Large Scale Web Application Chat

Tugdual de Kerviler

Supervisor: Nicolas Bonvin
Professor: Karl Aberer

Distributed Information Systems Laboratory
Outline

- **Real time web**
  - Polling, long polling, HTTP Streaming...

- **Push technologies**
  - Push engines, new server features

- **Architecture**
  - node.js, MongoDB

- **The chat application**
  - Demo

- **Test performance**
  - Benchmarking tools, results
Real time web

- HTTP

GET http://www.google.fr/

Listen on port 80

Connection closed

HTML response
Real time web

- Refresh the page?
Real time web

<table>
<thead>
<tr>
<th>GET</th>
<th>HTML response</th>
<th>GET</th>
<th>HTML response</th>
<th>GET</th>
<th>HTML response</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Listen on port 80</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Image of a laptop and a person sleeping]
Real time web

JavaScript doing F5 ?

Simple polling:

An XMLHttpRequest every X seconds

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
</table>
| • Very simple to implement  
  • Compatible with every browsers « AJAX ready » | • Scalability (waste of ressources in bandwidth and processing) |
Real time web

Remember 56K modems?

GET http://www.google.fr/

Listen on port 80

Connection closed

HTML response
Real time web

Comet

GET

Listen on port 80

Connection closed
Connection opened

HTML response
Long polling

**Comet**

### +
- Compatibility
- Scalability

### -
- Harder to implement
- Server: the « C10K problem »
- Client: handle browser’s timeout
Comet

HTTP Streaming
HTTP 1.1: Chunked Transfer Encoding

First idea: iFrame sent in chunks containing JavaScript

Better: Comet + HTTP Streaming Service streaming
# Comet

## Comet + HTTP Streaming

<table>
<thead>
<tr>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Same advantages as long polling</td>
<td></td>
</tr>
<tr>
<td>• No callback for each chunk: Need to periodically check the content of xhr.responseText</td>
<td></td>
</tr>
<tr>
<td>• Same disadvantages as long polling</td>
<td></td>
</tr>
</tbody>
</table>
Applet

Flash, Java...

Not restricted by HTTP.
Push technologies

Push Engines
  Ape-project, Orbited

Servers
  Apache, Tomcat, Glassfish, Jetty, Tornado
Push technologies

Push Engines

Cross-domain

HTTP Server

Push Server

Client

Comet

HTML page:
<script src="http://Push_Server_Address/static/script.js"/>
Push technologies

JSON: `{key1: value1, key2: [value2, value3]}`

AJAX: `jsObj = eval('('+response+')')`

jQuery: `{ dataType: 'json' }`

JSONP: for each query, modify the page to add:

`<script src='http://Push_Server_Address/getMessages?jsonp=callback'>`
Push technologies

Servers

- The C10K problem

In a thread-based server using blocking I/O, a thread is created for each request.

When a thread calls an I/O function like:

```php
$res = mysql_query($query);
```

It blocks and waits for the function to return.

So each thread is going to wait until it responds.

But it’s working right?
Push technologies

Servers
- The C10K problem

Each thread uses some memory!!

C10K refers to the limit of servers to handle more than 10,000 requests at the same time.

Comet: new event can take a really long time.

=> Explosion of the number of threads!
Push technologies

Servers
  - Java servers:

    ARP « Asynchronous Request Processing »
    Servers must implement NIO:
    « new I/O » or « non-blocking I/O »

Framework: Atmosphere/Grizzly
Event-based servers

- Nginx
- Lighttpd
- Node.js

Event-based?
Event-based servers

Event-based

- 1 main thread
- 1 thread per I/O
- I/O threads can notify main thread when "something new happens". For this, they use events.

*Event: significant change in state*
Event-based servers

So what?
Event-based servers

Memory ?
Node.js

Evented IO for V8?

- V8: a powerful JavaScript Engine. Developed by Google.
  Compiles code before executing.

Wait what?

**JavaScript ??**

Yes, server-side JavaScript!
Why server-side JavaScript?

JavaScript has interesting functionalities:

- Anonymous functions
- Closures
Anonymous functions?

```javascript
// declaring an anonymous function
var square = function (x) { return x*x; }
```
Node.JS

- Anonymous functions?

Callbacks:

```javascript
fs.readFile("myfile.txt", function (err, data) {
    if (err) throw err;
    // print content
    sys.puts(data);
});
```
Anonymous functions?

```javascript
// file fs.js
exports.readFile = function(filePath, callback) {
  if (fork() == 0) { // create new process
    try { // try to read the file
      f = open(filePath);
      buff = ""
      while ((byteRead = f.read(1000, buff)) != 0) {
        callback(null, buff);
      }
    } catch (err) { // catch any IOs error
      callback(err, null);
    }
  }
}
```
Node.JS

- Anonymous functions?

- Event model:
  - An event source
  - A listener
  - An event object which contains the information about the event
Closure

Context:

Each time a function is declared, a new context is created and variables from parent contexts are duplicated.
```javascript
var sys = require("sys") ;
// context 1
x = 0 ;
function a() {
    // context 2 defined after x = 0;
    // x == 0
    y = x+1; // y == 1
    var b = function(y) {
        // context 3 defined when a is called
        // x == 1
        // y == this.arguments[0] == x == 1 // fist argument
        return y+x ;
    };
    return b(x) ;
}
x = 1 ;
sys.puts(a()); // prints 2
```
var sys = require("sys")
// context 1
x = 0;
function a () {
  // context 2 defined after x = 0;
  // x == 0
  y = x+1; // y == 1
  var b = function ( y ) {
    // context 3 defined when a is called
    // x == 1
    // y == this.arguments[0] == x == 1 // fist argument
    return y+x ;
  }
  return b( x ) ;
}
x = 1 ;
sys.puts( a() ); // prints 2
```javascript
function parse (str) {
    // function which parses a string
    return str.split(",");
}

function readAndParse (file, callback) {
    fs.readFile(file, function (err, data) {
        if (err)
            callback(err, null);
        else
            // return parsed content
            callback(null, parse(data));
    });
}
readAndParse("myfile.txt", function (err, results) {
    if (err) throw err;
    else
        // process results
});
```
- NoSQL database
- Document Oriented
- Data saved as BSON (Binary JSON)
- Retrieved with JSON objects:
  ```javascript
  ex: db.rooms.find({‘roomID’: 84});
  ```
- GridFS: distributed file system
Chat application

- Demo
Chat application

- How does it work?
Stress test

- Benchmarking tools
  - Python
  - Mechanize
  - Multi-mechanize
Stress test

- Test on BC cluster
- 300 seconds
- 5 threads launched every \((300/\text{num\_thread})\) seconds
- 4 clients
- 1 server and 2 servers
Stress test

- Results

Average delay between the expedition of a message by one user and the reception by another user
Stress test

- Results

Throughput for messages
Stress test

- Results

Number of requests:

- 1 server: 174,358
- 2 servers: 320,620
Stress test

- Results

Real average delay between the expedition of a message by one user and the reception by another user.
Stress test

- Results

Real delay at 90% between the expedition of a message by one user and the reception by another user
Thank You