Optimize for the Developer's Time, Not the Machine's

W. Adam Koszek

Koszek ORG

wojciech@koszek.com

2022-10-05T17:46:54Z

It is sometimes easy to forget how computer engineers in the early days struggled just to make things work. In '70s and '80s, computing resources like memory were scarce; CPUs were slow, and disk space was limited. There was little or no networking, and communication with the computer was primitive. Heck, even Norton Commander on a 12" was suboptimal. Right now, I could probably achieve this:

And do it in a single Terminal window, which would make '70s engineers' eyeballs go red with jealousy. Have you ever wondered what the world was like without Vim or Emacs? Yet these are fairly recent accomplishments. To summarize: It's easy to forget all these things and keep asking questions such as *why are Microsoft Office file formats so complicated*, unless you understand the heritage of certain decisions and computers in general.

By now, I would expect system architects and designers to understand the quick decay of technology and the relative absence of "wall clock" time. I'd expect modern technologies, such as the Swift, Java, or Go languages, to be optimized for my work time. This includes not only how fast programs are written in these technologies but also how fast I can write such programs, how friendly the developer's environment is, and whether it stands in my way of achieving goals.

New Horizons' space mission is powered by the 12 MHz CPU, but desktop and mobile solutions aren't space ventures. In 2015, I'd expect most of these resource limitations to be abandoned since we moved on from slow to FAST systems. Computers sped up, and we no longer have to worry about 1 MB here or there. Each time I call Python on my laptop, each time I make an empty object in Objective C running on my iPhone, I'm probably making my CPU loop around numbers of instructions that would make New Horizons' software engineers cry. I'm not saying nor suggesting that writing sloppy code that wastes resources is the way to go. I'm instead pointing out that, if the resource is there and waiting

to be used, I'd expect it to be deployed as quickly as possible. Otherwise, they're wasting resources in a different sense.

But the world is much better now not only because computers are very fast. We also have the Internet, and I see little advancement in how the Internet is being used for writing software. One would expect that by now I could ask my development IDE environment to give me the functions that I need dynamically. And if there aren't any, it should query the Internet to search for the ones I need. Most of us do it already by querying *StackOverflow*. We just need to make it integrated.

Another example is disk space. The rate of increase of available disk space is dramatic, even assuming that we're talking about fast SSD storage. If we compare my current 256 GB SSD disk to my very first 1.2 GB disk ever, it's about 200x more. Yet most of the time my system lets it sit and do little, maybe with occasional swapping:

In 2015, I honestly want programs to take the maximum amount of memory whenever possible, as long as it makes my computer work faster. I want them to steal disk space and bandwidth and cycles, since I have a lot of those too. While I do understand that some cost of time must be spent on getting things to work, I'd expect us to be solving problems that are meaningful, not cumbersome. The resource I have very little of is time. Thus, I'd expect us all in the engineering field to optimize for time. Not asymptotic or algorithmic time, but my time, to be precise. Regardless of how much you charge per hour, your and my time, compared to our computer's time, is VERY expensive.