Node.js Community Benchmarking Efforts

Gareth Ellis

Node.js Live London 2016

About Gareth Ellis

Runtime Performance Analyst @ IBM

Looking at Performance since 2012
Originally solely Java
Started on Node 2015
Member of benchmarking workgroup

Contact me:
gareth.ellis@uk.ibm.com
github.com/gareth-ellis

Agenda

- **Introduction to benchmarking**
  - Key challenges
  - Approaches
  - Identifying source of regression
- **Benchmarking Node.js**
  - Tools
  - Examples
- **Node benchmarking workgroup**
  - Use cases
  - Current benchmarks
  - Results/Graphs
  - How to get Involved

Introduction to benchmarking

- **Change one thing and one thing only between runs**
  - Application code / benchmark
  - Runtime
  - Machine
  - An NPM
- **Performance testing is quite different to functional testing**

Key Challenges

- **Fundamental run-to-run Variance**
  - False positives
  - Collecting enough samples to be sure of the result
  - Documenting Expected Variance
- **Consistent Environment**
  - Known starting machine state
  - Machine Isolation
  - Interleave comparison runs
- **Jumping to Conclusions**

Approaches

- **Micro-Benchmarks**
  - Measure a specific function/API
    - Ex: Buffer.new()
  - Compare key characteristics
  - Micro-benchmark improvements may not mean real world improvements
  - Risk of not measuring exactly what you expect – especially where a JIT is involved
- **Whole System**
  - Benchmark expected customer use case
    - Ex: AcmeAir - http://github.com/acmeair/acmeair-nodejs
  - The more you test, the more chance for variance

I've found a regression, now what?

- **Are you sure ?**
  - Revalidate environment,
  - Expected variance
- **If so, what changed ?**
  - Your application
  - Node.js
  - Your environment
- **Compare between good/bad cases**
  - Tools
  - Binary search

Benchmarking Node.js

- **Sources of regressions**
  - Node.js
    - lib/*.js – buffer, cluster, etc
  - V8
  - OpenSSL
  - libuv
  - NPM Module
- **Tools**
  - Javascript profiler
    - V8 profiler
    - Appmetrics
  - Native profiler (ex perf, tprof, oprofile)

Example – Microbench

```
var harness = require('../../common/harness.js');

var ARRAY = [1, 2, 23829, 4, 5, 7, 12312321, 2131, 434832, 43792, 23421, 65345, 132210,
77777, 322131, 1, 2, 23829, 4, 5, 7, 12312321, 2131, 434832, 43792, 23421, 65345, 132210,
77777, 322131, 1, 2, 23829, 4, 5, 7, 12312321, 2131, 434832, 43792, 23421, 65345, 132210,
77777, 322131, 1, 2, 23829, 4, 5, 7, 12312321, 2131, 434832, 43792, 23421, 65345, 132210,
77777, 322131];
var ITERATIONS = 300000;
var result;

function test() {
	for(var i=0;i<ITERATIONS;i++) {
		result = new Buffer(ARRAY);
	}
}

harness.run_test(test);
```

Example – Microbench

Node 4.3.2:

```
./node benchmark.js
total time:5.079s / iterations:54 / ops/sec:10.63 / average time:0.09s / variance:0.89%
total time:5.076s / iterations:54 / ops/sec:10.64 / average time:0.09s / variance:0.75%
```

Node 4.4.0:

```
./node benchmark.js
total time:5.131s / iterations:31 / ops/sec:6.04 / average time:0.17s / variance: 2.32%
total time:5.106s / iterations:31 / ops/sec:6.07 / average time:0.16s / variance: 0.28%
```

= ~ 40% regression

V8 Profiler

- Part of Node.js binary
- Turn on with
  - --prof
- Test-tick-process to post-process
    - ./node --prof-process isolate-0x2818130-v8.log
- Other helper modules like
  - https://www.npmjs.com/package/v8-profiler

Appmetrics

- npm install appmetrics
- Can provide cpu, gc, memory, profiling + lots more
- Connect into IBM healthcenter for remote monitoring
- https://www.npmjs.com/package/appmetrics

```
var appmetrics = require('appmetrics');
var monitoring = appmetrics.monitor();
monitoring.on('initialized', function (env) {
                env = monitoring.getEnvironment();
        for (var entry in env) {
                console.log(entry + ':' + env[entry]);
        };
});
monitoring.on('cpu', function (cpu) {
        console.log('[' + new Date(cpu.time) + '] CPU: ' + cpu.process);
});
```

Binary chop

- Compare changes between good & bad
- git rev-list ^good bad^
- Bit more time consuming for where there’s a lot of change sets
- git bisect is quite useful though

Example – v8 profiler (--prof)

```
<    5585   23.7%   23.9%  LazyCompile: *fromObject buffer.js:121:20
<    1308    5.6%    5.6%  LazyCompile: *subarray native typedarray.js:165:28
<    1263    5.4%    5.4%  LazyCompile: *Uint8ArrayConstructByArrayBuffer native typedarray.js:35:42
<     964    4.1%    4.1%  Builtin: JSConstructStubGeneric
<     854    3.6%    3.7%  Stub: InstanceofStub
<     677    2.9%    2.9%  LazyCompile: *test benchmark.js:7:14
<     669    2.8%    2.9%  LazyCompile: *Uint8Array native typedarray.js:122:31
---
>   15227   47.1%   47.3%  LazyCompile: *fromObject buffer.js:121:20
>    1240    3.8%    3.9%  LazyCompile: *subarray native typedarray.js:165:28
>    1166    3.6%    3.6%  LazyCompile: *Uint8ArrayConstructByArrayBuffer native typedarray.js:35:42
>     967    3.0%    3.0%  Builtin: JSConstructStubGeneric
>     802    2.5%    2.5%  Stub: InstanceofStub
>     780    2.4%    2.4%  LazyCompile: *test benchmark.js:7:14
>     654    2.0%    2.0%  LazyCompile: *Uint8Array native typedarray.js:122:31
```

Example - Perf (system based profiler)

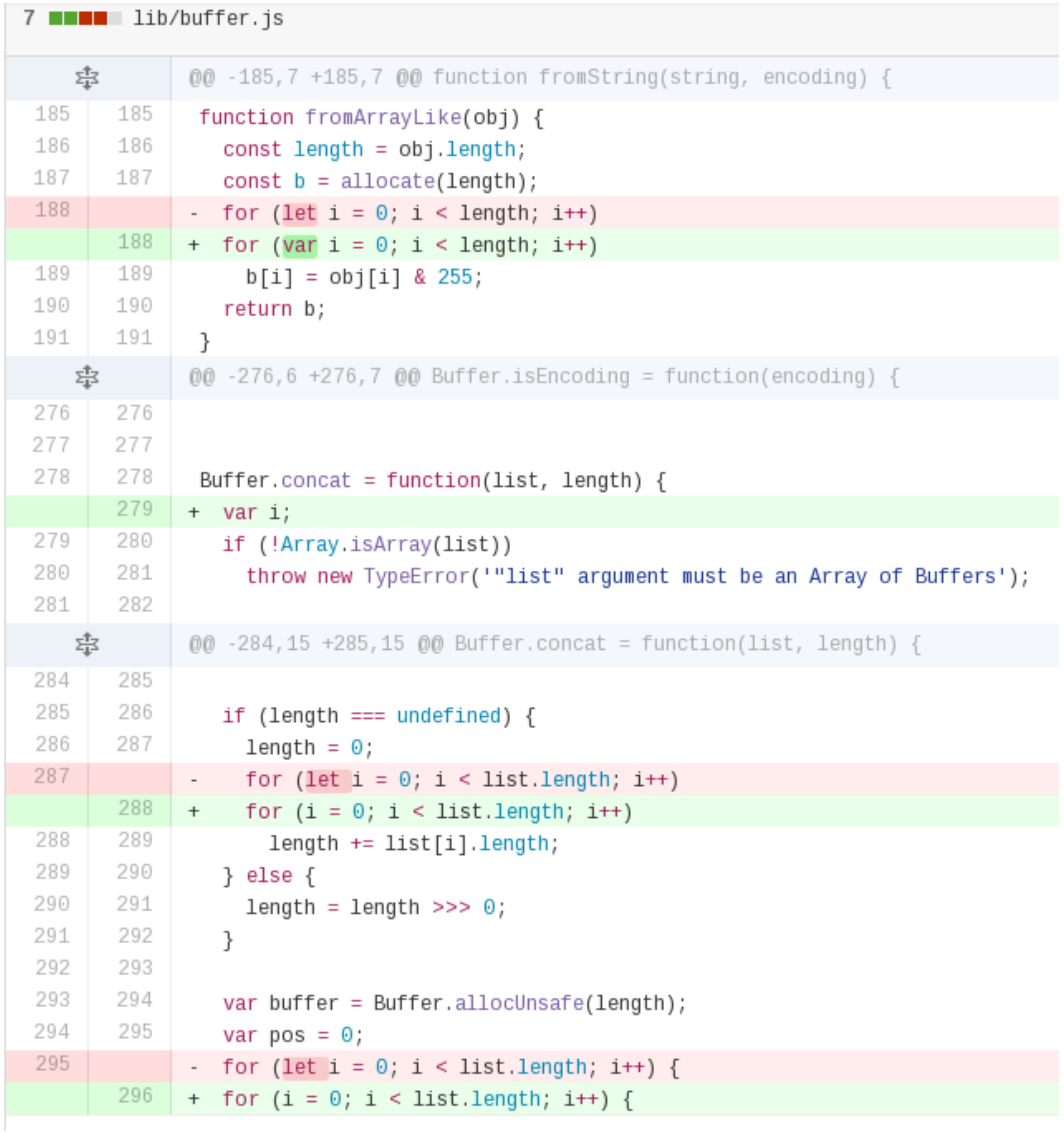
- perf record -i -g -e cycles:u -- ./node --perf-basic-prof benchmark.js
- perf report

```
diff perf_good.out perf_bad.out

327c302
<     91.52%    23.43%  node     perf-16993.map       [.] LazyCompile:*fromObject buffer.js:121
---
>     93.25%    46.56%  node     perf-16934.map       [.] LazyCompile:*fromObject buffer.js:121
331c306
```

--trace-opt --trace-deopt

```
[marking 0x39570dd44951 <JS Function fromObject (SharedFunctionInfo 0x39570dd12f91)> for
recompilation, reason: not much type info but very hot, ICs with typeinfo: 14/64 (21%), generic ICs: 0/64
(0%)]

[compiling method 0x39570dd44951 <JS Function fromObject (SharedFunctionInfo
0x39570dd12f91)> using Crankshaft]

[optimizing 0x39570dd44951 <JS Function fromObject (SharedFunctionInfo 0x39570dd12f91)> - took
0.315, 1.339, 0.511 ms]

[completed optimizing 0x39570dd44951 <JS Function fromObject (SharedFunctionInfo
0x39570dd12f91)>]

[deoptimizing (DEOPT eager): begin 0x39570dd44951 <JS Function fromObject (SharedFunctionInfo
0x39570dd12f91)>

[deoptimizing (eager): end 0x39570dd44951 <JS Function fromObject (SharedFunctionInfo
0x39570dd12f91)>
```

The result



```
7 ■■■■■ lib/buffer.js
```

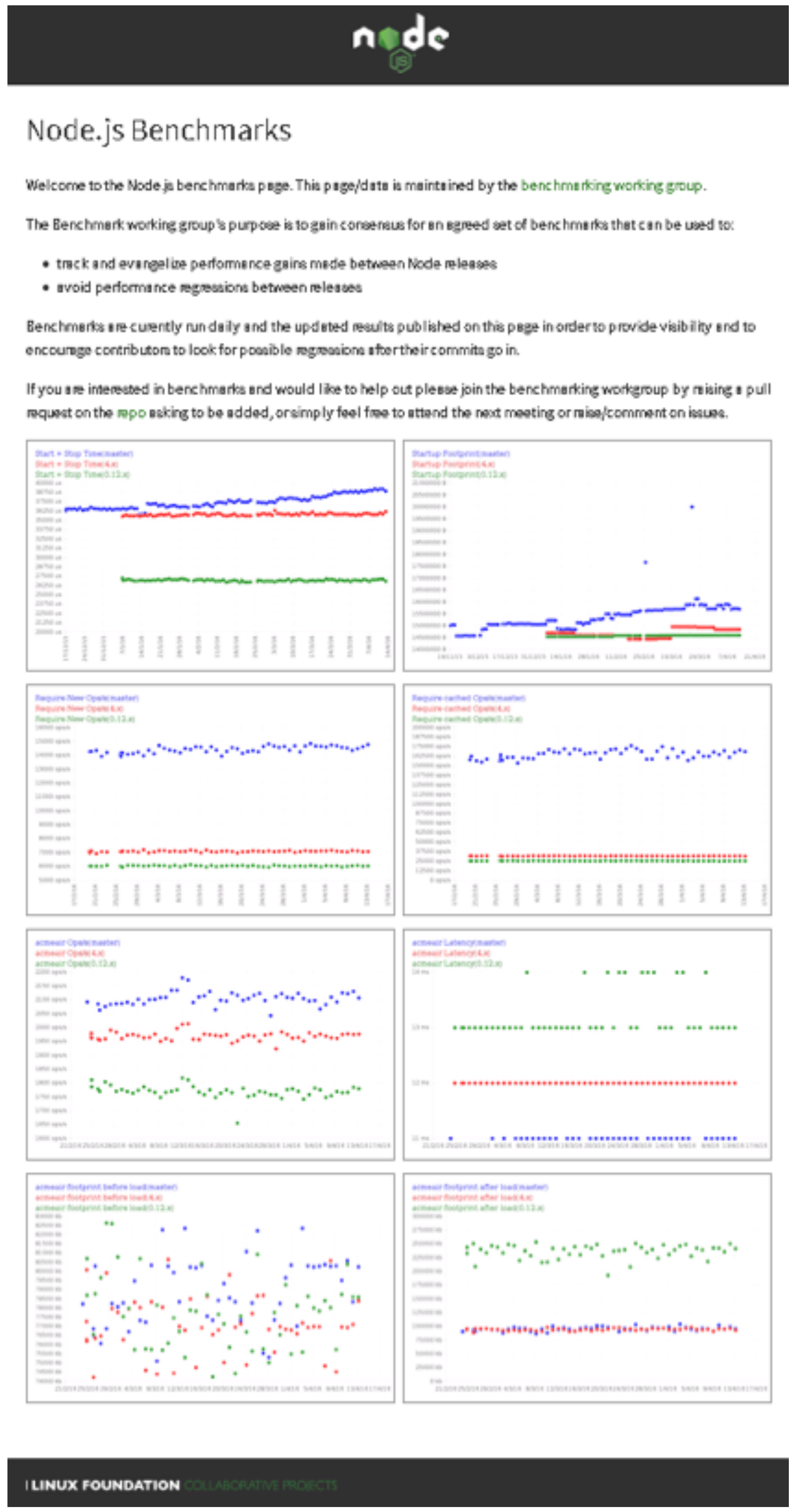
```diff
@@ -185,7 +185,7 @@ function fromString(string, encoding) {
 function fromArrayLike(obj) {
   const length = obj.length;
   const b = allocate(length);
-  for (let i = 0; i < length; i++)
+  for (var i = 0; i < length; i++)
     b[i] = obj[i] & 255;
   return b;
 }
@@ -276,6 +276,7 @@ Buffer.isEncoding = function(encoding) {


 Buffer.concat = function(list, length) {
+  var i;
   if (!Array.isArray(list))
     throw new TypeError('"list" argument must be an Array of Buffers');

@@ -284,15 +285,15 @@ Buffer.concat = function(list, length) {

   if (length === undefined) {
     length = 0;
-    for (let i = 0; i < list.length; i++)
+    for (i = 0; i < list.length; i++)
       length += list[i].length;
   } else {
     length = length >>> 0;
   }

   var buffer = Buffer.allocUnsafe(length);
   var pos = 0;
-  for (let i = 0; i < list.length; i++) {
+  for (i = 0; i < list.length; i++) {
```

- Issue in v8 optimiser.
- Will be fixed there once TurboFan becomes default
- Until then…
- github.com/nodejs/node/pull/5819

Node.js Benchmarking Workgroup

- Mandate to track and evangelize performance gains between node releases
- Key goals
  - Define Use Cases
  - Identify Benchmarks
  - Run/Capture results
- 13 current members
- Meetings every month or so

http://github.com/nodejs/benchmarking
http://benchmarking.nodejs.org



Node.js Benchmarks

Welcome to the Node.js benchmarks page. This page/data is maintained by the benchmarking working group.

The Benchmark working group's purpose is to gain consensus for an agreed set of benchmarks that can be used to:

- track and evangelize performance gains made between Node releases
- avoid performance regressions between releases

Benchmarks are curently run daily and the updated results published on this page in order to provide visibility and to encourage contributors to look for possible regressions after their commits go in.

If you are interested in benchmarks and would like to help out please join the benchmarking workgroup by raising a pull request on the repo asking to be added, or simply feel free to attend the next meeting or raise/comment on issues.

Start + Stop Time(master)
Start + Stop Time(4.x)
Start + Stop Time(0.12.x)
Startup Footprint(master)
Startup Footprint(4.x)
Startup Footprint(0.12.x)

LINUX FOUNDATION COLLABORATIVE PROJECTS

Benchmarking Use cases

- Back-end API services
- Service oriented architectures (SOA)
- Microservice-based applications
- Generating/serving dynamic web page content
- Single Page Applications with bidirectional communication over WebSockets and/or HTTP/2
- Agents and Data Collectors
- Small scripts

https://github.com/nodejs/benchmarking/blob/master/docs/use_cases.md

Benchmarking Use cases

- For these use cases one or more of the following are often important:
  - consistent low latency
  - ability to support high concurrency
  - throughput
  - fast startup/restart/shutdown
  - low resource usage (memory/cpu)

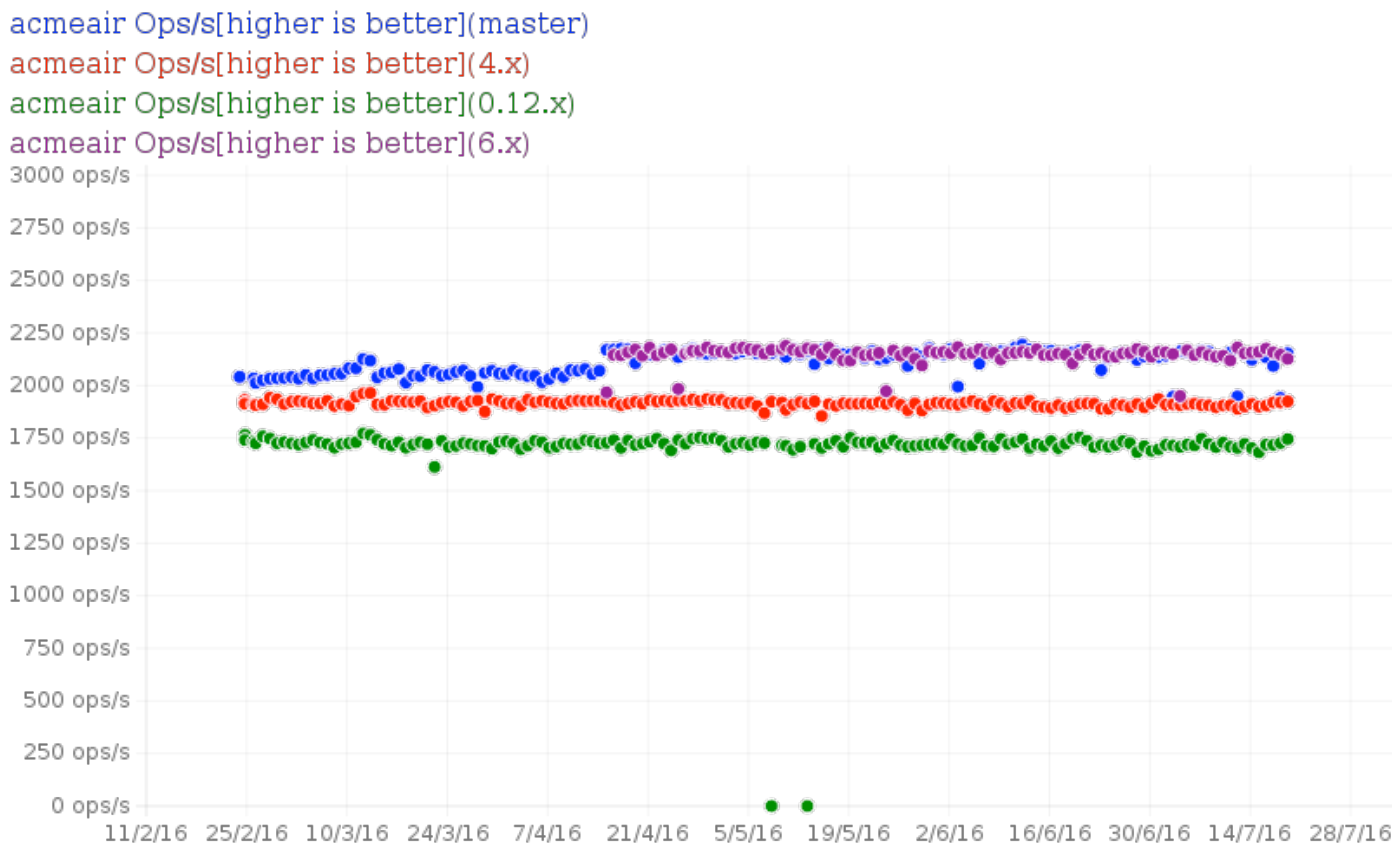
https://github.com/nodejs/benchmarking/blob/master/docs/use_cases.md

Benchmarks – Progress so far...

- Currently Running
  - Startup time
  - Footprint
  - Time to 'require' a module
  - AcmeAir – throughput, response time, footprint measurements
  - Docker

- In progress
  - URL performance
  - Additions from Node.js benchmarks directory

Charts


acmeair Ops/s[higher is better](master)
acmeair Ops/s[higher is better](4.x)
acmeair Ops/s[higher is better](0.12.x)
acmeair Ops/s[higher is better](6.x)
3000 ops/s
2750 ops/s
2500 ops/s
2250 ops/s
2000 ops/s
1750 ops/s
1500 ops/s
1250 ops/s
1000 ops/s
750 ops/s
500 ops/s
250 ops/s
0 ops/s
11/2/16
25/2/16
10/3/16
24/3/16
7/4/16
21/4/16
5/5/16
19/5/16
2/6/16
16/6/16
30/6/16
14/7/16
28/7/16

How to get involved

- http://www.github.com/nodejs/benchmarking
- Take a look at what we're running – and the areas we're looking to get benchmarks to cover
- Something missing?
- Something you don't think is quite right?
- Open an issue!