Node.js needs to die in a fire and Golang is the perfect arsonist

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Find file Clone or download

alexhultman committed on GitHub Update README.md Latest commit 02ad711 a day ago

README.md Update README.md a day ago

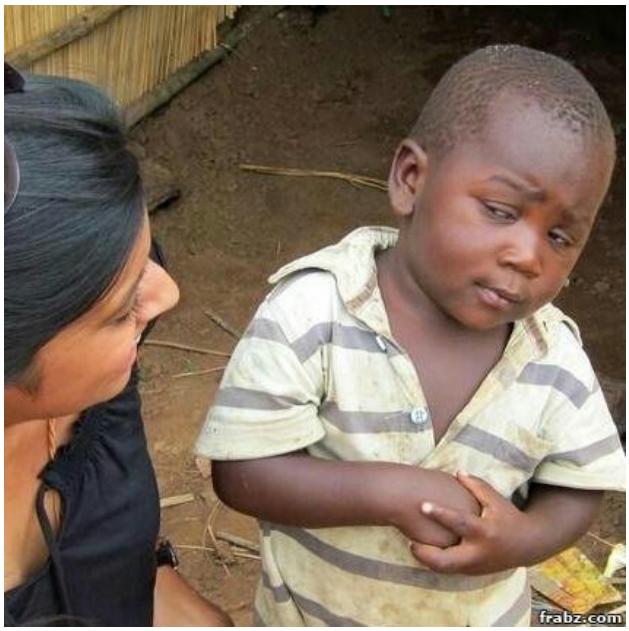
README.md

Node.js needs to die in a fire and Golang is the perfect arsonist

As a developer with 10+ years of C & C++ experience in various fields, my first face-to-face encounter with a Node.js web developer back in 2014 was probably the biggest cultural clash of my life. I had never used PHP, JavaScript, Python or any other scripting language before, at least not for more than a couple of lines here and there. In fact, I was not even fully aware of the dynamic any-typed nature of scripting languages.

This Nodester, my boss, was the complete opposite. He was a die-hard JavaScript fanatic, completely unaware of anything non-script. When I asked "what type is this variable?" to understand what a function did, he looked at me as if I was mentally challenged. I received plenty of preaching of how *extremely* fast and efficient Node.js was and how slow and useless Apache was. You see Node.js developers love to constantly mention Apache (designed in 1993, runs PHP) when talking about how fast Node.js is.

My boss was your typical pez-dispenser of Node.js marketing buzz. "JavaScript is assembly", "async networking is brand new" and the whole classic verbatim mantra. I especially remember how he would claim that certain 1970's C syntax was "taken from JavaScript". He never benchmarked anything, which goes without saying.



I quit my job after a year, only to meet with another identical Node.js zombie at the new job. Repeating the same old garbage nonsense over and over, being completely certain that Node.js would "outperform any server in the world". Same brainwashed soldier following Node.js headquarter marketing instructions. Of course, anything he wrote in Node.js would clog up and bottleneck our entire platform. He managed to add a 20 second overhead to my 8 second OpenCL calculation. He *moved data* with Node.js *slower than what we computed it*.

Node.js is king of the crippled

It is true that Node.js is extremely performant if you compare it with other, highly crippled solutions like PHP or Ruby. However, Ruby on Rails performs so bad that your Intel Xeon becomes as useful as a 99 cent Walmart calculator using it, so comparison doesn't say anything. Node.js certainly takes gold medal in the *special* olympics, no doubt about it.

I've noticed that golang is becoming popular in back-end and people generally agree that it does outperform Node.js. Some claim the gain is "limited because of networking bottlenecks". Well if your network is a bottleneck then your set-up is bad. Either increase bandwidth or decrease CPU spendings and save money. A faster server does not magically replace your networking infrastructure but it *does* decrease uptime cost.

I've done *many* benchmarks of servers, but this time I wanted to test things over a real physical network to see for myself what kind of gain golang can bring. Here are the results (and yes, all servers were at 100% CPU time in one single thread):

- Ruby on Rails could serve ~200 requests/second @ 1 thread.
- Node.js could serve 17k requests/second @ 1 thread.
- Golang with fasthttp could serve 122k requests/second @ 1 thread.
- Node.js with Express could serve 7k requests/second @ 1 thread.

- Golang with standard net/http could serve 36k requests/second @ 1 thread.
- µWS could serve 220k requests/second @ 1 thread.

Remember, these results are over an *actual ethernet cable*. Whatever software used, it and only it was responsible for this major performance spectra. We are talking about 500x - 1000x the networking performance depending on software used.

Now, µWS is more of a theoretical limit rather than an actual server you can use. The fact that golang performs at half this limit is incredible given that golang is actually starting to become widespread within the back-end industry. Even though I'm a C++:er at heart, I cannot stop being incredibly pleased with this evolution. If anything, golang is a major leap in the right direction.