The unreasonable effectiveness of the Julia programming language

reader comments

197 with 108 posters participating

Share this story

[embedded content]
Ain’t no party like a programming language virtual conference party
I’ve been running into a lot of happy and excited scientists lately. “Running into” in the virtual sense, of course, as conferences and other opportunities to collide with scientists in meatspace have been all but eliminated. Most scientists believe in the germ theory of disease.

Anyway, these scientists and mathematicians are excited about a new tool. It’s not a new particle accelerator nor a supercomputer. Instead, this exciting new tool for scientific research is… a computer language.

How can a computer language be exciting, you ask? Surely, some are better than others, depending on your purposes and priorities. Some run faster, while others are quicker and easier to develop in. Some have a larger ecosystem, allowing you to borrow battle-tested code from a library and do less of the work yourself. Some are well-suited to particular types of problems, while others are good at being general-purpose.

For scientists who compute, languages, the quality of compilers and libraries, and, of course, the machines they run on, have always been important. For those whose job it is to simulate the atmosphere, or design nuclear weapons, Fortran

was the traditional tool of choice (and still often is, although it has more competition now). That language has dominated the market because compilers are available that can take good advantage of the largest supercomputers. For the current breed of data scientists, Python is currently popular because of the momentum of its ecosystem and its interactivity and rapid development cycle.

Six years ago, I wrote in these pages about the enduring prominence of Fortran for scientific computing and compared it with several other languages. I ended that article with a prediction that, in 10 years, a new language called Julia stood a good chance of becoming the one that scientists would turn to when tackling large-scale numerical problems. My prediction was not very accurate, though.

It actually only took Julia about *half* that time.

Enough excitement for a Con

Talking with scientists in recent years, the computer language Julia has genuinely created a new wave of enthusiasm in the industry. But back when I wrote about its potential, I did not understand *why* the language would take off.

I based my assessment on Julia's unique combination of convenient syntax with uncompromising performance. At the time, although Julia was still in pre-1.0 status, there was already plenty of excited chatter. Julia seemed to have solved the "two-language problem"—a conundrum often facing Python programmers, as well as users of other expressive, interpreted languages. You write a program to solve a problem in Python, enjoying its pleasant syntax and interactivity. The program works on a test version of your problem, but when you try to scale it up to something more realistic, it's too slow. This is not your fault. Python is inherently slow—something that doesn't matter for some types of applications but does matter for your big simulation. After applying various techniques to

speed it up but only realizing modest gains, you finally resort to rewriting the most time-consuming parts of the calculation in C (most commonly). Now it's fast enough, but now you also need to maintain code in both languages, hence the two-language problem.

Although Julia's solution to this problem attracted scientists and others to the language, this is not the reason for the newfound excitement around the platform. There is something else.



Enlarge / The banner for JuliaCon 2020

While I was working on this article, this year's JuliaCon, the annual Julia convention, took place (online, of course). Usually the schedule for a computer meeting is filled with titles about things related to programming, compilers, algorithms, optimization, and other computer sciencey subjects. And while there was plenty of that at this year's Julia meetup, skimming through the titles leaves the impression that one has stumbled into a science conference. There are presentations on everything from fluid dynamics to brain imaging to language processing. Despite the stunning variety of fields, however, watching the presentations gives a sense of community around a shared attitude that seems to have been influenced by the free software movement.

Everyone's code is on GitHub. If you are interested in using someone's algorithm in your research, you can read the source, and you will have access to the latest version as it is developed. Scientists of a certain age will know how vastly

different this is from how computational research used to proceed. In the old days, code rarely left the lab.

The Julia community is unified by something else, as well: a shared delight in the magical (this word cropped up more than once) power of Julia to facilitate collaboration and code reuse. Consider just some of the praise coming from JuliaCon 2020 presenters:

> *That's one of the things that makes Julia so powerful in the solution of these problems […] This integration gives Julia an advantage over other languages […] we have been able to develop these solutions in a very short period of time:*
>
> *León Alday, molecular modeling*
>
> *Julia is really the language that allows such a project to exist:*
>
> *George Datseris, Dr. Watson, a scientific assistant*
>
> *Julia is a joy to program in:*
>
> *Mauro Werder, Glacier ice thickness*
>
> *The Julia language […] is a particularly agile tool:*
>
> *Valeri Vasquez, Disease vector dynamics*
>
> *Julia was the obvious choice:*
>
> *Rafael Schouten, Spatial simulations*
>
> *[Julia allows] me to harness tools from across disciplines to advance cancer research:*

Meghan Ferrall-Fairbanks, Tumor dynamics

This work has been very nice to do in Julia because of the nice abstractions that allow very general code:

Vilim Štih, Zebrafish brain dynamics

It is really nice to have a fast language that can be used to write everything. […] but what really impresses me these days is something else—Julia is somehow able to increase my productivity […]. Julia makes it easy to think at the right level of abstraction.":

Petr Krysl, Partial differential equations

Why doing science in Julia is awesome […] Inter-package interaction = pure magic!:

George Datseris Analysis of music performance

These scientists have all discovered that Julia boosts the opportunities for collaboration and makes it easier than ever before to incorporate the work of others and to allow them to write code that can be used *by* others in unforeseen ways. The key to these powers is in Julia's solution to a different old conundrum, this time from computer science—the *expression problem*.

https://arstechnica.com/?p=1709834