

共感#5

# Apache Thrift

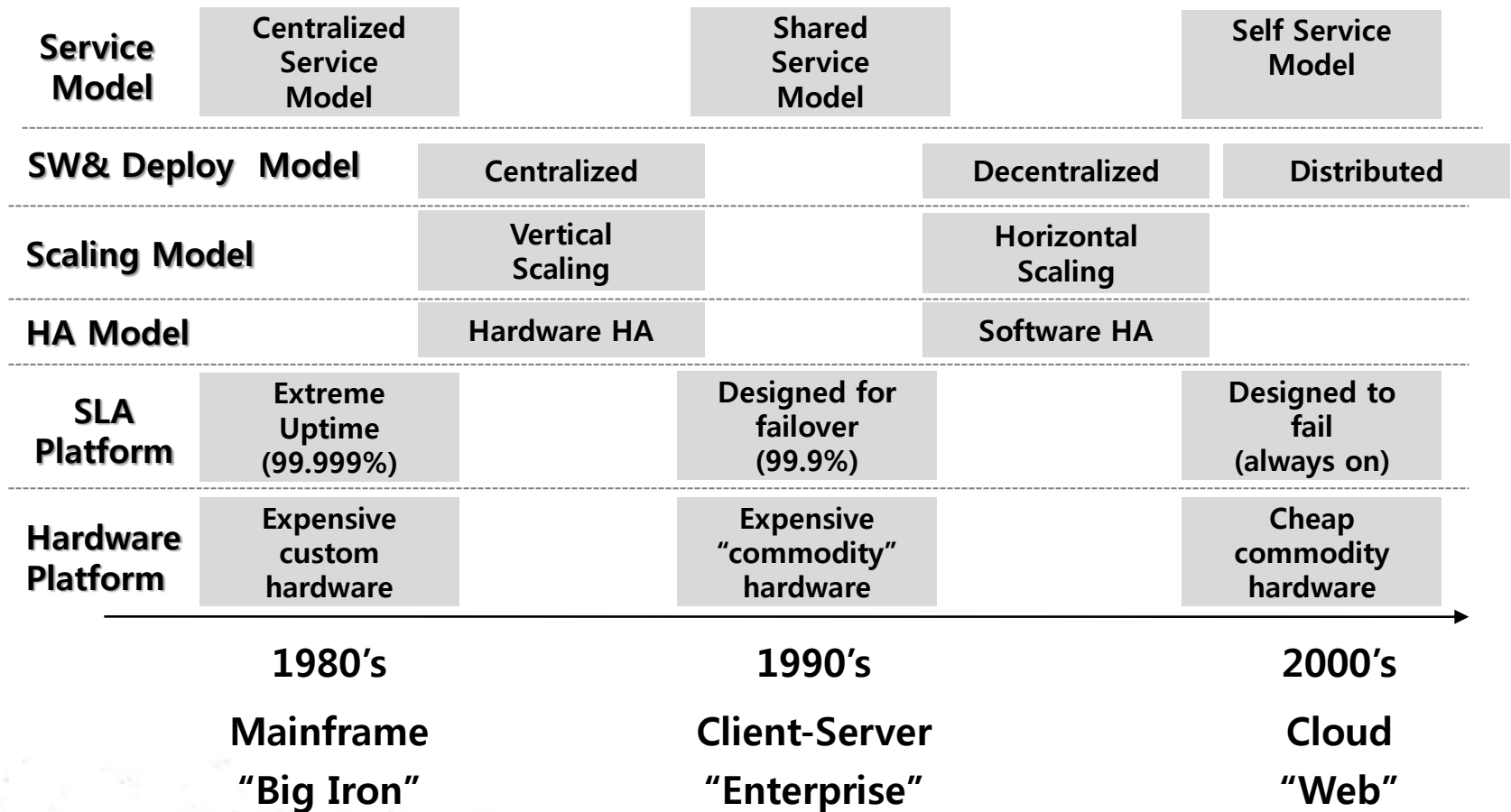
Scalable Cross Language Services Implementation

김병곤

fharenheit@gmail.com

JBoss User Group

# IT System Evolution



# 현대 IT System이 요구하는 속성

**Self-Service**

**Service Oriented**

**Easy**

**Management**

**Multi Core**

**Virtualization**

**Multi Device**

**Compact**

**Low Cost**

**Rapid Development**

# Mission

이기종의 다양한 언어를 지원할 방법은 없을까?

서비스의 최적화를 위해서는 다른 언어를 써야하는데...

서비스 구현만 집중할 수 없을까?

저사양 서버에서 고성능, 대용량 부하를  
지원하는 서비스를 만들 수 없을까?

# LAMP + Service



# Apache Thrift's Customers



**Cassandra**



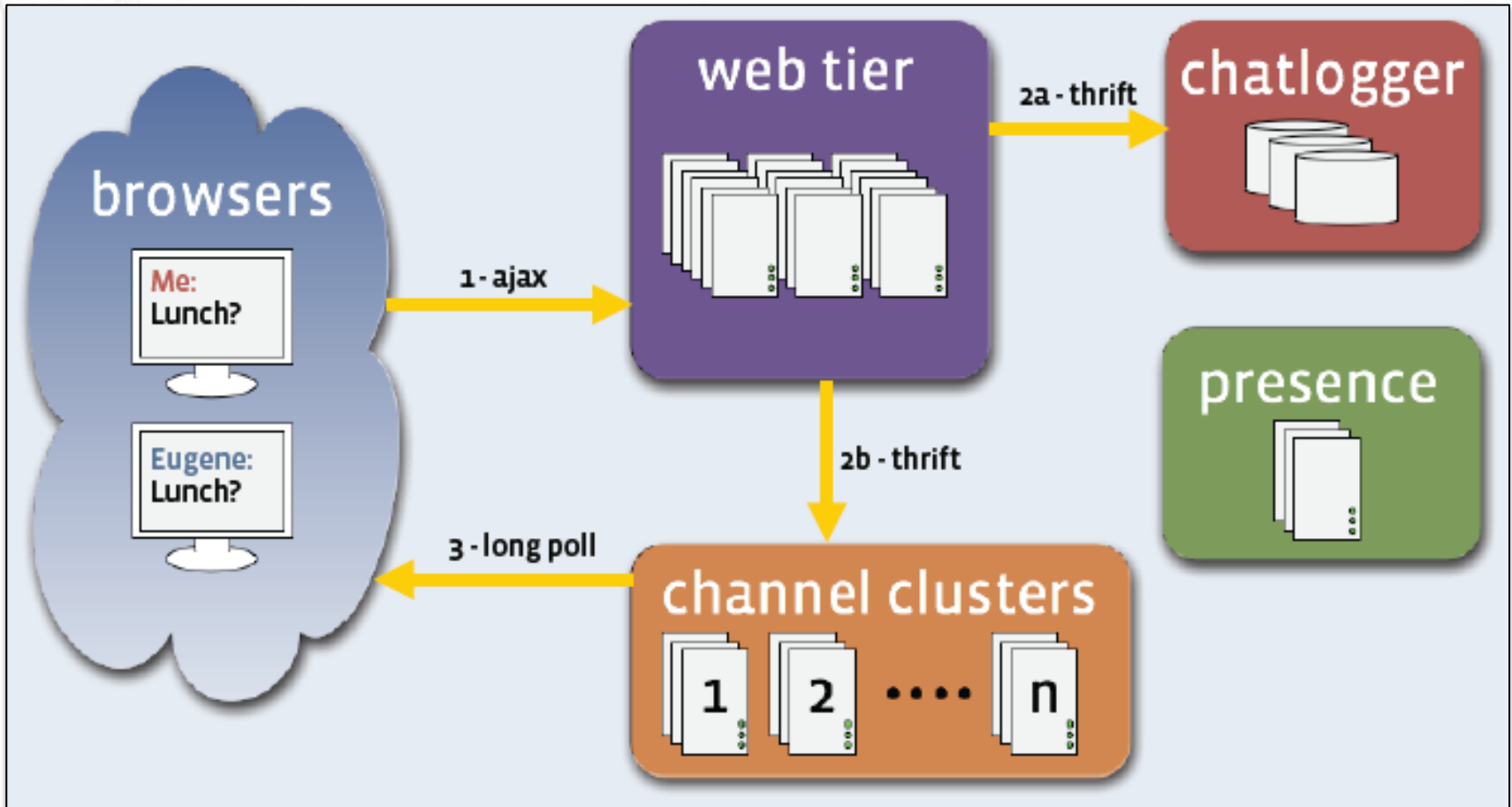
songza



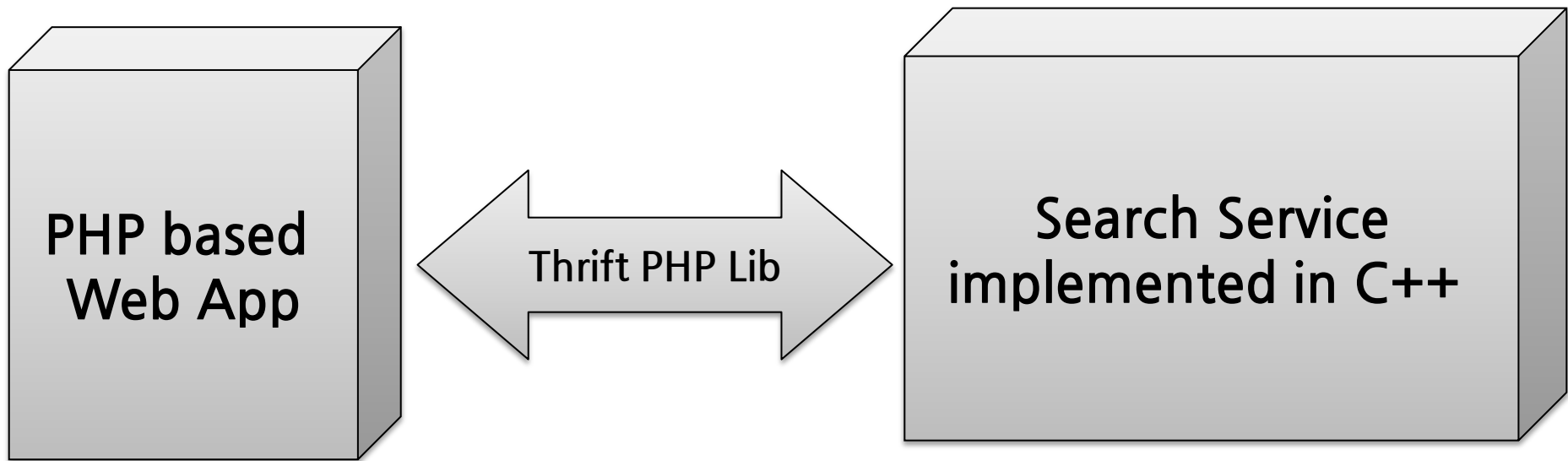
**facebook**



# Facebook Chat



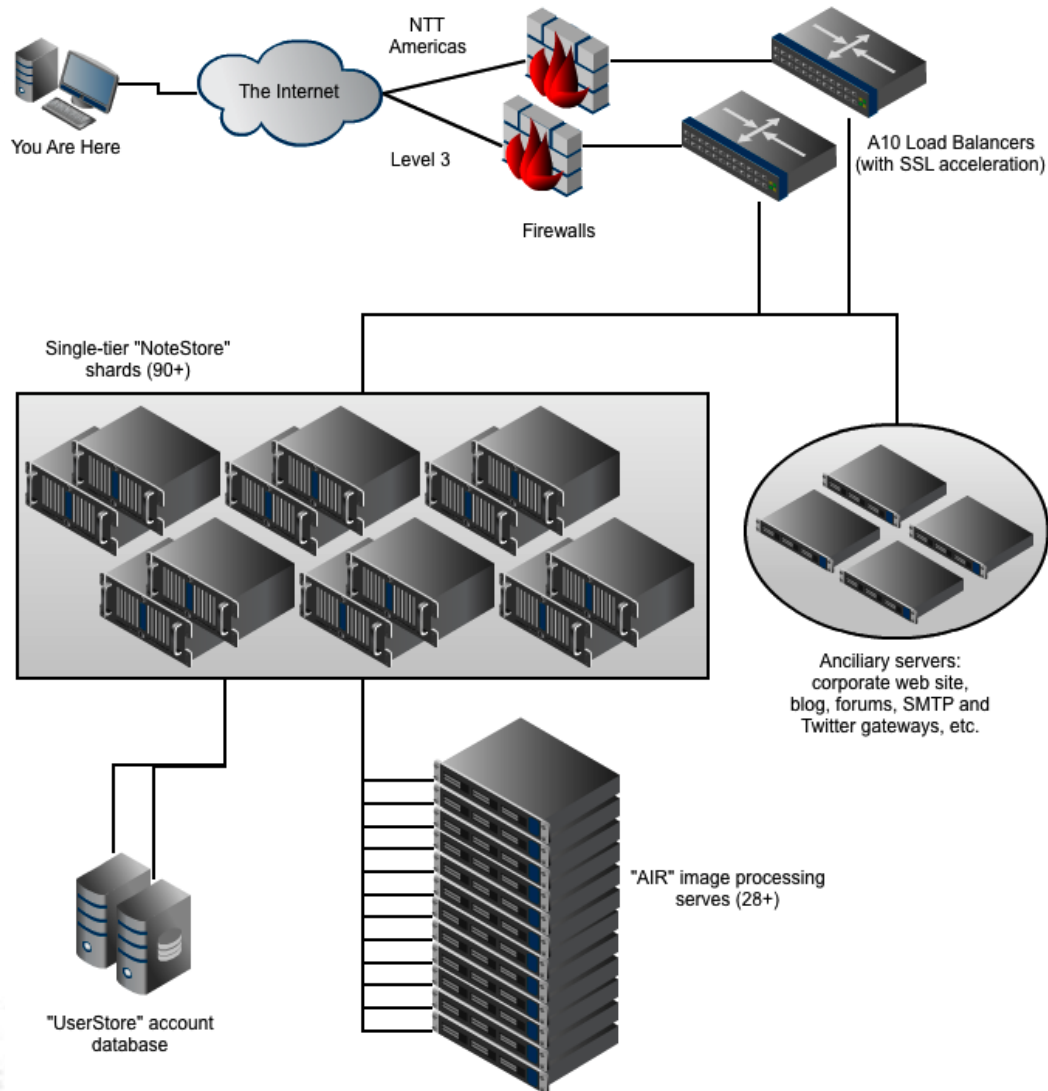
# Facebook Search



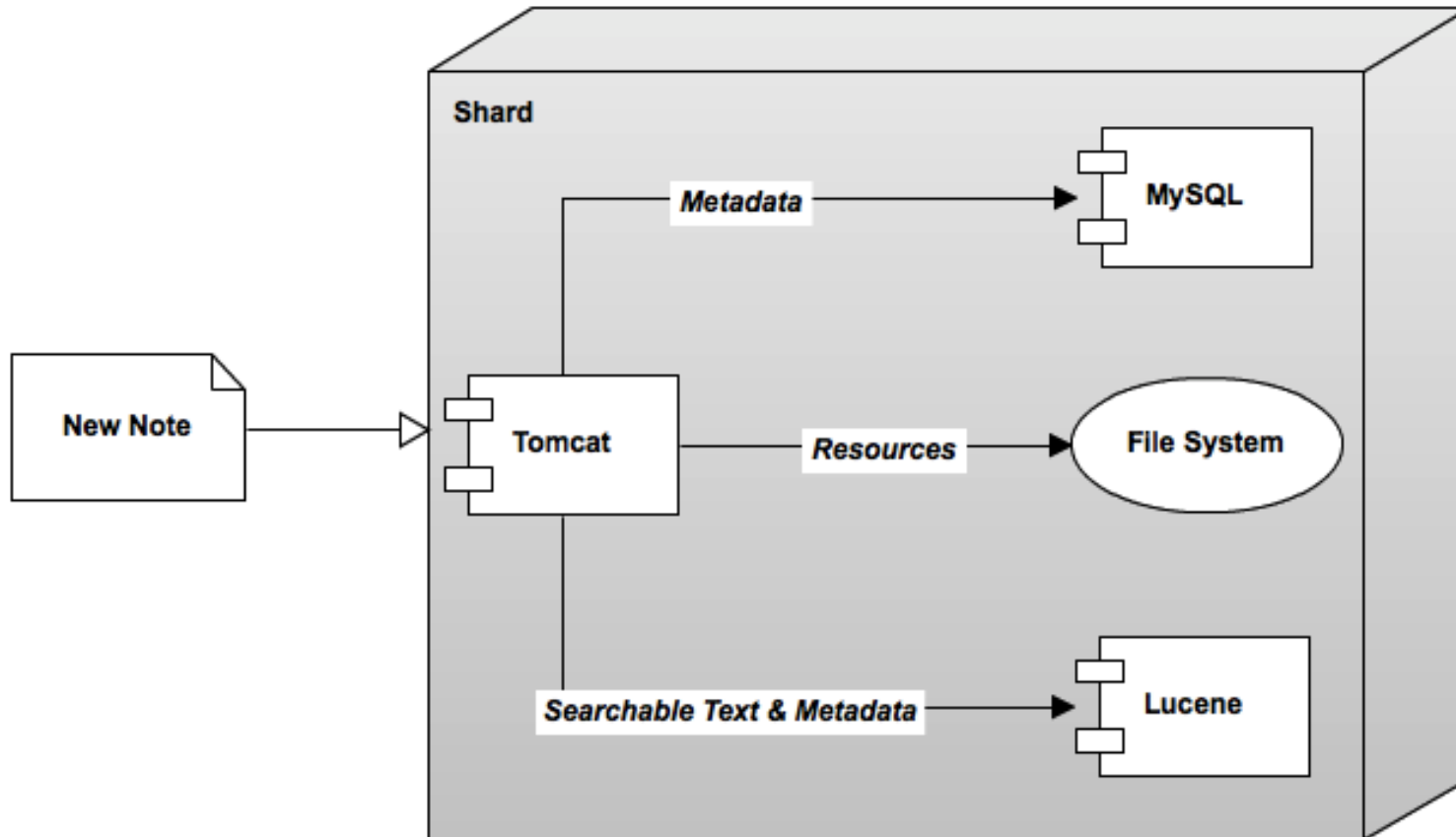
# Evernote



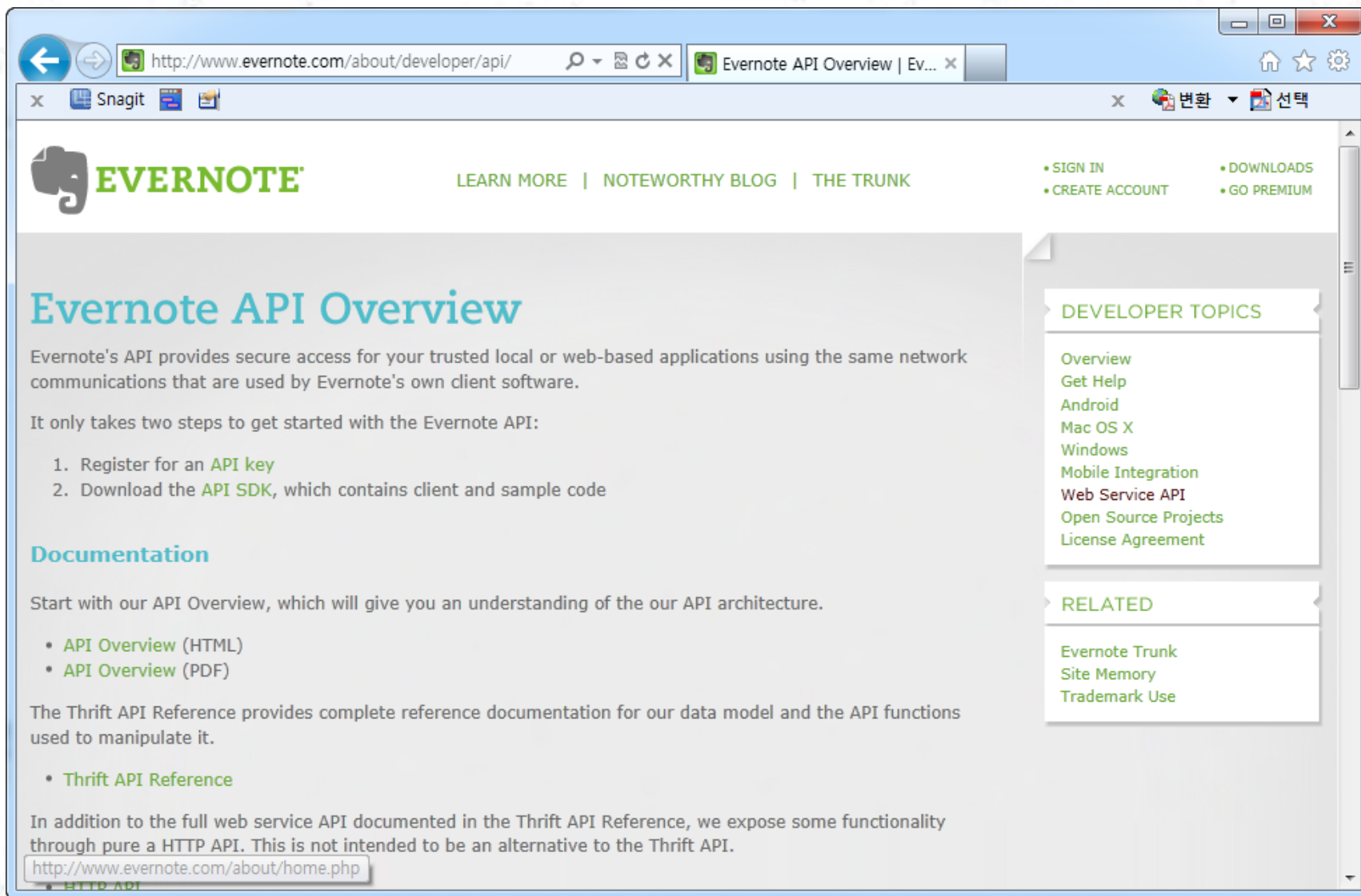
# Evernote Architecture



# Evernote Architecture



# Evernote Thrift SDK API



The screenshot shows a web browser window displaying the Evernote API Overview page. The browser's address bar shows the URL <http://www.evernote.com/about/developer/api/>. The page features the Evernote logo and navigation links for 'LEARN MORE', 'NOTEWORTHY BLOG', and 'THE TRUNK'. On the right side, there are links for 'SIGN IN', 'CREATE ACCOUNT', 'DOWNLOADS', and 'GO PREMIUM'. The main content area is titled 'Evernote API Overview' and includes a brief description of the API, a two-step process to get started, and a 'Documentation' section with links to HTML and PDF versions of the API Overview, and a Thrift API Reference. A sidebar on the right contains 'DEVELOPER TOPICS' and 'RELATED' sections with various links.

**EVERNOTE**      LEARN MORE | NOTEWORTHY BLOG | THE TRUNK      • SIGN IN      • DOWNLOADS  
• CREATE ACCOUNT      • GO PREMIUM

## Evernote API Overview

Evernote's API provides secure access for your trusted local or web-based applications using the same network communications that are used by Evernote's own client software.

It only takes two steps to get started with the Evernote API:

1. Register for an [API key](#)
2. Download the [API SDK](#), which contains client and sample code

### Documentation

Start with our [API Overview](#), which will give you an understanding of the our API architecture.

- [API Overview](#) (HTML)
- [API Overview](#) (PDF)

The [Thrift API Reference](#) provides complete reference documentation for our data model and the API functions used to manipulate it.

- [Thrift API Reference](#)

In addition to the full web service API documented in the [Thrift API Reference](#), we expose some functionality through [pure a HTTP API](#). This is not intended to be an alternative to the Thrift API.

<http://www.evernote.com/about/home.php>

#### DEVELOPER TOPICS

- [Overview](#)
- [Get Help](#)
- [Android](#)
- [Mac OS X](#)
- [Windows](#)
- [Mobile Integration](#)
- [Web Service API](#)
- [Open Source Projects](#)
- [License Agreement](#)

#### RELATED

- [Evernote Trunk](#)
- [Site Memory](#)
- [Trademark Use](#)

# Apache Thrift

*Software **Framework**  
for **scalable cross-language**  
**services development***

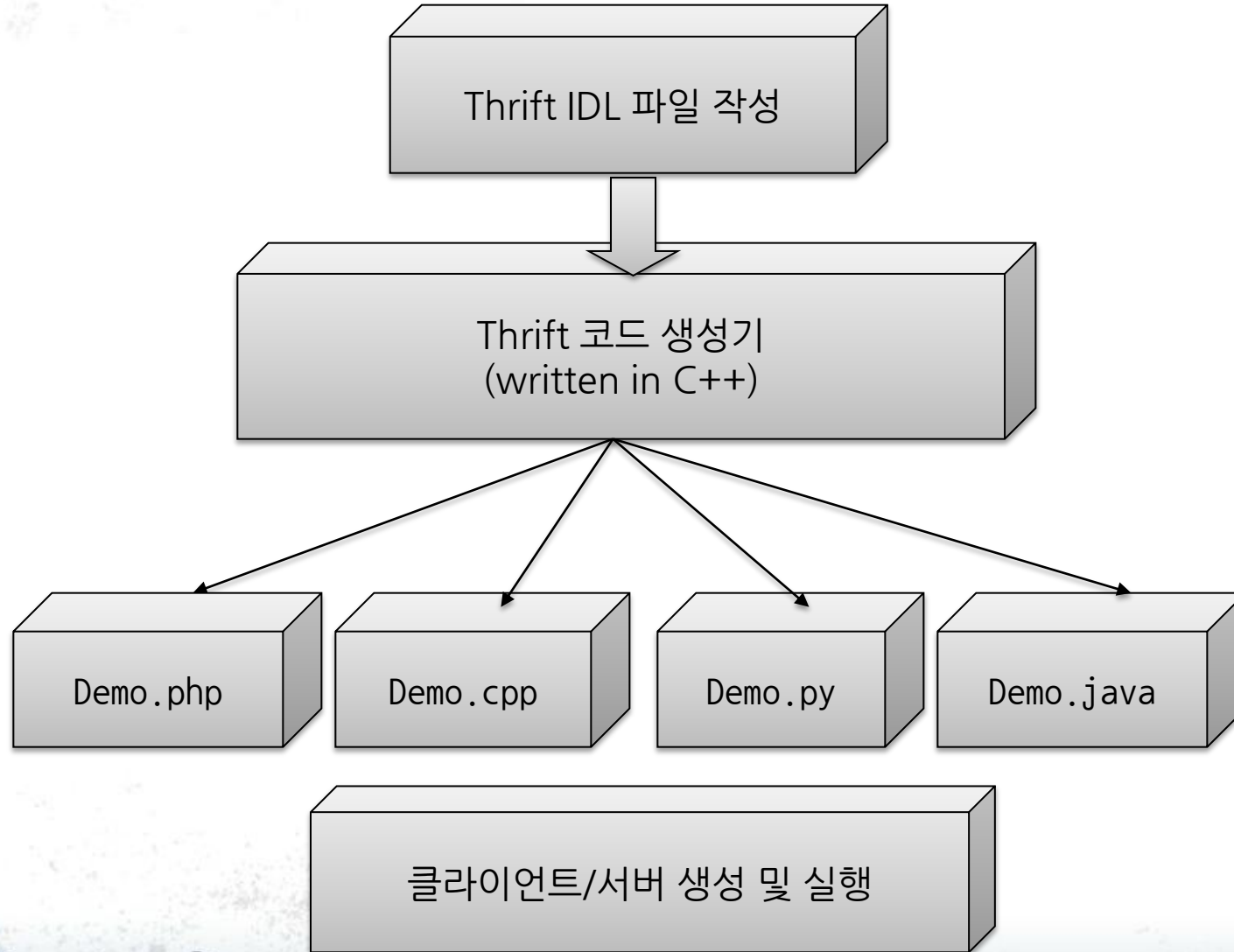
# Apache Thrift의 강점

- 불필요한 시간 낭비 최소화
- 유지보수 비용 최소화
- 다중 언어 직렬화
- 단순함
- XML 설정 없음!!
- 미리 정의되어 있는 직렬화 포맷 :: Binary, JSON, ...
- 프로토콜 버저닝
- 의존성 없음
- 네이티브 언어별 바인딩
- Application Level Wire Format과  
Serialization Level Wire Format의 분리

# History

- Facebook에서 개발
- 2007년 4월 오픈소스화
- 데이터 교환을 쉽게 하기 위한 목적
- 최소의 부하로 이기종 언어간 Serialization 지원
- Thrift 도구를 통해 다양한 언어용 코드를 자동 생성
  - C++, Java, C#, PHP, Python
  - Ruby, Erlang, Perl, Haskell, Cocoa, Smalltalk, Ocaml, ...
- 2008년 Apache Incubator
- 2010년 10월 Apache Top Level Project로 승격

# Thrift Application 개발 절차



# Thrift Compiler Requirement

*Thrift's compiler is written in C++ and designed to be portable, but there are some system requirements:*

- Basic requirements
  - A relatively POSIX-compliant \*NIX system
  - Cygwin or MinGW can be used on Windows
  - g++ 3.3.5+
  - boost 1.33.1+ (1.34.0 for building all tests)
  - Runtime libraries for lex and yacc might be needed for the compiler.
- Requirements for building from SVN
  - GNU build tools: autoconf 2.59+ (2.60+ recommended), automake 1.9+, libtool 1.5.24+
  - pkg-config autoconf macros (pkg.m4) (Use MacPorts for Mac OS X)
  - lex and yacc (developed primarily with flex and bison)
  - libssl-dev

# Language Requirement

- C++
  - Boost 1.33.1+
  - libevent (optional, to build the nonblocking server)
  - zlib (optional)
- Java
  - Java 1.5+
  - Apache Ant, Apache Ivy (recommended)
  - Apache Commons Lang (recommended)
  - SLF4J
- C#: Mono 1.2.4+ (and pkg-config to detect it) or Visual Studio 2005+
- Python 2.4+ (including header files for extension modules)
- PHP 5.0+ (optionally including header files for extension modules)
- Ruby 1.8+ (including header files for extension modules)
- Erlang R12 (R11 works but not recommended)
- Perl 5
  - Bit::Vector, Class::Accessor

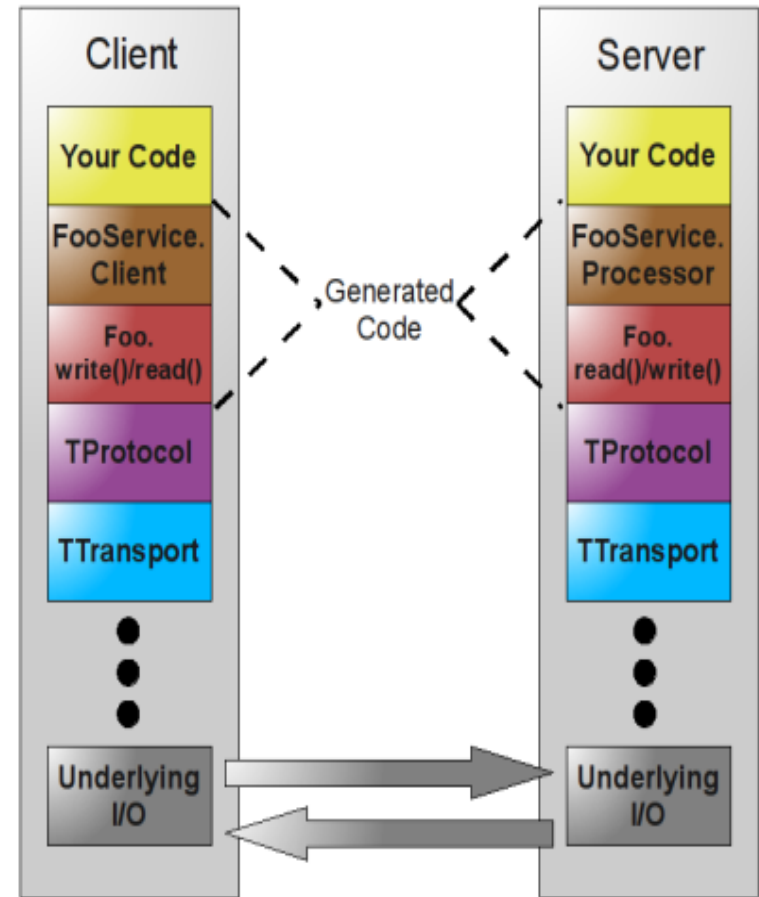
# Language Support

Language	Lang Features	Protocol Support				Transports		Servers			Clients		OS Support		
	Unions	Binary	Dense	Compact	JSON	Framed	SSL	Basic	Non- blocking	HTTP	Basic	HTTP	Win	OSX	Linux
Action Script 3 (as3)															
C Glib (c_glib)		0.6				0.6		0.6			0.6				0.6
C++(cpp)							0.7			0.4					
C# (csharp)					0.5		THRIFT-181			THRIFT-322					
Erlang (erl)															
Haskell (hs)															
Java (java)		0.2		0.2	0.2	0.2	0.5	0.2	0.2	0.4	0.2				
JavaScript (js)					0.3		0.3					0.3			
Node.js (js:node)		0.6				0.6			0.6		0.6				
Objective C (cocoa)															
OCaml (ocaml)															
Perl (perl)															
PHP (php)															
Python (py)							0.7								
Ruby (rb)															
Smalltalk(st)															

Legend	
Color	Description
green	available since version X.X
orange	patch available (JIRA-Ticket)
yellow	planned for version X.X
white	unknown
red	won't fix (JIRA-Ticket)

# Thrift Architecture

- Service Client/Server
- TProtocol
  - 프로토콜 처리의 추상화 계층
  - 프로토콜에 따른 다양한 구현체 제공
    - Binary, JSON, ...
- TTransport
  - 메시지 송수신의 추상화 계층
  - File, Frame, Compression, ...
- TServer
  - 소켓 서버
  - 다양한 구현체 제공
    - Blocking, Non-Blocking
    - Single or Multi-Thread



# IDL

```
struct Column {
    1: required binary name,
    2: optional binary value,
    3: optional i64 timestamp,
    4: optional i32 ttl,
}
struct SuperColumn {
    1: required binary name,
    2: required list<Column> columns,
}
enum ConsistencyLevel {
    ONE = 1,
    QUORUM = 2,
    LOCAL_QUORUM = 3,
}

exception InvalidRequestException {
    1: required string why
}

service Cassandra {
    void login(1: required
                AuthRequest auth_request)
    throws (1:AuthException authnx,
           2:AuthorizationException authzx),
    ...
}
```

<http://svn.apache.org/viewvc/cassandra/trunk/interface/cassandra.thrift?view=co>

# Thrift Server

```
// Create Non-Blocking Server Socket
final TNonblockingServerSocket socket =
    new TNonblockingServerSocket(PORT);

// Service Handler (Your Role!!)
final CourseService.Processor processor =
    new CourseService.Processor(new Handler());

// Create Server
final TServer server = new THsHaServer(processor, socket,
    new TFramedTransport.Factory(),
    new TCompactProtocol.Factory()
);

// Start Server
server.serve();
```

# Thrift Protocol

- 메시지의 유형에 따라 다양한 프로토콜 구현체 사용 가능
- Binary, JSON 등의 메시지 기본 구현체 제공

Protocol	Description
TBinaryProtocol	<ul style="list-style-type: none"><li>• 단순한 바이너리 프로토콜. 텍스트 인코딩 보다 효율적이지만 디버깅이 어려움.</li></ul>
TJSONProtocol	<ul style="list-style-type: none"><li>• JSON 포맷</li><li>• Read &amp; Write의 모든 기능을 지원하는 프로토콜</li></ul>
TSimpleJSONProtocol	<ul style="list-style-type: none"><li>• JSON 포맷</li><li>• Write-Only 프로토콜</li><li>• 스크립트 언어가 파싱하는데 적당</li><li>• TJSONProtocol과 혼동 가능성 있으므로 주의 필요</li></ul>
TCompactProtocol	<ul style="list-style-type: none"><li>• 매우 효율적인 dense encoding 프로토콜.</li></ul>

# Thrift Transport

- 메시지를 송수신하는 추상화 레이어 제공

Transport	Description
TFileTransport	<ul style="list-style-type: none"><li>• 파일을 기록하는 송수신 방식</li></ul>
TFramedTransport	<ul style="list-style-type: none"><li>• Non-Blocking IO 사용시 사용</li><li>• 데이터와 길이 정보를 포함하는 데이터를 송수신하는 방식</li></ul>
TSocket	<ul style="list-style-type: none"><li>• Blocking Socket I/O</li></ul>
TZlibTransport	<ul style="list-style-type: none"><li>• zlib를 이용한 압축</li><li>• 다른 Transport와 결합 가능</li></ul>
TMemoryTransport	<ul style="list-style-type: none"><li>• I/O 처리시 메모리 사용</li><li>• 자바의 경우 내부적으로 ByteArrayOutputStream을 사용</li></ul>
TFastFramedTransport	<ul style="list-style-type: none"><li>• TFramedTransport와 호환</li><li>• 버퍼의 재사용 지원</li><li>• Thread Not Safe</li></ul>
TSaslServerTransport	<ul style="list-style-type: none"><li>• SASL Server Negotiation을 지원</li></ul>

# Thrift Server

- I/O 처리 방식에 따른 서버 구현체
- 기본 Non-Blocking 방식 권장

Server	Description
THsHaServer	<ul style="list-style-type: none"><li>• TNonBlockingServer의 확장</li><li>• Half-Sync/Half-Async Server</li></ul>
TNonblockingServer	<ul style="list-style-type: none"><li>• Non-Blocking Tserver 구현체</li><li>• TFrmedTransport 사용해야 함</li></ul>
TThreadPoolServer	<ul style="list-style-type: none"><li>• Java의 내장 Thread Pool Management 기반</li></ul>
TServlet	<ul style="list-style-type: none"><li>• Servlet 구현체</li></ul>
TSimpleServer	<ul style="list-style-type: none"><li>• 테스트 용도</li><li>• Single Thread 서버</li></ul>

# Apache Thrift Compilation (Linux)

- 테스트 환경 : Ubuntu Linux 10.10 Server x64

- 설치

```
#apt-get install build-essential libboost-all-dev sun-java6-jdk  
python-all apache2 php5 libapache2-mod-php5
```

```
#wget http://apache.tt.co.kr/thrift/0.6.1/thrift-0.6.1.tar.gz
```

```
#tar xvfz thrift-0.6.1.tar.gz
```

```
#cd thrift-0.6.1
```

```
#!/configure;sudo make;sudo make install
```

```
#ls /usr/local/lib
```

```
libthrift.a libthrift.so libthrift.so.0.0.0 libthriftz.la libthriftz.so.0  
libthrift.la libthrift.so.0 libthriftz.a libthriftz.so libthriftz.so.0.0.0
```

- 설정

```
- export LD_LIBRARY_PATH=/usr/local/lib:$LD_LIBRARY_PATH
```

# Versioning

- 메시지 규격의 변화가 발생하는 경우
  - 메시지는 새로운 멤버를 추가해야 한다.
  - 함수에는 새로운 인자를 추가해야 한다.

```
struct Work {  
    1: i32 num1 = 0,  
    2: i32 num2,  
    3: Operation op,  
    4: optional string comment,  
}
```

# Versioning

- 새로운 필드가 추가된 경우
  - New Client, Old Server
    - 서버는 새로 추가된 필드의 ID를 이해하지 못하므로 무시함.
  - Old Client, New Server
    - 신규 서버는 새로 추가된 필드를 찾을 수 없으므로 Object에 값을 설정하지 않고 서버는 그대로 처리
- 필드가 삭제된 경우
  - New Client, Old Server
    - 서버가 삭제된 필드를 찾을 수 없으므로 기존과 동일하게 처리.
  - Old Client, New Server
    - 예전 클라이언트가 사용하지 않는 필드를 송신하게 되면 서버는 무시하게 됨
    - 그외 처리는 기존과 동일

# Thrift Service - IDL Definition

```
enum PhoneType {  
    "HOME",  
    "WORK",  
    "MOBILE"  
    "OTHER"  
}
```

```
struct Phone {  
    1: i32    id,  
    2: string number,  
    3: PhoneType type  
}
```

```
struct Person {  
    1: i32    id,  
    2: string firstName,  
    3: string lastName,  
    4: string email,  
    5: list<Phone> phones  
}
```

```
struct Course {  
    1: i32    id,  
    2: string number,  
    3: string name,  
    4: Person instructor,  
    5: string roomNumber,  
    6: list<Person> students  
}
```

# Thrift Service - IDL Definition

```
exception CourseNotFound {  
    1: string message  
}
```

```
exception UnacceptableCourse {  
    1: string message  
}
```

```
service CourseService {  
    list<string> getCourseInventory(),  
    Course getCourse(1:string courseNumber)  
                                     throws (1: CourseNotFound cnf),  
    void addCourse(1:Course course) throws (1: UnacceptableCourse uc)  
    void deleteCourse(1:string courseNumber)  
                                     throws (1: CourseNotFound cnf)  
}
```

# Thrift Service - Generate Source

```
#thrift --gen java course.thrift
#thrift --gen py course.thrift
```

```
|-- gen-java
|   |-- com
|       |-- example
|           |-- course
|               |-- thrift
|                   |-- gen
|                       |-- Course.java
|                       |-- CourseNotFoundException.java
|                       |-- CourseService.java
|                       |-- Person.java
|                       |-- Phone.java
|                       |-- UnacceptableCourseException.java
```

# Thrift Service - Java Server

```
private static class Handler implements CourseService.Iface {

    @Override
    public List<String> getCourseInventory() throws TException {
        return db.getCourseList();
    }

    @Override
    public Course getCourse(String courseNumber)
        throws CourseNotFoundException, TException {
        final Course dbCourse = db.getCourse(courseNumber);
        if (dbCourse != null) {
            return ConversionHelper.fromDbCourse(dbCourse);
        }

        return null;
    }
    ...
}
```

# Thrift Service - Java Server

```
// Create Non-Blocking Server Socket
final TNonblockingServerSocket socket =
    new TNonblockingServerSocket(PORT);

// Service Handler (Your Role!!)
final CourseService.Processor processor =
    new CourseService.Processor(new Handler());

// Create Server
final TServer server = new THsHaServer(processor, socket,
    new TFramedTransport.Factory(),
    new TCompactProtocol.Factory()
);

// Start Server
server.serve();
```

# Thrift Service - Java Client

```
//Setup the transport and protocol
final TSocket socket = new TSocket(HOST, PORT);
socket.setTimeout(SOCKET_TIMEOUT);
final TTransport transport = new TFramedTransport(socket);
final TProtocol protocol = new TCompactProtocol(transport);

final CourseService.Client client = new CourseService.Client(protocol);

//The transport must be opened before you can begin using
transport.open();

//All hooked up, start using the service
List<String> classInv = client.getCourseInventory();
System.out.println("Received " + classInv.size() + " class(es).");

client.deleteCourse("WINDOWS_301");

classInv = client.getCourseInventory();
System.out.println("Received " + classInv.size() + " class(es).");

transport.close();
```

# Thrift Service - Python Client

```
#Setup the transport and protocol
socket = TSocket.TSocket("localhost", 8000)
socket._timeout = 1000
transport = TTransport.TFramedTransport(socket)
protocol_factory = TBinaryProtocol.TBinaryProtocolFactory()
protocol = protocol_factory.getProtocol(transport)

client = Client(protocol)

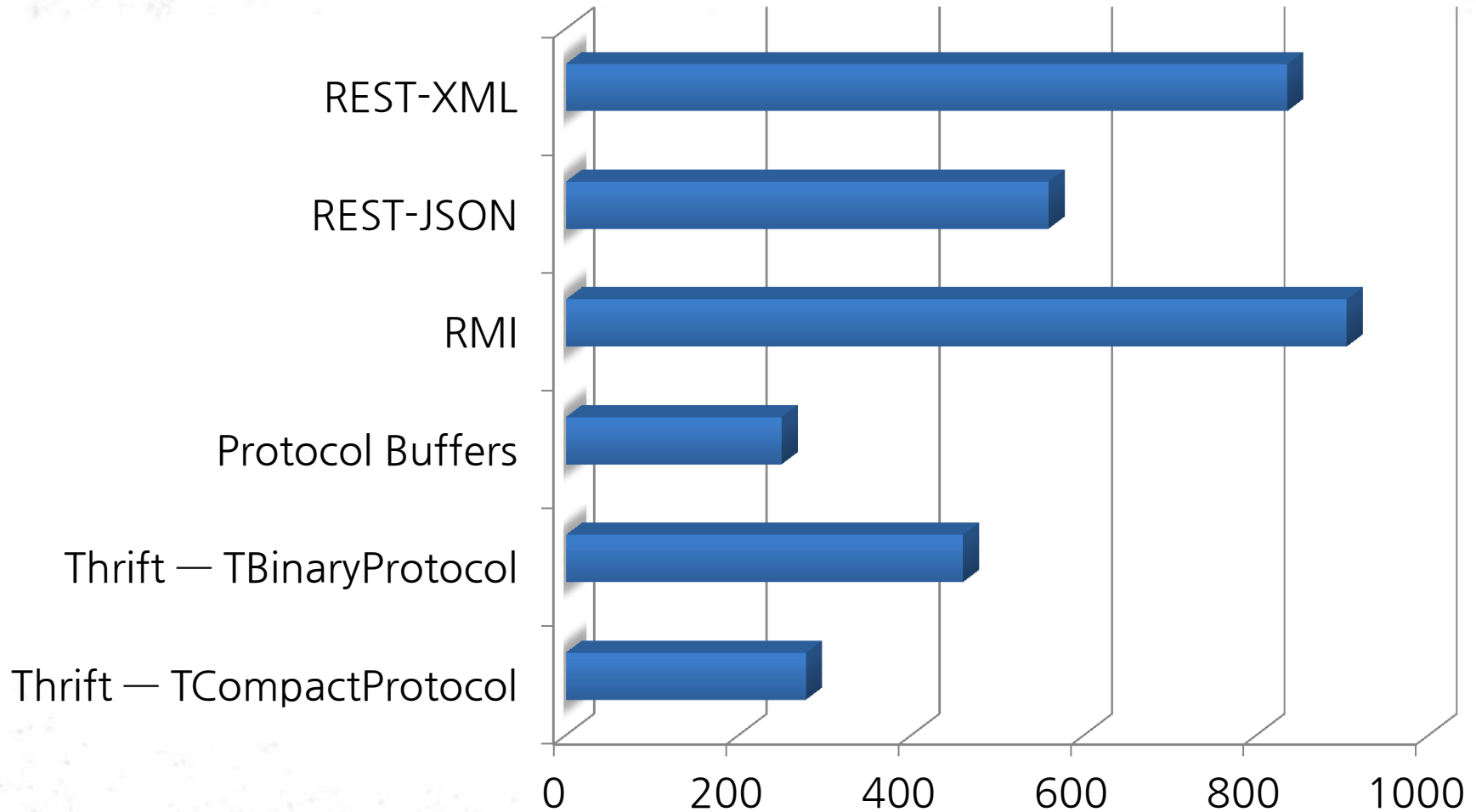
#The transport must be opened before you can begin using
transport.open()

classInv = client.getCourseInventory()
print "Received", len(classInv), "class(es)"

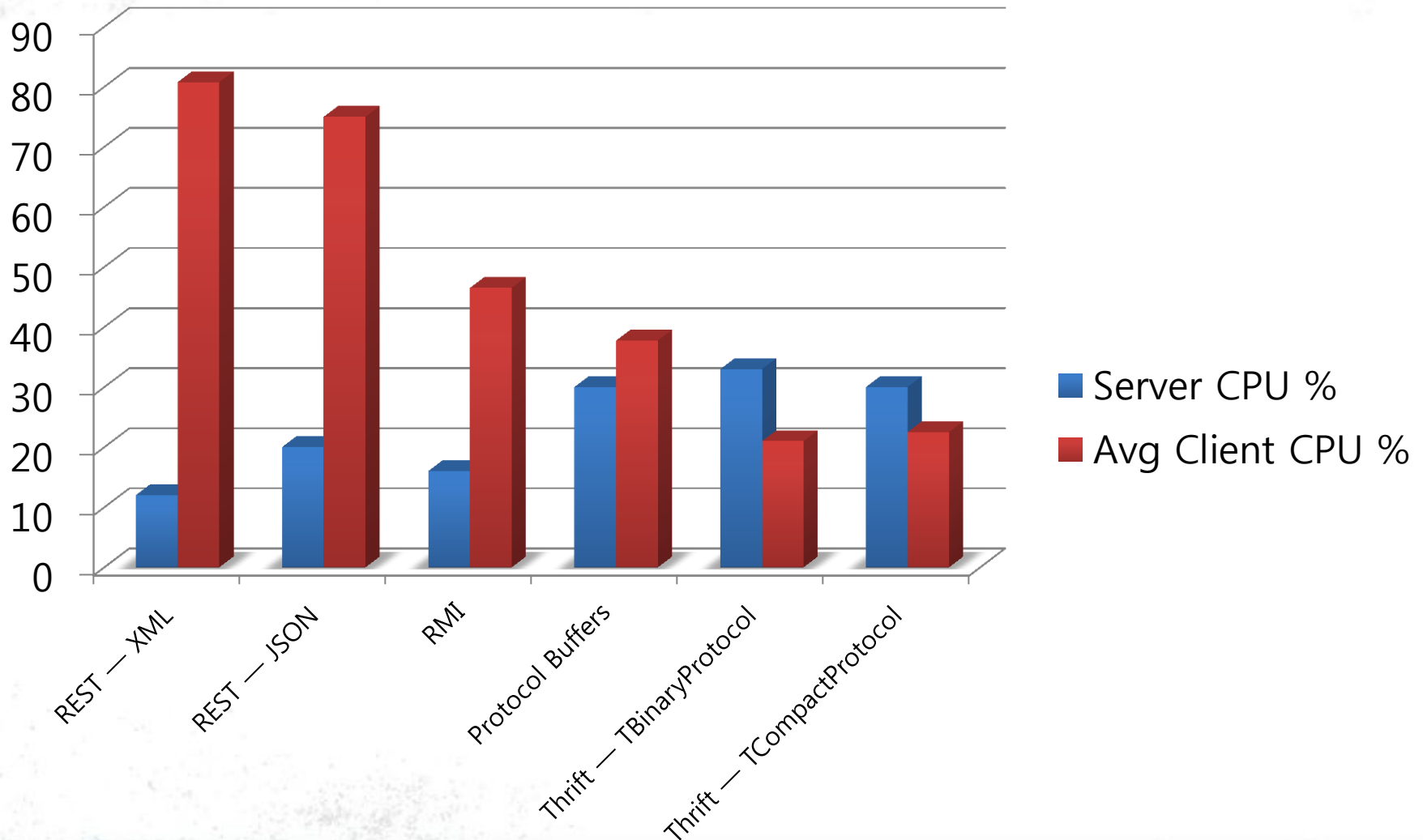
client.deleteCourse("WINDOWS_301")

classInv = client.getCourseInventory()
print "Received", len(classInv), "class(es)"
```

# Comparing Thrift - Size (Bytes)



# Comparing Thrift - CPU%



# Comparing Thrift - Size/CPU

Protocol	Size(bytes)	% Larger than TCompactProtocol
<b>Thrift — TCompactProtocol</b>	<b>278</b>	<b>N/A</b>
Thrift — TBinaryProtocol	460	65%
<b>Protocol Buffers</b>	<b>250</b>	<b>-10%</b>
RMI	905	225%
REST-JSON	559	101%
REST-XML	836	200%

Protocol	Server CPU %	Avg Client CPU %	Avg Wall Time
REST — XML	12	80.75	05:27.45
REST — JSON	20	75	04:44.83
RMI	16	46.5	02:14.54
<b>Protocol Buffers</b>	<b>30</b>	<b>37.75</b>	<b>01:19.48</b>
Thrift — TBinaryProtocol	33	21	01:13.65
<b>Thrift — TCompactProtocol</b>	<b>30</b>	<b>22.5</b>	<b>01:05.12</b>

# JBoss Community

JBoss Community (<http://www.jboss.org>)  
Korea JBoss User Group (<http://cafe.naver.com/jbossug>)

