Why are native ES6 promises slower and more memory-intensive than bluebird?

In this benchmark, the suite takes 4 times longer to complete with ES6 promises compared to Bluebird promises, and uses 3.6 times as much memory.

How can a JavaScript library be so much faster and lighter than v8's native implementation written in C? Bluebird promises have exactly the same API as native ES6 promises (plus a bunch of extra utility methods).

Is the native implementation just badly written, or is there some other aspect to this that I'm missing?

javascript performance io.js

edited Apr 10 '15 at 20:18

asked Apr 10 '15 at 20:12

callum
2,921 6 17 22

Keep in mind that modern JavaScript implementations are heavily optimized, and may even run natively using JIT. – Snowman Apr 10 '15 at 20:28

1 Answer

Bluebird author here.

V8 promises implementation is written in JavaScript not C. All JavaScript (including V8's own) is compiled to native code. Additionally user written JavaScript is optimized, if possible (and worth it), before compiled to native code. Promises implementation is something that would not benefit much or at all from being written in C, in fact it would only make it slower because all you are doing is manipulating JavaScript objects and communication.

The V8 implementation simply isn't as optimized as bluebird, it for instances allocates arrays for promises' handlers. This takes a lot of memory when each promise also has to allocate a couple of arrays (The benchmark creates overall 80k promises so that's 160k unused arrays allocated). In reality 99.99% of use cases never branch a promise more than once so optimizing for this common case gains huge memory usage improvements.

Even if V8 implemented the same optimizations as bluebird, it would still be hindered by specification. The benchmark has to use `new Promise` (an anti-pattern in bluebird) as there is no other way to create a root promise in ES6. `new Promise` is an extremely slow way of creating a promise, first the executor function allocates a closure, secondly it is passed 2 separate closures as arguments. That's 3 closures allocated per promise but a closure is already a more expensive object than an optimized promise.

Bluebird can use `promisify` which enables lots of optimizations and is a much more convenient way of consuming callback APIs and it enables conversion of whole modules into promise based modules in one line (`promisifyAll(require('redis'));`).

edited Dec 14 '16 at 22:53

Benjamin Toueg
103 5

answered Apr 14 '15 at 7:22

Esailija
4,192 1 12 16

8 "still be hindered by specification" - Not sure what that means. Are you saying that ES6 is following a spec that is inherently slow, and if that's the case, does that mean bluebird is not following the same spec (and if that's the case, is it following a different one, and which one)? And is there any reason why ES6 couldn't have a better way of creating a root Promise besides `new Promise` or improve the instantiation to make it less expensive (like not creating 3 closures per instance)? – Anthony Jun 3 '15 at 12:40

6 That doesn't sound good at all (for JS). I really don't want to use a Promise library when there's an internal

implementation. This is a more than unfortunate situation for everyone if this is all true. But I already have trouble seeing the Promise-hype anyway, I've written 100,000 LoC JS apps and I still don't see any real need for this, it's a very minor improvement if at all *to me*, mostly in error handling, no improvement in callback handling (I've never been in "callback hell" with my coding style). – Mörre Jun 15 '15 at 8:44

13 In ES6, can't you use `Promise.resolve()` to create a "root promise"? – zetlen Nov 13 '15 at 16:02

7 @MörreNoseshine (continued) Years later, the ES6 authors came along and said "hey, let's specify that JS engines must provide a generic Promises/A+ conforming utility out-of-the-box, so people always have a basic promise tool to hand". This is a nice convenience (not having to import a library just to do a quick `Promise.resolve()` or whatever), but it's a very basic implementation, and its existence should not put you off using more serious promise-related tools like bluebird! – callum Jan 21 '16 at 12:39

7 @MörreNoseshine 100k LOC Javascript app that probably never had any async functionality. Good luck writing a 100k LoC JS game with a mysql / redis library without bluebird. – NiCk Newman Mar 7 '16 at 15:26