Eclipse Tooling for Julia

Tobias Verbeke (Open Analytics NV)
March 8, 2016
Open Analytics
Data Science Company
Data Science Company

Statistical Consulting  Statistical Programming  Application Development and Integration  Data Analysis Hardware and Hosting
Julia
We are Greedy

We are greedy: we want more.

We want a language that's open source, with a liberal license. We want the speed of C with the dynamism of Ruby. We want a language that's homoiconic, with true macros like Lisp, but with obvious, familiar mathematical notation like Matlab. We want something as usable for general programming as Python, as easy for statistics as R, as natural for string processing as Perl, as powerful for linear algebra as Matlab, as good at gluing programs together as the shell. Something that is dirt simple to learn, yet keeps the most serious hackers happy. We want it interactive and we want it compiled.

http://julialang.org/blog/2012/02/why-we-created-julia
Background

- 4 people at MIT started working on such a language in 2009
- Jeff Bezanson, Stefan Karpinski, Viral Shah, Alan Edelman (Applied Math)
- in 2012 they open sourced their efforts
LLVM

- modern compiler infrastructure which allows for JIT compilation for dynamic languages
- no more two-languages paradigm (Python/C, R/C++), but a single language
- workflow: write something that works, apply a few tricks
Can't R do the same?

Computer scientist's answer*:


- R has many reflective features and therefore is a dead end for this type of compiler optimizations
- Julia is a language designed by computer scientists, but computer scientists who know about floating point arithmetic

*Pascal Costanza (Intel)
Peak Inside

JuliaLang / julia

Issues 1,383
Pull requests 307
Pulse
Graphs

Branch: master / julia / src /

eschnett: Do not require the "cx16" host feature on i686 systems

- flisp
  a few more micro-optimizations to parsing
- support
  move `dirname.c` from flisp/ to support/
- .gitignore
  add julia_version.h for version define in C API and dll export
- APIInt-C.cpp
  Remove unnecessary extern "C"
- APIInt-C.h
  Extend checked integer arithmetic
- Makefile
  Move LLVM-version-based CMake detection into common helper file
Peak Inside (contd.)

- JuliaLang / julia
  - Code
  - Issues: 1,383
  - Pull requests: 307
  - Pulse
  - Graphs

Branch: master → julia / base / strings /

- JeffBezanson: more float to string performance improvements

- basic.jl: Merge pull request #13727 from bjarthur/stringmultiply

- io.jl: more float to string performance improvements

- search.jl: Reorganize base/string.jl, base/utf*, test/strings.jl, test/unicode.jl

- types.jl: Merge pull request #12330 from JuliaLang/jb/deprecateropestring

- util.jl: bytes2hex: reimplement efficiently (close #14341)
Peak Inside (contd.)

- femtolisp: parser written in a very efficient Scheme dialect
- C/c++ core: deals with LLVM back-end work
- dependencies on the traditional scientific computing libraries (LAPACK and OpenBLAS for linear algebra operations etc.)
- all the rest: pure Julia
Feel of the Language
Data Structures or Types

julia> subtypes(Real)
4-element Array{Any,1}:
AbstractFloat
Integer
Irrational{sym}
Rational{T<:Integer}

julia> subtypes(AbstractFloat)
4-element Array{Any,1}:
BigFloat
Float16
Float32
Float64
Tempered Methods and Data Types

Statisticians like sums of squares

```plaintext
function sumsq{T <: Number}(V::Vector{T})
    s = zero(T); for v in V; s += v*v; end
    s
end
```

Inline function example

```plaintext
mean(A::AbstractArray) = sum(A) / length(A)
```
Define Custom Types

type DSL
    name :: ASCIIString
    license :: ASCIIString
end

one = DSL("Julia", "MIT")

julia> one.license
"MIT"
Multiple Dispatch

- OO paradigm: `object.method(arg1, arg2)`
- multiple dispatch: `method(object, arg1, arg2)`
- similar to Closure, S4 methods in R
- generic methods and functions
Multiple Dispatch (contd)

julia> methods(+)
# 171 methods for generic function "+",
+(x::Bool) at bool.jl:33 +(x::Bool, y::Bool) at bool.jl:36
+(y::AbstractFloat, x::Bool) at bool.jl:46 +(x::Int64, y::Int64) at int.jl:8
+(x::Int8, y::Int8) at int.jl:16 +(x::UInt8, y::UInt8) at int.jl:16
+(x::Int16, y::Int16) at int.jl:16 +(x::UInt16, y::UInt16) at int.jl:16
+(x::Int32, y::Int32) at int.jl:16 +(x::UInt32, y::UInt32) at int.jl:16
+(x::UInt64, y::UInt64) at int.jl:16

- making the first object important is counter-intuitive in scientific operations
- all operators are defined as functions
Integrate with Other Languages

julia> x = ccall((:rand, "libc"), Int32, ())
719885386

julia> using PyCall
julia> @pyimport math as pymath
julia> pymath.exp(1)
2.718281828459045
Umbrella Organizations on Github

JuliaStats, JuliaOpt, JuliaParallel, JuliaDB, JuliaGPU, BioJulia, JuliaQuantum, JuliaAstro, JuliaQuant, JuliaSparse, JuliaDiff, JuliaWeb, JuliaDSP, JuliaGraphs
Tooling
Atom Editor?

[...] we think we can have our cake and eat it too by building on top of the excellent Atom editor (Julia Blog, Jan 7 2016)
Editor and Outline View
Julia interpreters
Julia run configurations
We want it to be interactive?

- interactive console missing
- no graphic window
- tricky since requires communication with Java
- we have been there with our R tooling for Eclipse
Jupyter
Jupyter notebooks

```python
plt.figure()
plt.plot(time-tevent,strain_H1_whitenbp,'r',label='H1 strain')
plt.plot(time-tevent,strain_L1_shift,'g',label='L1 strain')
plt.plot(NRtime+0.002,NR_H1_whitenbp,'k',label='matched NR waveform')
plt.xlim([-0.1,0.05])
plt.ylim([-4,4])
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('whitened strain')
plt.legend(loc='lower left')
plt.title('Advanced LIGO WHITENED strain data near GW150914')
plt.savefig('GW150914_strain_whitened.png')
```

The signal is now clearly visible in the whitened and bandpassed data. The "DC" offset between H1 and L1 data visible in the first plot is no longer visible here: the bandpassing cuts off frequency components below around 20 Hz and above 300 Hz.

https://losc.ligo.org/s/events/GW150914/GW150914_tutorial.html
tmpnb server
Jupyter Console Plugin

- develop JSON model implementing the Jupyter protocol
- develop ZMQ interface to interact with kernels
- develop a WebSocket interface to interact with kernels
- Eclipse UI for console and Jupyter graphics view
2+2 in Julia

```julia
using Gadfly
plot([sin, cos], 0, 25)
```

Notebook server spawned: https://tmp40.tmpnb.org/user/oxMx
Julia 0.3.2 kernel launched, id: 6078a5cd-6cca-4d1c-a885-
Session started. Enjoy!

2+2
4

using Gadfly
plot([sin, cos], 0, 25)
2+2 in Python

```python
import matplotlib.pyplot as plt
import numpy as np

def plot_sin_squared():
    x = np.linspace(0, 3*np.pi, 500)
    plt.plot(x, np.sin(x**2))

plot_sin_squared()
```

Python 3 kernel launched, id: 96754efc-b752-4505-8e7d-c1
Session started. Enjoy!

```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 3*np.pi, 500)
plt.plot(x, np.sin(x**2))
```
2+2 in R
2+2 in Ruby

Jupyter <Ruby 2.1.5> <ready>

Notebook server spawned: https://tmpl41.tmpnb.org/user/nZB
Ruby 2.1.5 kernel launched, id: 0ce45d34-21ec-4c2c-b580-b
Session started. Enjoy!

2+2
4
2+2 in bash

Jupyter <Bash> <ready>

```
Notebook server spawned: https://tmp39.tmpnb.org/user/vJNC
Bash kernel launched, id: bb46e69b-49e0-4aab-8452-40d8ce5d6
Session started. Enjoy!
awk "BEGIN {print 2+2}"
4
```
2+2 in ...

http://jupyter.org
Donald Trump answers the question: What is 2+2?

"I have to say a lot of people have been asking this question. No, really. A lot of people come up to me and they ask me. They say, 'What's 2+2'? And I tell them look, we know what 2+2 is. We've had almost eight years of the worst kind of math you can imagine. Oh my God, I can't believe it. Addition and subtraction of the 1s the 2s and the 3s. It's terrible. It's just terrible. Look, if you want to know what 2+2 is, do you want to know what 2+2 is? I'll tell you. First of all the number 2, by the way, I love the number 2. It's probably my favorite number, no it is my favorite number. You know what, it's probably more like the number two but with a lot of zeros behind it. A lot. If I'm being honest, I mean, if I'm being honest. I like a lot of zeros. Except for Marco Rubio, now he's a zero that I don't like. Though, I probably shouldn't say that. He's a nice guy but he's like, '10101000101,' on and on, like that. He's like a computer! You know what I mean? He's like a computer. I don't know. I mean, you know. So, we have all these numbers, and we can add them and subtract them and add them. TIMES them even. Did you know that? We can times them OR divide them, they don't tell you that, and I'll tell you, no one is better at the order of operations than me. You wouldn't believe it. So, we're gonna be the best on 2+2, believe me."
Conclusions
Conclusions

- Julia is hot... as hell
- It is Eclipse-Ready
- Jupyter is cool
- It is Eclipse-Ready

https://github.com/JuliaComputing/JuliaDT/releases/tag/v0.0.1
http://jupyter-console.openanalytics.eu (update site)
Acknowledgements

- Julia developers
- Jupyter project
- David Dossot (Apple)
- Frederick Michielssen (Open Analytics)
- Science Working Group
Questions?

tobias.verbeke@openanalytics.eu
Thanks!

Evaluate the Sessions

Sign in and vote at **eclipsecon.org**

-1 0 +1