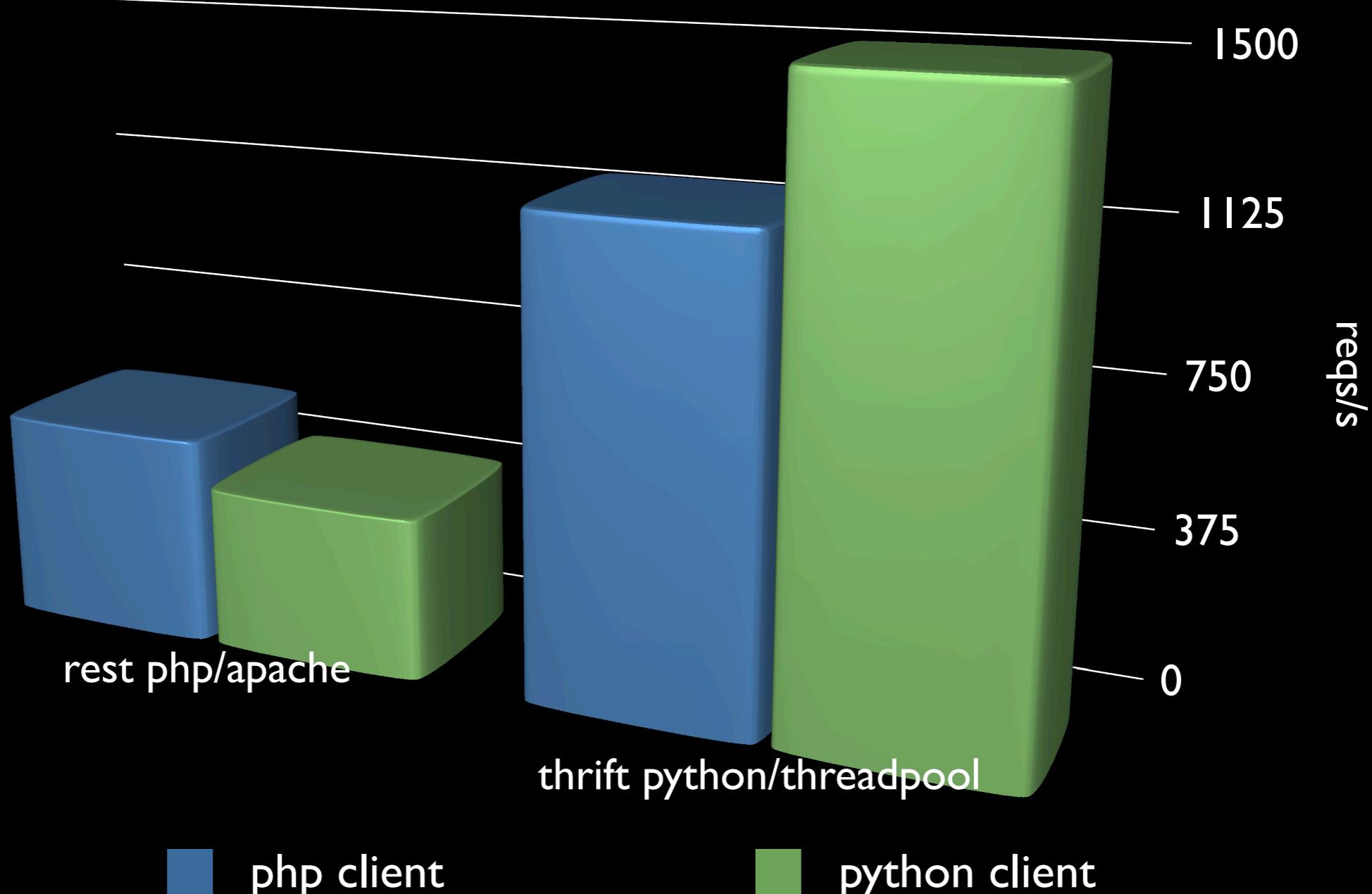
```
# client.php  
  
22 $transport->open();  
23  
24 $user = $client->retrieve(1);  
25 var_dump($user);  
26  
27 $new_user = new example_UserProfile(array(  
28     "uid" => '123',  
29     "name" => "Ralph Waldo Emerson",  
30     "style" => example_MartialArt::KARATE  
31 ));  
32 $client->store($new_user);  
33  
34 $transport->close();  
  
# server.py  
  
17 class UserStorageHandler:  
18     def __init__(self):  
19         pass  
20  
21     def store(self, user):  
22         print "stored " + str(user)  
23  
24     def retrieve(self, id):  
25         print "retrieved " + str(id)  
26         return UserProfile(  
27             uid=id,  
28             name="Ralph Waldo Emerson",  
29             style=MartialArt.KARATE  
30         )  
31
```



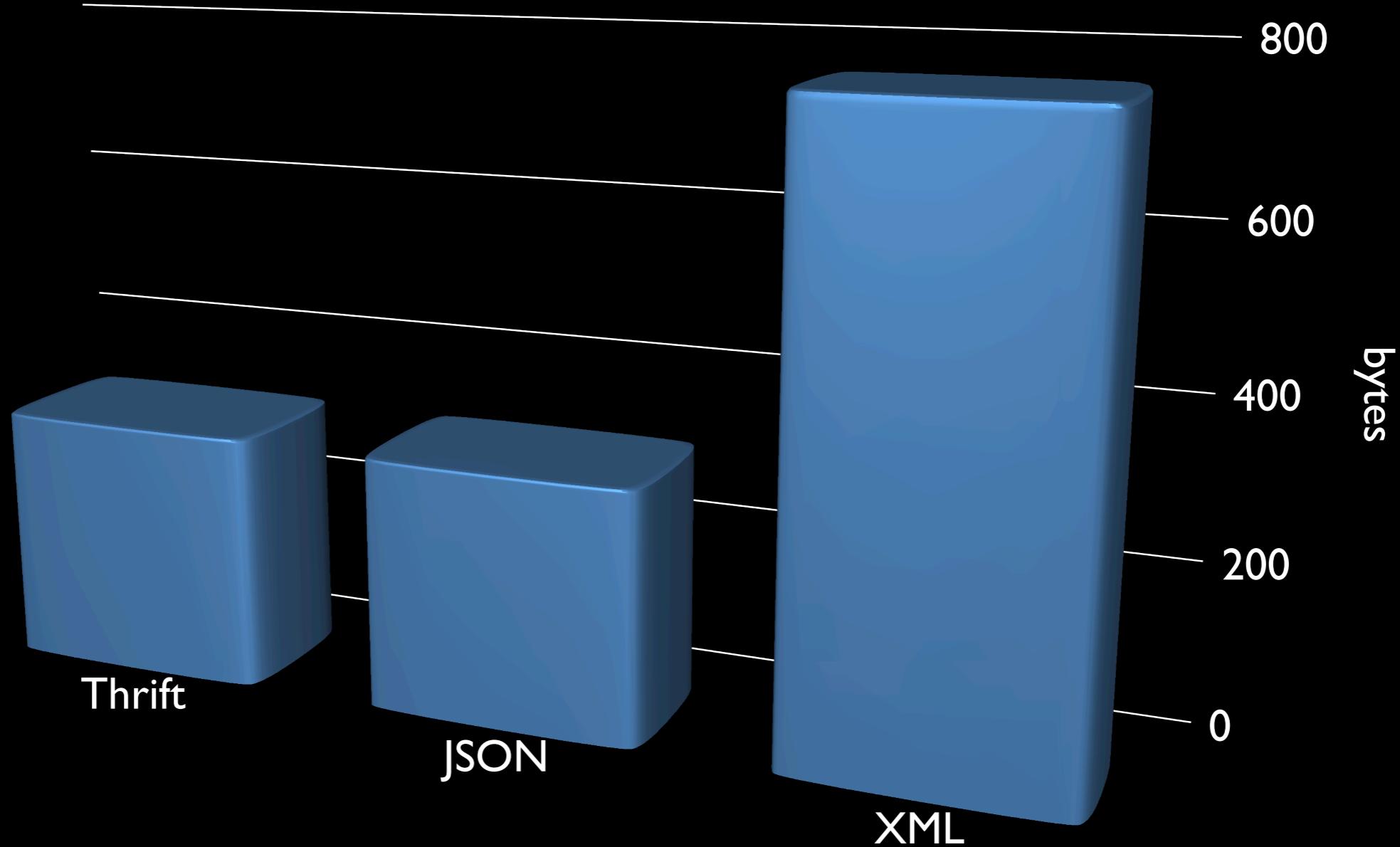
# Benchmarks



## Thrift versus Rest - speed



## Serialization Size



# Desvantagens

Não tão ubíquo (quanto HTTP)

Não tão maduro (quanto HTTP)

Não tão óbvio (quanto HTTP) ?!

# Pontos fortes

Compatibilidade entre linguagens

Serialização built-in

Performance!

meme  
DO YAHOO!



<http://meme.yahoo.com>

Mais

Versionamento da interface

Thrift + Protocol buffers

# Referências

<http://incubator.apache.org/thrift/>

<https://github.com/bzanchet/presentation-thrift-fisl10/>