Julia: A High-Level Language for Supercomputing and Big Data

September 6, 2017 by Daniel Gutierrez 1 Comment

Sponsored Post

The key software stack used for high-performance computing (HPC) workloads has typically been a statically compiled language such as C/C++ or Fortran, in conjunction with OpenMP/MPI. This environment has stood the test of time and is almost ubiquitous in the HPC world. The key reason for this is its efficiency in terms of the ability to use all available compute resources while limiting memory usage. This level of efficiency has always been beyond the scope of more "high-level" scripting languages such as Python, Matlab/Octave, or R. This is primarily because such languages are designed to be high-productivity environments that facilitate rapid prototyping—and are not designed to run efficiently on large compute clusters. However, there is no technical reason why this should be the case, and this represents a tooling problem for scientific computing.

Julia is a new language for technical computing that is meant to address this problem. It reads like Python or Octave, but performs as well as C. It has built-in primitives for multi-threading and distributed computing, allowing applications to scale to millions of cores. In addition to HPC, Julia is also gaining traction in the data science community.

The first natural questions that spring to mind are: Why can't the other high-level languages perform this well? What's unique about Julia that lets it achieve this level of efficiency while still allowing the user write readable code? The answer is that Julia uses a combination of type inference and code specialization to generate tight machine code.

Julia Computing was formed to consult with industry and fosters adoption of Julia in the community. One of Julia Computing's flagship products is JuliaPro, a Julia distribution that includes an IDE, a debugger, and more than 100 tested packages. JuliaPro will soon ship with Intel® Math Kernel Library (Intel® MKL) for accelerated BLAS operations and optimizations for multi-core and the latest Intel® processors.

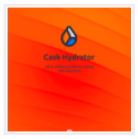
Julia has several built-in primitives for parallel computing at every level: vectorization (SIMD), multi-threading, and distributed computing. An interesting use-case would be having different processes solve, for example, linear equations and then serialize their output and send it to the master process. Scalable machine learning is one such example that involves many independent backsolves. The data visualization below shows the performance of RecSys.jl, which processes millions of movie ratings to make predictions. It is based on an algorithm called Alternating Least Squares (ALS), which is a simple, iterative method for collaborative filtering. Since each solve is independent of every other, this code can be

INDUSTRY PERSPECTIVES

Swing and a Miss - How to Avoid Striking Out with the Wrong Data

In this special guest feature, Mark Budzinski, CEO at WhereScape, discusses how baseball can teach valuable data management lessons for enterprise by outlining how to get the right mix of automation and intelligent human analytical input. Context with any data set is just as critical to the result as is the collection of such data. [Read More...]

WHITE PAPERS



Cask Hydrator: Self Service Data Lakes on Hadoop

The Cask Hydrator empowers organizations to blend, transform, and analyze data across multiple sources to create an on-demand analytical view across key customer interactions. This provides powerful insights into buyers, brand performance, products, and services.

Download

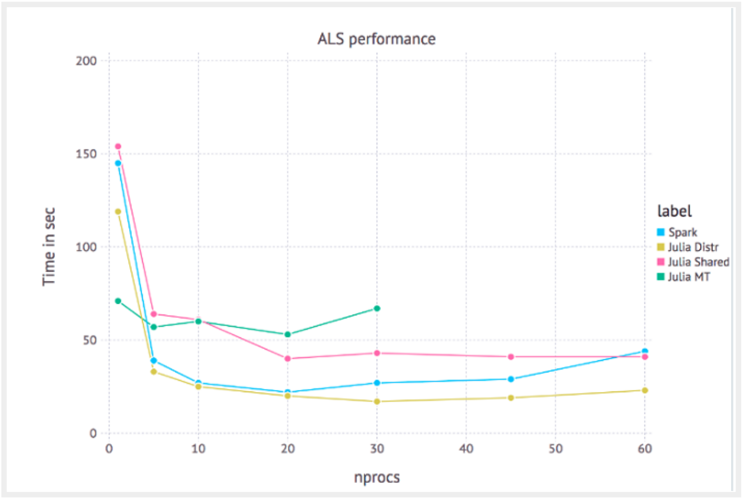
See More White Papers »

LATEST VIDEO

Video Archive

FIND US ON:

parallelized and scaled easily. The data visualization is a performance chart that shows the scaling characteristics of the system.



Scaling chart of the ALS Recommender System: "nprocs" refers to the number of worker processes in the shared memory and distributed setting and number of threads in the multithreaded setting

In the visualization, we look at the scaling performance of Julia in multi-threaded mode (Julia MT), distributed mode (Julia Distr), and shared memory mode (Julia Shared). Shared memory mode allows independent Julia worker processes (as opposed to threads) to access a shared address space. An interesting aspect of this chart is the comparison between Julia's distributed capabilities and that of Apache Spark, a popular framework for scalable analytics that is widely adopted in the industry. What's more interesting, though, are the consequences of this experiment: Julia can scale well to a higher number of nodes.

The Julia project has come a long way since its inception in 2012, breaking new boundaries in scientific computing and data science.

CLICK TO TWEET

The Julia project has come a long way since its inception in 2012, breaking new boundaries in scientific computing and data science. Conceived as a language that retained the productivity of Python while being as fast and scalable as C/C++. This approach allows Julia to be the first high-productivity language to scale well on clusters. The Julia project continues to double its user base every year and to be increasingly adopted in a variety of industries. Such advances in tooling for computational science not only bode well for scientific research to address the challenges of the coming age, but will also result in rapid innovation cycles and turnaround times in the industry.

Download Intel® Math Kernel Library

Filed Under: Big Data, Featured, Google News Feed, Intel, Machine Learning, News / Analysis, Uncategorized Tagged With: hpc, Intel TEC, Weekly Featured Newsletter Post

Comments

Ben says:
September 8, 2017 at 1:34 pm

> This approach allows Julia to be the first high-productivity language to scale well on clusters.

This claim is debatable.. Has the author of this article heard of Chapel?

– http://chapel.cray.com/
– https://www.dursi.ca/post/julia-vs-chapel.html

Reply

Leave a Comment

Name *

Email *

Website

Post Comment

☐ Notify me of follow-up comments by email.
☐ Notify me of new posts by email.

inside BIGDATA

About insideBIGDATA
Contact
Advertise with insideBIGDATA

Visit Our Other Site – insideHPC
Terms of Service & Copyright
Privacy Policy