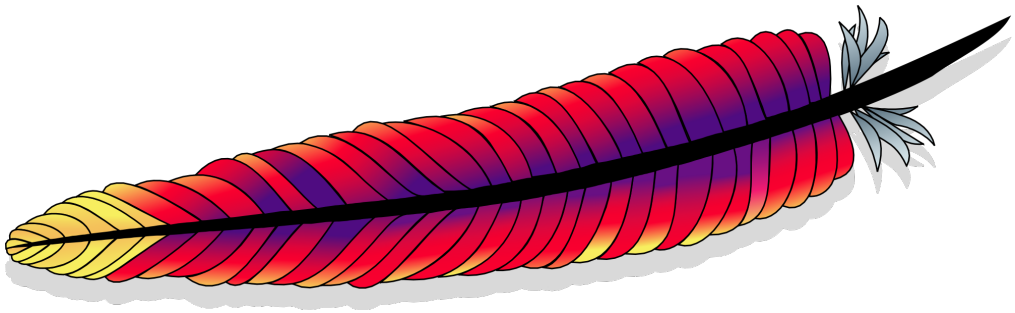
Xposé 2014-2015

# Apache Thrift

## Framework RPC multilangage



Eric LEFORT
IR3

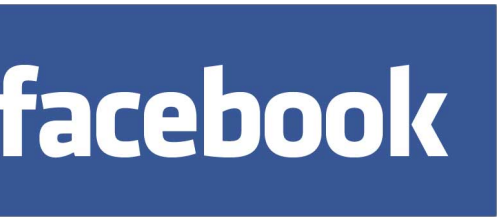
# Sommaire

# Introduction

# Historique

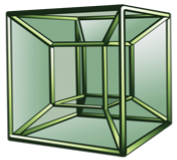
- Framework RPC (cross-language)
- Indépendant du langage
- Inclus : protocoles de transport, de sérialisation / déserialisation et création de serveurs
- Projet initié par Facebook (2006)
- Projet Apache depuis 2007
- Licence Apache (2.0)
- Version stable 0.9.2 (07-11-2014)

# Qui l'utilise ?









HYPERTABLE INC

# Contexte

- 80% des langages les plus représentés

2003

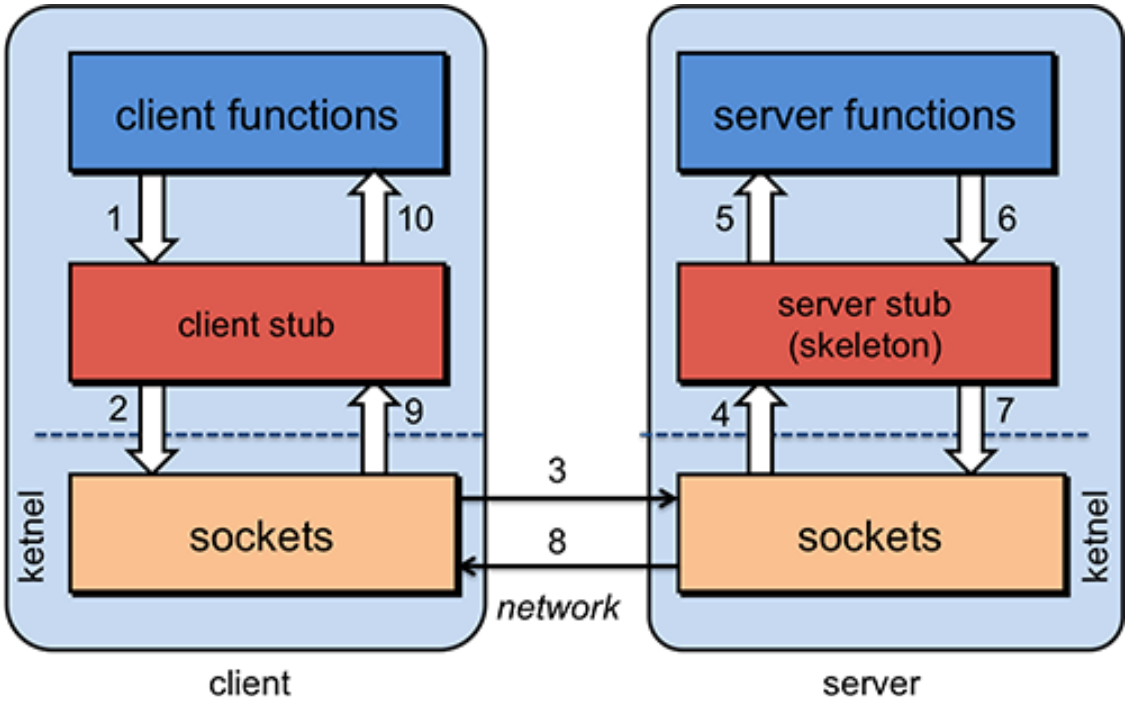
| Langage | Rang | Cumulé |
|---|---|---|
| Java | 23.08 % | 23.08 % |
| C | 18.47 % | 41.55 % |
| C++ | 15.56 % | 57.12 % |
| Perl | 9.42 % | 66.54 % |
| (Visual) Basic | 7.81 % | 74.35 % |
| PHP | 4.76 % | 79.11 % |

2013

| Langage | Rang | Cumulé |
|---|---|---|
| C | 17.81 % | 17.81 % |
| Java | 16.66 % | 34.47 % |
| Objective-C | 10.36 % | 44.82 % |
| C++ | 8.82 % | 53.64 % |
| PHP | 5.99 % | 59.63 % |
| C# | 5.78 % | 65.41 % |
| (Visual) Basic | 4.35 % | 69.76 % |
| Python | 4.18 % | 73.94 % |
| Perl | 2.27 % | 76.21 % |
| JavaScript | 1.65 % | 77.87 % |
| Ruby | 1.48 % | 79.35 % |

# Apache Thrift

# Remote Procedure Call (RPC)

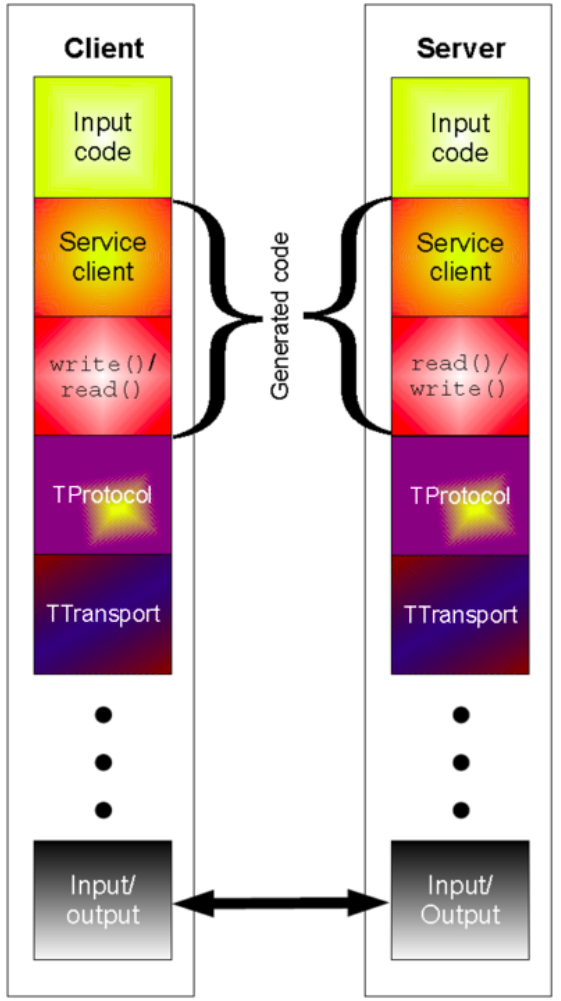


# Richesse

- Langages
  - C / C++ / C#
  - Java
  - JavaScript
  - Objective-C
  - Python
  - PHP …
  - … Perl / Ruby / Smalltalk / OCami  / Go / Haskell
- Plate-formes : Windows, Linux, OS X, IOS et Android
- Architecture : x86 et x64

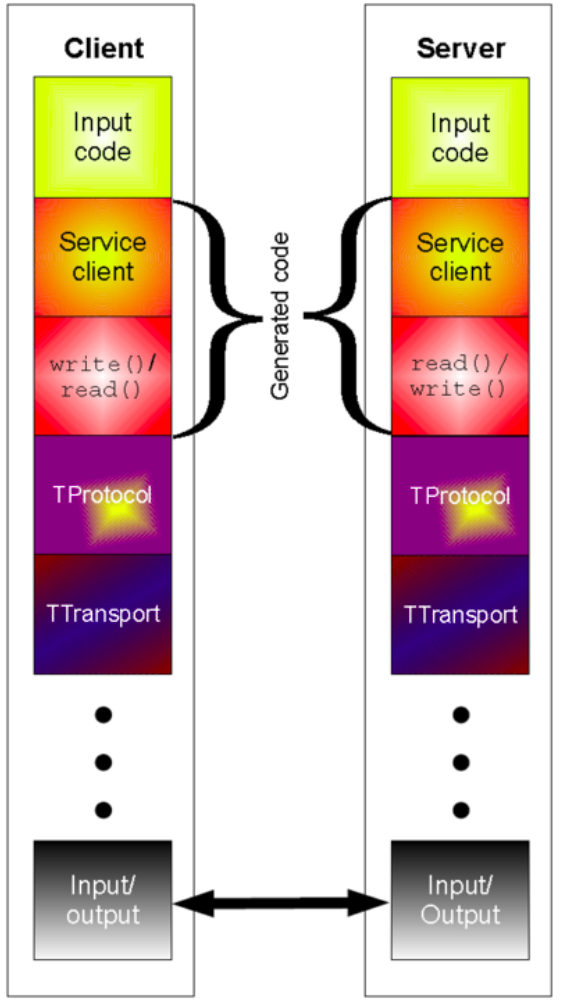
# Pile de protocoles (1/2)

- Protocoles (sérialisation)
  - TBinaryProtocol
  - TCompactProtocol
  - TDenseProtocol
  - TJSONProtocol
  - TSimpleJSONProtocol
  - TDebugProtocol

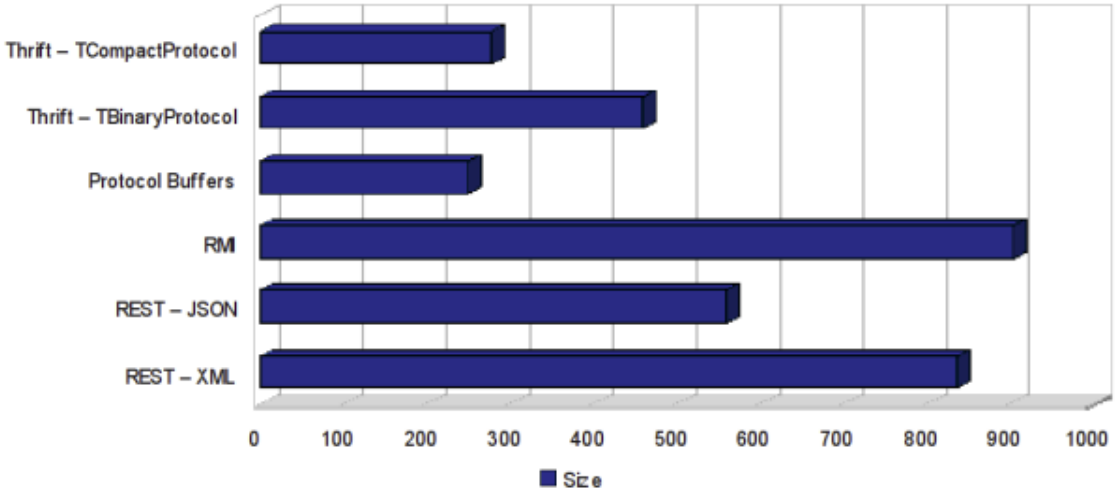


# Pile de protocoles (2/2)

- Transport (interface I/O)
  - TSocket
  - TFramedTransport
  - TFileTransport
  - TMemoryTransport
  - TZlibTransport
- Serveurs
  - TSimpleServer
  - TThreadPoolServer
  - TNonblockingServer

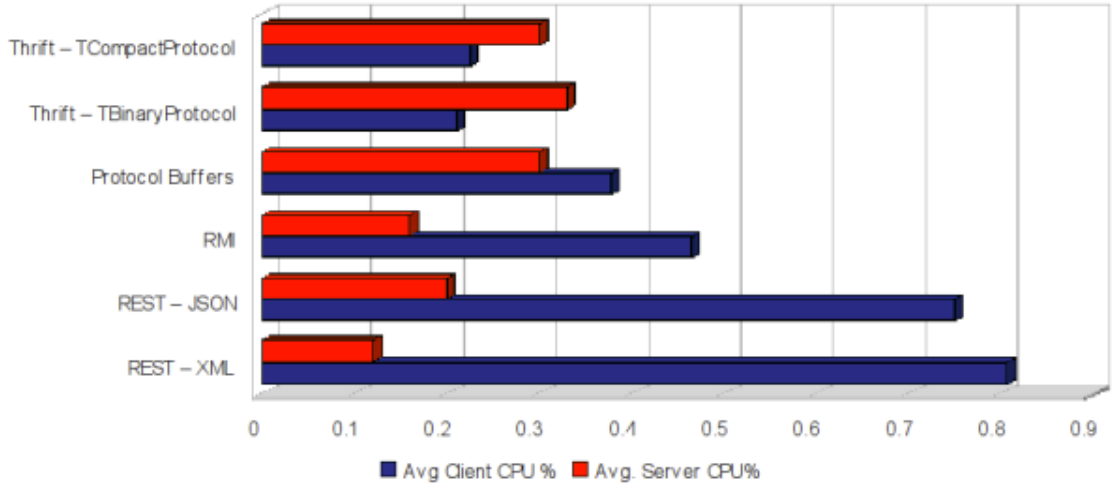


# Performance




Taille des données en octets

Utilisation moyenne CPU client / serveur



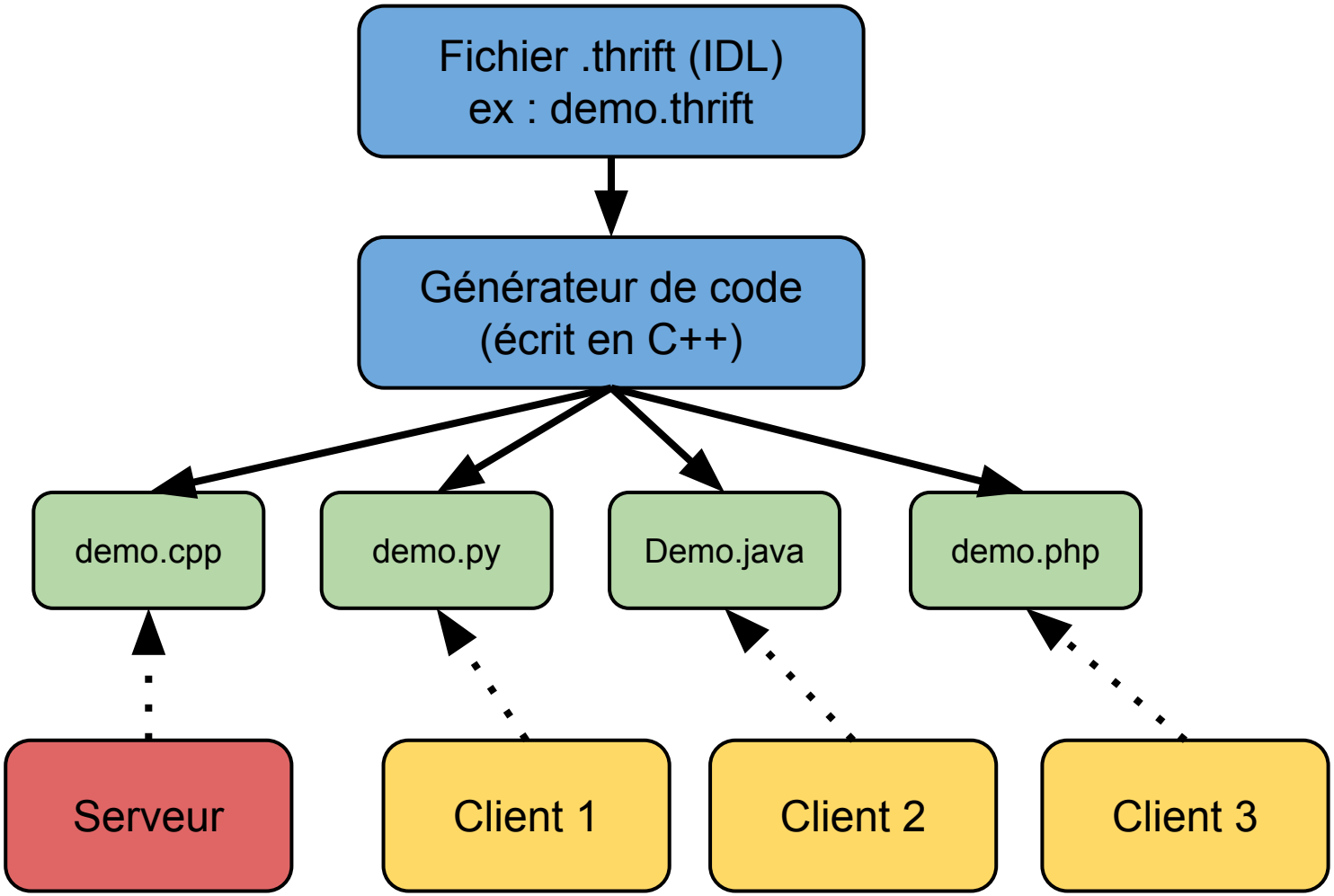
# Applications

- **Streaming -** Communications caractérisées par un flux continu d'octets à partir d'un serveur vers un ou plusieurs clients.

- **Messaging -** Transmission de messages de façon asynchrone, souvent en file d'attente, les communications sont faiblement liées.

- **Remote Procedure Calls (RPC) -** Remote Procedure Call permet l’appel de fonctions entre processus sur différentes machines.

# Principe


Fichier .thrift (IDL)
ex : demo.thrift
Générateur de code
(écrit en C++)
demo.cpp
demo.py
Demo.java
demo.php
Serveur
Client 1
Client 2
Client 3

# Interface Definition Language (IDL)

- Proche de l'annotation C
  - types (de base, structures, containers, exceptions, services)
  - namespaces
  - enum
  - constante
  - includes
  - typedef
  - commentaires (“#”, “/* */” ou “//”)
- Extension « **.thrift** »
- Génération de code

  thrift --gen <language> <Thrift filename>

  « gen-<langage>/ »

# Typage de Thrift (1/3)

- Types de bases

| | |
|---|---|
| **bool** | valeur booléenne (true or false) |
| **byte** | 8-bit signed integer |
| **i16** | 16-bit signed integer |
| **i32** | 32-bit signed integer |
| **i64** | 64-bit signed integer |
| **double** | 64-bit floating point number |
| **string** | chaîne de caratères encodé en UTF-8 |

- Type spécial

| | |
|---|---|
| **binary** | séquence d'octets non-encodé |

# Typage de Thrift (2/3)

- Structures

  Équivalent des classes en POO (sans l'héritage)

```
struct <name> {
    <id>: <required | optional> <type> <field>;
}
```

```
struct Example {
  1:i32 number=10,
  2:i64 bigNumber,
  3:double decimals,
  4:string name="thrifty"
}
```

- Containers

| | |
|---|---|
| **list<type>** | liste ordonnée d'éléments |
| **set<type>** | set non-ordonné d'éléments uniques |
| **map<type 1,type 2>** | map de valeurs à clé unique |

# Typage de Thrift (3/3)

- Exceptions

```
exception <name> {
    <id>: <type> <field>;
}
```

```
exception MyException {
    1: string message;
}
```

- Services

  Équivalent des interfaces en POO

  Héritage possible : extends <service>

```
service <name> {
  <returntype> <name>(<arguments>)
    [throws (<exceptions>)]
  ...
}
```

```
service StringCache {
  void set(1:i32 key, 2:string value),
  string get(1:i32 key) throws (1:KeyNotFound knf),
  void delete(1:i32 key)
}
```

# Opérations avancées (1/2)

- Enum

```
enum fb_status {
  DEAD = 0,
  STARTING = 1,
  ALIVE = 2,
  STOPPING = 3,
  STOPPED = 4,
  WARNING = 5,
}
```

- Includes

```
include "tweet.thrift"
...
struct TweetSearchResult {
    1: list<tweet.Tweet> tweets;
}
```

# Opérations avancées (2/2)

- Namespaces

```
namespace cpp com.example.project
namespace java com.example.project
```

C++ : **namespace com { namespace example { namespace project {**

Java : **package com.example.project**

- Constantes

```
const i32 INT_CONST = 1234;
const map<string,string> MAP_CONST = {"hello": "world", "goodnight": "moon"}
```

- Typesdef

```
typedef i64 UserId
```

# Versioning

- Ajout d'un nouveau champ
  - Nouveau Client, ancien Serveur

    le serveur ignore le nouveau champ, il ne le connaît pas.
  - Ancien Client, nouveau Serveur

    le serveur ne trouve pas le champ attendu, laisse le champ inchangé.

- Suppression d'un champ
  - Nouveau Client, ancien Serveur

    le serveur ne trouve pas le champ attendu, laisse le champ inchangé.
  - Ancien Client, nouveau Serveur

    le client envoi un champ déprécié, le serveur l'ignore.

# Exemple

# Création de l'IDL

- Création du service
  - Fichier « **maths.thrift** »

```
namespace java fr.upem.maths

service Operation {
    i32 multiply(1:i32 val1, 2:i32 val2)
}
```

- Génération de l'IDL en Python et Java
  - « thrift --gen java --gen py maths.thrift »
  - « gen-py/ » et « gen-java/ »

# Serveur Python

- Fichier « server.py »

```
import sys
sys.path.append("gen-py")
from maths import Operation

from thrift.transport import TSocket
from thrift.server import TServer

class OperationHandler:
    def multiply(self, val1, val2):
        print "[Server] request :", val1, "*" , val2
        return val1 * val2

handler = OperationHandler()
processor = Operation.Processor(handler)

listening_socket = TSocket.TServerSocket(port=8585)
server = TServer.TSimpleServer(processor,listening_socket)
server.serve()
```

# Client Java (1/2)

- Fichier « Client.java »

```java
import org.apache.thrift.protocol.TBinaryProtocol;
import org.apache.thrift.transport.TSocket;
import org.apache.thrift.TException;

import fr.upem.maths.Operation;

public class Client{

    public static void main(String[] args) throws TException {
        TSocket socket = new TSocket("localhost",8585);
        socket.open();

        TBinaryProtocol protocol = new TBinaryProtocol(socket);
        Operation.Client client = new Operation.Client(protocol);

        int res = client.multiply(Integer.parseInt(args[0]), Integer.parseInt(args[1]));

        System.out.println("[Client] the answer is: " + res);

        socket.close();
    }
}
```

# Client Python (2/2)

- Fichier « client.py »

```python
import sys
sys.path.append("gen-py")

from thrift.transport import TSocket
from thrift.protocol import TBinaryProtocol
from maths import Operation

socket = TSocket.TSocket("localhost",8585)
socket.open()
protocol = TBinaryProtocol.TBinaryProtocol(socket)

client = Operation.Client(protocol)
msg = client.multiply(int(sys.argv[1]), int(sys.argv[2]))
print("[Client] the answer is : %s" % msg)
```

# Alternatives

# Protocol Buffers

- Similaire à Apache Thrift
- Développer par Google (2001)
- Licence BSD depuis 2008
- Version stable 2.6.1 (20-10-2014)
- C++, Java, Python et JavaScript
- Google, Netty

# Apache Avro

- Similaire à Apache Thrift
- Apache Software Foundation
- Licence Apache 2.0
- Version stable 1.7.7 (23-07-2014)
- Développé pour Apache's Hadoop
- Schémas définis en JSON
- Pas de génération de code

# Conclusion

- Remote Procedure Call
- Richesse (langages, plate-formes)
- Implémentation de protocoles de transport, sérialisation / déserialisation et création de serveurs
- Branche Thrift de Facebook https://github.com/facebook/fbthrift
  - Amélioration du générateur de codes
  - Optimisation des protocoles de transport
  - Intégration futur au projet Apache Thrift

# Bibliographie


- Apache Thrift
  - http://thrift.apache.org/
  - http://wiki.apache.org/thrift/
  - https://thrift.apache.org/docs/BuildingFromSource
  - http://thrift.apache.org/static/files/thrift-20070401.pdf
  - http://diwakergupta.github.io/thrift-missing-guide/
- Google Protocol Buffer
  - https://code.google.com/p/protobuf/
- Apache Avro
  - http://avro.apache.org/

# Questions / Réponses