WebServices e Load Balancing

Alberto Mulazzani Chiara Ceccarini

alberto.mulazzani@studio.unibo.it
chiara.ceccarini5@studio.unibo.it

Universita' di Bologna

February 1, 2017

Outline

Introduction

Definition

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.[W3C, 2004]

- The purpose of a WebService is to offer an abstract set of functionalities.

Communicating with WebServices

- WebServices communicate by exchanging messages and communication can be syncronous or asynchronous. The message exchanging mechanism is documented in a WebService Description (WSD): a machine-processable specification of the WS interface, written in WSDL. It defines the message formats, datatypes, trasport protocol and the expected return message. The WSD is an agreement about the interaction between services.
- WSD is usually used in Service Oriented Architecture, but in this case, as we use JSON as format, we used JSON APIs to determine the correct structure of a JSON Message for our exercises.

Definition

JSON API is a specification for how a client should request that resources be fetched or modified, and how a server should respond to those requests.[JSON API]

Architectural Approaches: SOA

Service Oriented Architectures (SOA) are composed by services that communicate between each other exchanging datas and informations. Every service has the capability to modify itself independently from the others, in such a way that the integration of the services remains intact.[Joydip Kanjilal, 2013]

- HTTP — as the underlying transport protocol
- SOAP — as the real transport protocol
- WSDL — for service description
- XML — for formatting the messages exchanged
- WS-* — set of specifications for handling high-level features

Architectural Approaches: ROA

Resource Oriented Architectures (ROA) concentrate on the concept of resource, a component that could be accessed by a standard interface. Every resource is associated to an univocous identifier: URI. ROA is a specific architecture for a RESTful service, so you work on the resources using HTTP methods as GET, POST, PUT and DELETE and using stateless requests.

- HTTP — as the real transport protocol
- XML — for formatting the messages exchanged
- WADL — for service description [Oracle (2016)]

Architectural Approaches: ROA vs SOA I

Attribute	Resource Oriented	Service Oriented
Granularity	Resource instances	Service instances
Support for caching responses	Yes	No
Payload	No, you have nothing that is linked to a particular address or URL	Yes, the WSDL schema
Addressing or request routing	Unique address of a particular resource	Endpoint address of the service
Coupling between server and client	Loose coupling due to late binding to resource data	Loose coupling due to late binding to service interfaces

Figure: ROA vs SOA.[Joydip Kanjilal, 2013]

Architectural Approaches: ROA vs SOA II

The different choice between SOA or ROA fully depends on the needed usage. SOA is better used in applications requiring reusability and interoperability. ROA is a simplier architecture that doesn't require a consistent XML message exchange, but uses everything offered by HTTP.

Introduction

JSON (JavaScript Object Notation) is suitable format for data exchange in client-server applications. It's easily understandable by men and machines. JSON is completely language independent format, so it's ideal in a great number of different cases. JSON bases itself on two structures: [JSON.org]

- a collection of name/value pairs, realised in programming languages as objects, records, dictionaries...
- an ordered list of values, realised in programming languages as an array, vector and so on.

JSON Object

A JSON Object defines a set of determined properties associated to values. An object can contain a number of properties that can also contain JSON Objects or JSON Arrays.

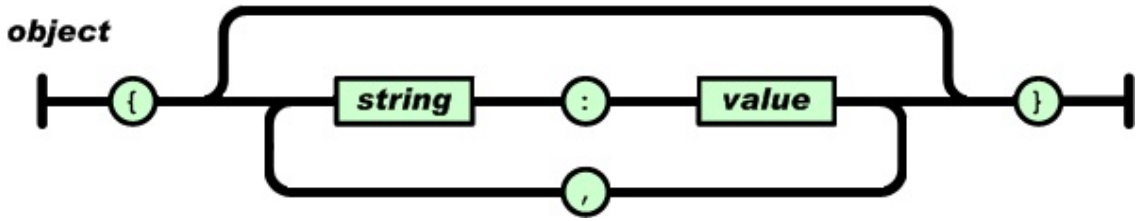

Figure: JSON Object structure.[JSON.org]

```
"links": {
  "self": "http://example.com/articles",
  "next": "http://example.com/articles2",
  "last": "http://example.com/articles3"
}
```

Listing 1: A JSON Object.

JSON Array

A JSON Array defines a series of values that can be JSON Objects or JSON Arrays.

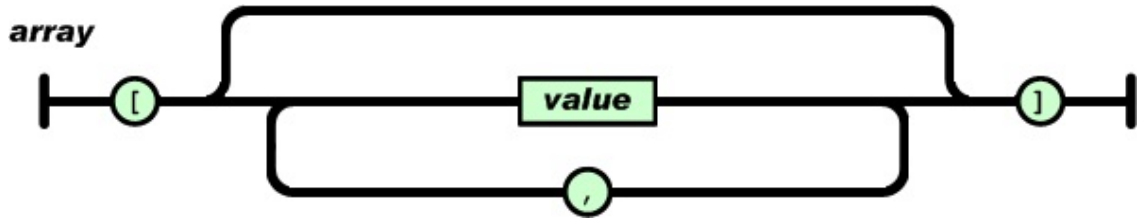

Figure: JSON Array structure.[JSON.org]

```
"data": [
  { "type": "1"},
  { "type": "2"}
]
```

Listing 2: A JSON Array.

SOAP and JSON messages : SOAP

A SOAP message is an XML document that contains this set of elements:[w3c]

1. An Envelope — The root of the SOAP message, it defines the XML as a SOAP message.
2. A Header — An optional element that contains application-specific information. If present, it must be the first child element of the Envelope.
3. A Body — Contains call and response information. It's the actual SOAP message.

SOAP and JSON messages : JSON

A JSON message is a combination of JSON Objects and JSON Arrays.

object
- {}
- { *members* }

members
- *pair*
- *pair* , *members*

pair
- *string* : *value*

array
- []
- [*elements*]

elements
- *value*
- *value* , *elements*

value
- *string*
- *number*
- *object*
- *array*
- **true**
- **false**
- **null**

SOAP vs JSON

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
   <soap:Header>...</soap:Header>
   <soap:Body>...</soap:Body>
</soap:Envelope>
```

Figure: A SOAP Message.[W3C]

```
"array" : [
{ "key" : "value" }
{ "key" : "value" }
]
```

Listing 4: A JSON Message.

Using JSON in PHP : json_encode

json_encode

```
string json_encode ( mixed $value [, int $options = 0
    [, int $depth = 512 ]] )
```

Returns the string containing the JSON representation of $value or FALSE on failure.

$value can be any type except a resource. Strings must be UTF-8 encoded.[PHPe]

Using JSON in PHP : json_decode

json_decode

```
mixed json_decode ( string $json [, bool $assoc =
    false [, int $depth = 512 [, int $options = 0 ]]]
    )
```

Takes a JSON encoded string and converts it to a PHP variable.

$json: Is the JSON string being processed.

The function returns an appropriate PHP type for the encoded JSON or NULL if the JSON cannot be decoded of if the data is deeper than the recursion limit. [PHPd]

JSON message structure

```
{"name":"Matrix",
"year":"1999"}
```

Listing 3: JSON Message Structure.

- ”name” is a string containing the Name of the movie.
- ”year” is the Year the movie came out.

This is the JSON message structure we’ll be using in the exercises.

Building a WebService : getConn.php

```php
function getInfo() {
  $conn = getConn(); //Connection to the DB
  $sql = "SELECT name, year FROM movies";
  $res = $conn->query($sql);
  $conn->close();
  //The query returns a set of pair <name, year>
  //from the db
  return $res;
}
```

Figure: getConn.php – getInfo()

Building a WebService : index.php I

```
if(isset($_POST["submit"])) {
    /*$res = getInfo();
    $jsonArray = convertToJSon($res);

   //$res is an organized PHP structure and convertToJSon is the method that
   //will transform that structure in a JSON Array.
   //After convertToJSon it's done, convert to a simplier php structure, like an array
   //and write each title and year in the section below.

        <section id ='movies'>
        <p><strong>Title:</strong> ". TITLE OF THE MOVIE ."</p>
        <p><strong>Year:</strong> ". YEAR OF THE MOVIE ."</p>
        </section>";
     */
```

Figure: index.php

Building a WebService : index.php II

```
function convertToJSon($info) {
if($info->num_rows>0) {
    $jsonArray= array();
    while($row = $info->fetch_assoc()) {
    //Iterates on every row of $info.
    //the elements could be accessed like a c# Dictionary
    //info[key] = value. You should create a jsonObject
    //for every movie and return a JSON array that contains them.
    }
}
return null;
}
```

Figure: index.php – convertToJSon()

Goals

1. **convertToJSON** should return a JSON Array, where every cell is a JSON Object, made of two pair <key,value>: one for the movie's name and the other for its year.
2. **index** should transform the JSON Array into a PHP structure, like an array, and print the movie's info contained in it.

Introduction

Definition

Load balancing refers to efficiently distributing incoming network traffic across a group of backend servers, also known as a server farm or server pool.[NGINX]

A Load Balancer aim is to optimize the resources use, minimize response time and avoid the overload of the servers.
A dedicated Load balancer:

- Distributes client requests or network load across multiple servers.
- Ensures availability and reliability by continuously monitoring server's activity and status.
- It's scalable and it lets server subscribe or unsubscribe from the server-list.

But we can have different types of Load Balancer.

Types of load balancers

There are different types of load balancers. Load balancers could be client-side or server-side.

- Server-side — A Server-side load balancer is an added layer, built on the server, between the client and the server itself that distributes the load between the servers. It could also be a dedicated software (E.g. Nginx, AWS Load Balancer etc...).
- Client-side — A Client-side load balancer is an added logic to the client application itself. If the Server-side load balancer distributes the load between the servers, the client-side distributes the request itself to the right server. It's easier to implement, but also has downsides (E.g. It can't monitor the server status or the server load).

Types of load balancers : Algorithms

Load-balancing could be achieved by different algorithms, for example:[pipelink][JSCAPE (2015)]

- Round Robin — Classical Round robin algorithm, it assigns a number to every server and distributes evenly the requests in a circular way (E.g. 4 Servers A,B,C,D and 12 request: ABCDABCDABCD).
- Priority or Weighted Round Robin — Similar to Round Robin but you assign a priority or a weight to each server, so more requests are given to the server with the highest priority or the least weight.
- Client-Randomizer — The client has the list of all the different servers and decides on-the-fly which server to use in a random way. For the Law of Large Numbers the load will be reasonably evenly distributed across the servers.
- Least-Connection, Scheluded, Lowest Latency, Least-Used etc...

Request.php

```php
$webServices = array("ws1/", "ws2/", "ws3/", "ws4/", "ws5/", "Default");
//Folders where out WebServices are stored
//they could be also added by Subscribing

$requestUrl = "";

$server = 5;
//Server must be changed to the right server to send the request to.

$_SESSION[$webServices[$server]] = $_SESSION[$webServices[$server]] + 1;
//Build the url to send the request to the right WS
//the url should be wsX/getConn.php where X is a number from 1 to 5
//or the index from 0 to 4, 5 is only used as a default setting.

$res = null;
return $res;
```

Figure: Exercise 2 : request.php

Goals

1. **requestUrl** shall become the path to the WebService.
2. **server** shall become the index of the chosen WebService.
3. **res** shall become the content of getConn.getInfo() of the chosen WebService.
4. The load **must** be evenly divided among the WSs.

Conclusion

1. Does the load balancer work?
2. What can we do to improve it?
3. Is it worth it?

Does the load balancer work?

Yes, it works. It may be not so easy to understand with a low number of request, but these are the results with higher numbers:

On a total of 5000 requests
ws1 processed the 20.12% of the requests
ws2 processed the 20.24% of the requests
ws3 processed the 19.94% of the requests
ws4 processed the 19.68% of the requests
ws5 processed the 20.02% of the requests

Figure: 5000 requests.

On a total of 500000 requests
ws1 processed the 19.9554% of the requests
ws2 processed the 20.0368% of the requests
ws3 processed the 20.019% of the requests
ws4 processed the 20.0714% of the requests
ws5 processed the 19.9174% of the requests

Figure: 500000 requests.

On a total of 5000000 requests
ws1 processed the 19.99244% of the requests
ws2 processed the 20.00704% of the requests
ws3 processed the 20.02714% of the requests
ws4 processed the 19.96554% of the requests
ws5 processed the 20.00784% of the requests

Figure: 5000000 requests.

On a total of 50000000 requests
ws1 processed the 19.993556% of the requests
ws2 processed the 20.006094% of the requests
ws3 processed the 19.99522% of the requests
ws4 processed the 20.005932% of the requests
ws5 processed the 19.999198% of the requests

Figure: 50000000 requests.

What can we do to improve it?

We used a Client-side randomizer that distributed evenly the number of requests among the WSs, but we can't know if the actual work of the machine will be the same or not. With this kind of balancing, as we said before, we can't control our server's workload and we can't know their status. If a server goes down we just lose that part of requests, until we fix it or we remove it from the server-list. We could use a much more complicated and complete load balancer that will surely improve further our performance.

Is it worth it?

Is it worth to use a Load Balancer? Yes it is. If your application has a great number of requests or a heavy workload, a load balancer is what you need. One of the most important things in the late years is to minimize the response time of our applications, load balancers help us reducing it, making the application and it's usage better for the user.

References I

W3C Working Group (2004)

Web Services Architecture

www.w3.org/TR/ws-arch/

JSON API

Latest Specification (v1.0)

jsonapi.org/format/

Joydip Kanjilal (2013)

ASP.NET Web API

Oracle (2016)

WADL home

http://wadl.java.net

JSON.org

Introduction to JSON

http://www.json.org/index.html

References II

W3C Working Group

XML Soap

http://www.w3schools.com/xml/xml_soap.asp

PHP

json_encode

https://secure.php.net/manual/en/function.json-encode.php

PHP

json_decode

https://secure.php.net/manual/en/function.json-decode.php

NGINX

WHAT IS LOAD BALANCING?

https://www.nginx.com/resources/glossary/load-balancing/

References III

pipelink

Load Balancing Methods & Algorithms

https://www.peplink.com/technology/load-balancing-algorithms/

JSCAPE (2015)

Comparing Load Balancing Algorithms

http://www.jscape.com/blog/load-balancing-algorithms

WebServices e Load Balancing

Alberto Mulazzani Chiara Ceccarini

alberto.mulazzani@studio.unibo.it
chiara.ceccarini5@studio.unibo.it

Universita' di Bologna

February 1, 2017