Benchmarking Node.js - basic performance tests against Apache + PHP

19 August 2010

(http://zgadzaj.com/sites/zgadzaj.com/files/field/image/horse-race.jpg)

node.js (tag: nodejs) Apache (tag: apache) benchmark (tag: benchmark)

Several days ago I have done few very basic performance tests to see how node.js server behaves compared to Apache when serving very simple pages.

All tests were executed on dual-core Intel T4200 2 GHZ machine with 4 GB RAM running Ubuntu 10.04 Lucid (with X).

For comparison I have used node.js 0.1.103 on one side, and Apache 2.2.14 with prefork MPM and PHP 5.2.10 on the other, hitting them with ApacheBench 2.3 and total of 100,000 request with 1,000 concurrent requests during first test:

```
ab -r -n 100000 -c 1000 <url>
```

and then with total of 1,000,000 requests and 20,000 concurrent requests during the second one:

```
ab -r -n 1000000 -c 20000 <url>
```

Basic "Hello World" node.js server used for testing:

```
1  var sys = require('sys'),
2  http = require('http');
3
4  http.createServer(function(req, res) {
5    res.writeHead(200, {'Content-Type': 'text/html'});
6    res.write('<p>Hello World</p>');
7    res.end();
8  }).listen(8080);
```

gistfile1.js view raw (https://gist.github.com/maciejzgadzaj/538142/raw/376a70691703cecaeb9e5f633abe59ca748a6c32/gistfile1.js) (https://gist.github.com/maciejzgadzaj/538142#file-gistfile1-js) hosted with by GitHub (https://github.com)

and equally basic "Hello World" PHP file for Apache:

```
1  <?php
2  echo '<p>Hello World</p>';
```

gistfile1.php view raw (https://gist.github.com/maciejzgadzaj/538147/raw/0a8ffb135a479c8ff0e45ab00cbd3042f9a2cbf9/gistfile1.php) (https://gist.github.com/maciejzgadzaj/538147#file-gistfile1-php) hosted with by GitHub (https://github.com)

Results

1) Total request: 100,000; concurrency level: 1,000

node.js results:

```
1  Concurrency Level:      1000
2  Time taken for tests:   21.162 seconds
```

```
Complete requests:      100000
Failed requests:        147
   (Connect: 0, Receive: 49, Length: 49, Exceptions: 49)
Write errors:           0
Total transferred:      8096031 bytes
HTML transferred:       1799118 bytes
Requests per second:    4725.43 [#/sec] (mean)
Time per request:       211.621 [ms] (mean)
Time per request:       0.212 [ms] (mean, across all concurrent requests)
Transfer rate:          373.61 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0  135 821.9      0    9003
Processing:     1   40 468.5     25   21003
Waiting:        1   30  64.1     25   12505
Total:          2  175 949.1     26   21003

Percentage of the requests served within a certain time (ms)
  50%     26
  66%     33
  75%     36
  80%     39
  90%     55
  95%     94
  98%   3030
  99%   3090
 100%  21003 (longest request)
```

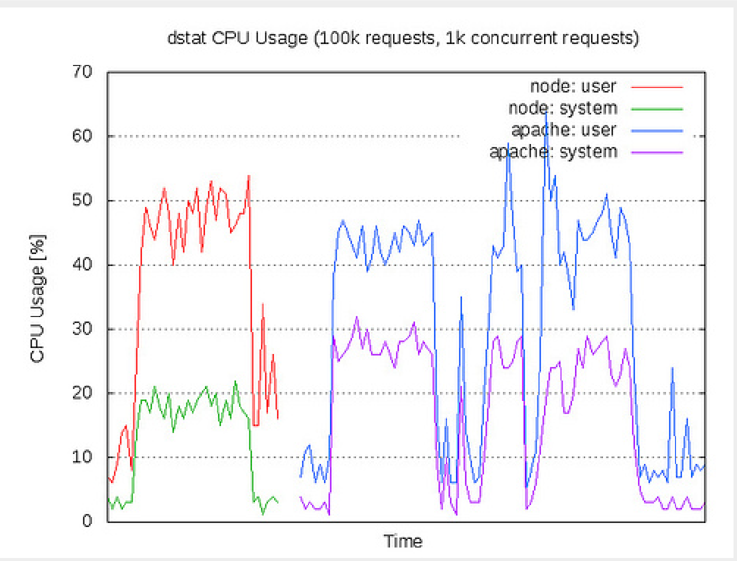
gistfile1.txt view raw (https://gist.github.com/maciejzgadzaj/538164/raw/95c0788cfacdb985a4ac736874a5a330819970d1/gistfile1.txt) (https://gist.github.com/maciejzgadzaj/538164#file-gistfile1-txt) hosted with by GitHub (https://github.com)

Apache results:

```
Concurrency Level:      1000
Time taken for tests:   121.451 seconds
Complete requests:      100000
Failed requests:        879
   (Connect: 0, Receive: 156, Length: 567, Exceptions: 156)
Write errors:           0
Total transferred:      29338635 bytes
HTML transferred:       1889607 bytes
Requests per second:    823.38 [#/sec] (mean)
Time per request:       1214.510 [ms] (mean)
Time per request:       1.215 [ms] (mean, across all concurrent requests)
Transfer rate:          235.91 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0   38 321.8     20    9032
Processing:     0  565 5631.0     51  121380
Waiting:        0  262 2324.1     41   52056
Total:         29  603 5641.7     73  121431

Percentage of the requests served within a certain time (ms)
  50%     73
  66%     78
  75%     82
  80%     83
  90%     89
  95%    105
  98%   4251
  99%  13205
 100%  121431 (longest request)
```

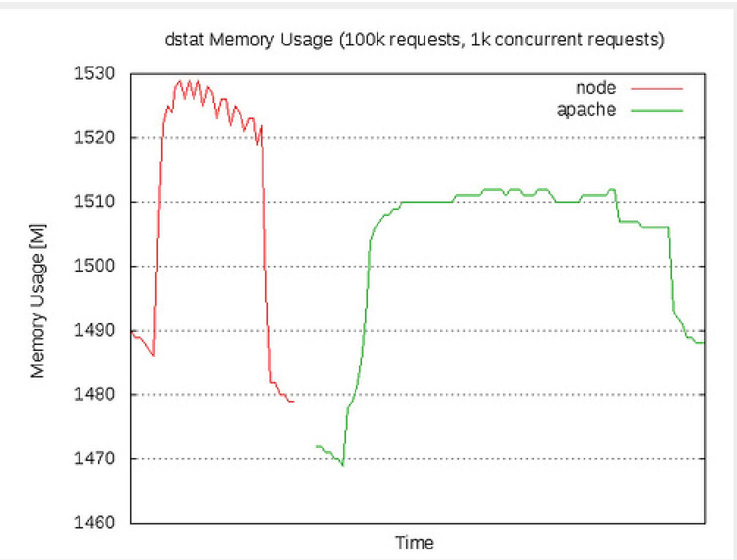
v
//gist.github.com/maciejzgadzaj/5381561/raw/de4c683a0d345567f2687de2704810ea974b914bc/AB%20100k%2010k%20Apache) AB 100k 10k Apache hosted with by GitHub
(https://github.com)

To illustrate CPU and memory load changes I have measured them during tests using dstat (first for node.js, and then Apache right afterwards), with following results:



(http://www.flickr.com/photos/maciejzgadzaj/5891807273/)

CPU Usage: node.js vs Apache/PHP in ApacheBench test - 100k requests, 1k concurrent requests



(http://www.flickr.com/photos/maciejzgadzaj/5892373812/)

Memory Usage: node.js vs Apache/PHP in ApacheBench test - 100k requests, 1k concurrent requests

2) Total requests: 1,000,000; concurrency level: 20,000

noe.js results:

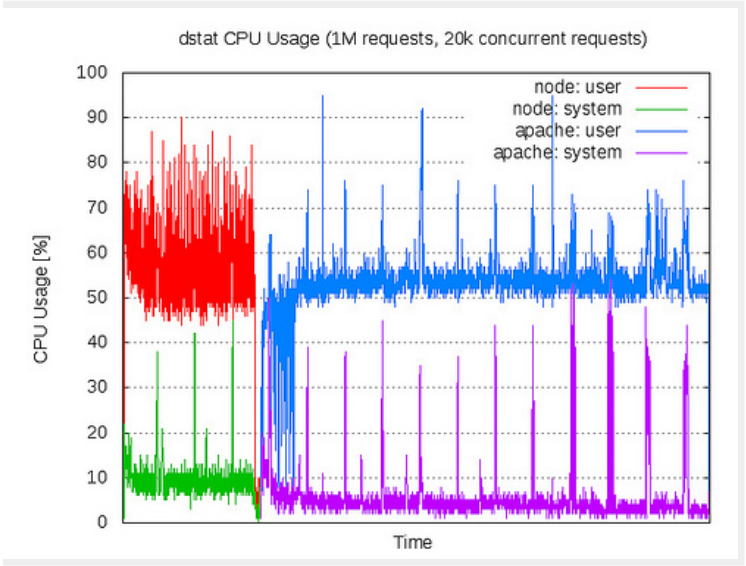
```
Concurrency Level:      20000
Time taken for tests:   1043.076 seconds
Complete requests:      1000000
Failed requests:        25227
   (Connect: 0, Receive: 8409, Length: 8409, Exceptions: 8409)
Write errors:           0
Total transferred:      81265680 bytes
HTML transferred:       18059040 bytes
Requests per second:    958.70 [#/sec] (mean)
Time per request:       20861.529 [ms] (mean)
Time per request:       1.043 [ms] (mean, across all concurrent requests)
Transfer rate:          76.08 [Kbytes/sec] received

Connection Times (ms)
```

```
              min  mean[+/-sd] median   max
Connect:        0 10201 2391.8  10840   20177
Processing:   595 10455 3239.1  10904   39809
Waiting:        0 8323 2331.0   8728   38740
Total:       1181 20656 4758.5  21795   44333

Percentage of the requests served within a certain time (ms)
  50%  21795
  66%  21929
  75%  22047
  80%  22135
  90%  22667
  95%  24252
  98%  24727
  99%  25942
 100%  44333 (longest request)
```


Apache results:

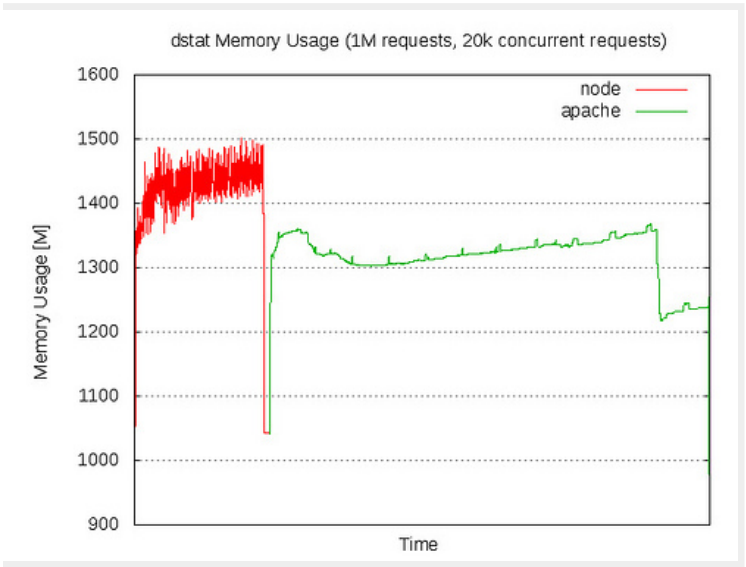
```
Concurrency Level:      20000
Time taken for tests:   3570.753 seconds
Complete requests:      1000000
Failed requests:        2617614
   (Connect: 0, Receive: 848121, Length: 886497, Exceptions: 882996)
Write errors:           0
Total transferred:      36832520 bytes
HTML transferred:       2372264 bytes
Requests per second:    280.05 [#/sec] (mean)
Time per request:       71415.058 [ms] (mean)
Time per request:       3.571 [ms] (mean, across all concurrent requests)
Transfer rate:          10.07 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0 4259 14734.0      0   79497
Processing:     4 64979 51442.2  65543  381910
Waiting:        0 2725 16784.2      0  249108
Total:         87 69238 56233.8  68138  426365

Percentage of the requests served within a certain time (ms)
  50%  68138
  66%  80099
  75%  84390
  80%  85475
  90%  91309
  95%  134983
  98%  303390
  99%  333308
 100%  426365 (longest request)
```


Again, CPU and memory usage comparison:



(http://www.flickr.com/photos/maciejzgadzaj/5891807343/)

CPU Usage: node.js vs Apache/PHP in ApacheBench test - 1M requests, 20k concurrent requests



(http://www.flickr.com/photos/maciejzgadzaj/5892373854/)

Memory Usage: node.js vs Apache/PHP in ApacheBench test - 1M requests, 20k concurrent requests

Conclusions

As the above tests show, node is fast. Really fast. Much faster than Apache - many more requests per second, higher transfer rate with much smaller number of failed requests at the same time. Really shining.

Obviously it is more hungry for system's CPU and memory, but this should not be surprising considering its performance.

What needs to be kept in mind though is that everything really depends on what you want to use Node for. As already said in many places, Node is not something that should replace Apache everywhere.

When considering using Node for a new project (or rewriting an old one) absolutely first thing to do is to make sure that Node really is a good fit for it. Also, depending on project's planned features, used libraries, connections with other systems and/or databases, Node performance could be completely different and not that exciting anymore.

The point is - all those tests above are only the very first point of reference - but to make sure that Node really meets your own specific requirements, you need to test your own system yourself.