julia

Roadmap

Stefan Karpinski

Julia computing

NUMFOCUS
OPEN CODE = BETTER SCIENCE

NYU
Center For Data Science

Julia 1.0

Obviously, it's not out yet – we just released 0.6

- ¯_(ツ)_/¯ always knew the timeline was aggressive
- much more happened in 0.6 than originally expected

Some questions:

- how much of 1.0 have we done?
- how does this compare to past releases?

	Feature	Status June 2017	Complete	Defered	Total
			48%	**25%**	**73%**
8	#265	DONE!! □□□	100%	0%	100%
10	Type system redesign	DONE!! □□□	100%	0%	100%
5	Vectorized operations: fuse dot ops	DONE!! □□□	100%	0%	100%
20	Converting docs from RST to Markdown	DONE!! □□□	100%	0%	100%
30	Finish String-apalooza	DONE!! □□□	100%	0%	100%
11	Reimplement string with buffer	done (obviated by Jeff)	100%	0%	100%
29	Unit Testing Infrastructure	much improved, need line numbers or	90%	10%	100%
34	Standalone binaries for companies to ship	close but ~5 issues left	85%	15%	100%
22	Multi-threading: run-time safety	done but always needs work	80%	20%	100%
19	Debugger	needs a lot of work	70%	30%	100%
25	Stack optimization	partially done	50%	50%	100%
15	GPU codegen	much better, not 1.0 blocker	50%	50%	100%
18	Compile time Latency problems	getting worse; work on it post 1.0	20%	80%	100%
26	Doc and tutorial writing	not done	15%	85%	100%
36	Buffer type	not crucial	0%	100%	100%
1	Vectorized operations: eliminate boundschecks on maps	post 1.0	0%	100%	100%
2	Vectorized operations: loop fusion	post 1.0 (syntactic works, optimization	0%	100%	100%
4	Replace / improve @printf insanity	post 1.0 (remove from Base?)	0%	100%	100%
32	Reimplement arrays with buffer	not happening in 1.0	0%	100%	100%
27	Linear Algebra changes	do A_mul_B stuff	75%	0%	75%
7	Pkg3	so close	75%	0%	75%
31	Finalize I/O API	this is probably close to done	75%	0%	75%
6	Record types / named tuples	coming soon	70%	0%	70%
14	Restructuring standard library	in progress (Alex in charge)	60%	0%	60%
28	Multi-threading: scheduler	depends on Kiran, August?	50%	0%	50%
24	Documentation for all public types and functions (eg. bitstypes)	ask Alex to work on this?	50%	0%	50%
12	Nullable support / result type / discriminated union	under way	50%	0%	50%
35	Parallel computing - API revamp	much progress, more needed (Amit, A	50%	0%	50%
3	Multi-threading: I/O	make it not segfault, give warning	0%	50%	50%
9	Replace llvmcall	design done. TODO	25%	0%	25%
23	Finalize collections API	pick an option from Milan's Julep, do it	25%	0%	25%
16	Package/module name conflicts	not done (covered by above item)	10%	0%	10%
21	Issues in language modularity features (relative using, method m	not done, needs to happen	0%	0%	0%
33	intersection / conditional modules	not done, needs to happen	0%	0%	0%
13	BinDeps2	not done, needs to happen	0%	0%	0%

Julia release history

v0.1 – 2013 Feb 13

276 days ≈ 9.2 mo

v0.2 – 2013 Nov 16

277 days ≈ 9.2 mo

v0.3 – 2014 Aug 20

413 days ≈ 13.8 mo

v0.4 – 2015 Oct 7

348 days ≈ 11.6 mo

v0.5 – 2016 Sep 19

273 days ≈ 9.1 mo

v0.6 – 2017 Jun 19

Julia 1.0

Milestone on GitHub is now a fairly accurate reflection of work to do

- nothing huge planned for this release!
- a lot of cleanup and planning ahead for 1.x

Need to move everything that can be non-breaking to 1.1+

- optimizations, compiler improvements, features
- e.g. don’t upgrade LLVM from 3.9.1 to 5.0 until later

Prioritize the most disruptive changes early in the release

Julia 0.7 ?!?!1?

You may have heard and seen "Julia 0.7" being talked about

- `VERSION` file contains "`0.7.0-DEV`"

Don't worry, we're not doing an additional release cycle!

- 0.7 = 1.0 with deprecations

To upgrade from 0.6:

- port your code to 0.7 and fix deprecation warnings as usual
- switch to 1.0 and fix anything else that breaks (ideally nothing)

Beyond 1.0

I’ve previously said that 2.0 might be only 1-2 years after 1.0

- we’ve been rethinking this – might be annoying
- people want a long-term stable platform

Beyond 1.0

There are a ton of things we can do that aren’t breaking

- optimizations, optimizations, optimizations
- upgrading infrastructure: LLVM 5.0, libuv, …
- adding features like traits, protocols, multiple inheritance, …
- work on key packages: DataStructures, DataFrames/Tables, …
- tooling: debugging, profiling, static analysis, …