**Php Socket.IO:**

**Time for server receiving client-data:**

**Method:**

Client-side code:

<**script**>**for**(**var *i***=0; ***i***<=220000; ***i***++){  
 **var *socket*** = **io**(**'http://127.0.0.1:3120'**);  
 ***socket***.emit(**'testEvent'**, ***i***);  
 }  
</**script**>

Client-side code is a simple script consisting of a for-loop emitting 220,000 events to server-side.

Server-Side code:

$io = **new** SocketIO(3120);  
$io->on(**'connection'**, **function**($socket)**use**($io){  
 $time\_pre = *microtime*(**true**);  
 $socket->on(**'testEvent'**, **function**($msg)**use**($socket, $time\_pre){  
 *//echo $msg . "\n";* **if**($msg == 220,000){  
 $time\_post = *microtime*(**true**);  
 **echo** ($time\_post - $time\_pre);  
 }  
 });  
});

$time\_pre was defined immediately upon a new client connection.

When $msg is equal to 220,000, time is recorded and the time difference is calculated to obtain the total time consumed to receive these events.

Note: I set the limit to 220,000 because my machine only allocates so many bits for PHP to use for calculations. Anywhere beyond 220,000 would receive the following error:

Fatal error: Allowed memory size of 134217728 bytes exhausted (tried to allocate 50033 bytes)

**Data:**

|  |  |
| --- | --- |
| Trial number | Seconds |
| 1 | 17.236232995987 |
| 2 | 17.676249980927 |
| 3 | 18.23201918602 |
| 4 | 17.616749048233 |
| 5 | 17.373829841614 |

Average time: 17.621 seconds

Time for client receiving server data:

Client-side code:

***socket***.on(**'testEvent'**, **function**(data){  
 **if**(data == 10000){  
 **var** time\_post = Date.now();  
 **console**.log((time\_post - ***time\_pre***) \* 0.001);  
 }  
})

Server-side code:

$io = **new** SocketIO(3120);  
$io->on(**'connection'**, **function**($socket)**use**($io){**for**($i = 0; $i <= 10000; $i++){  
 $socket->emit(**"testEvent"**, $i);  
 }  
 **echo "done\n"**;  
});

similar method was used to measure time cost for data sent from server-side to client-side, except the upper bound of for-loop was much smaller (10,000 in this case), anywhere beyond this causes my browser to freeze, this maybe due to the limitation of my machine.

**Data:**

|  |  |
| --- | --- |
| Trial number | Seconds |
| 1 | 2.031 |
| 2 | 2.051 |
| 3 | 1.981 |
| 4 | 2.004 |
| 5 | 2.03 |

Average time: 2.0194 seconds

**Socket.IO on Javascript**

Client to server data transmission time:

Client-side code:

var socket = io();

for(var i=0; i<=220000; i++){

socket.emit('testEvent', i);

}

Similar to php’s client-side, a for loop emitting 220,000 events is used.

Server-side code:

io.on('connection', function(socket){

var time\_pre = Date.now();

socket.on('testEvent', function(data){

if(data == 220000){

var time\_post = Date.now();

console.log((time\_post - time\_pre) \* 0.001);

}

})

});

**Data:**

|  |  |
| --- | --- |
| Trial number | Seconds |
| 1 | 5.536 |
| 2 | 4.73 |
| 3 | 5.229 |
| 4 | 4.584 |
| 5 | 7.07 |

Average time cost: 5.4298 seconds

Server to client data transmission time cost:

Client-side code:

var time\_pre = Date.now();

socket.on('testEvent', function(data){

if(data == 220000){

var time\_post = Date.now();

console.log((time\_post - time\_pre) \* 0.001);

}

})

Client-side code:

$(function(){

var socket = io();

var time\_pre = Date.now();

socket.on('testEvent', function(data){

if(data == 220000){

var time\_post = Date.now();

console.log((time\_post - time\_pre) \* 0.001);

}

})

})

Surprisingly, Javascript was able to handle much larger data even when receiving events from server side. Hence, it is possible to transmit up to 220,000 events from server-side to clients.

**Data:**

|  |  |
| --- | --- |
| Trial number | Seconds |
| 1 | 7.205 |
| 2 | 6.818 |
| 3 | 4.513 |
| 4 | 5.078 |
| 5 | 4.776 |

Average time cost: 5.6778 seconds

**Conclusion:**

Socket.IO has much more desired performance than php Socket.IO, in terms of server to client, client to server, and quantity of events emitted. I believe one of the main factor is that Javascript has better performance than PHP in general, so it would be reasonable to choose Socket.IO over phpSocket.IO.