The impact of operating systems and environments on build results

Mahdis Zolfagharinia Master Thesis Defense

Faculty of Computer Engineering

Advisors: Dr. Bram Adams and Dr. Yann-Gaël Guéhéneuc

December 12, 2017



Continuous Integration (CI)

cornerstone of modern quality assurance, providing on-demand builds of code changes or software releases.



Travis CI Build Environment

Travis Cl Blog	Status Help		Sign in with GitHub 💭
rails / rails (Current Branches Build	History Pull Requests		More options 🗮
★ 5-0-stable Matthew Draper	Merge pull request #29022 from y-yagi/allow_	#43037 failed	 S hrs 34 min 29 sec 2 days ago
✓ 5-1-stable↑ Matthew Draper	Merge pull request #29022 from y-yagi/allow_	#43036 passed	 6 hrs 8 min 37 sec 27 2 days ago
✓ master ↑ Matthew Draper	Merge pull request #29022 from y-yagi/allow_1	-∽- #43035 passed ♀ 97bd56e	 6 hrs 26 min 41 sec 27 2 days ago
X 3-2-stable	CRON Remove `DEFAULT NULL` for primary	௴ #43033 failed ♀ e17e25c	 9 hrs 26 min 35 sec 27 2 days ago
✓ 5-1-stable↑ Matthew Draper	Merge pull request #28995 from jcoyne/update	-∽- #43032 passed ♀ e8af77a	 6 hrs 12 min 3 sec 27 2 days ago
I 4-2-stable Pafael França	CRON Merge pull request #28815 from joshnu	௴ #43029 errored ♀ f626b47	 6 hrs 48 min 8 sec 3 days ago

Travis CI Build Environment

rails / rails ($\ensuremath{\wp}$ build passing

Current Branches Build History Pull Requests	More options	\equiv
X 3-2-stable CRON Remove `DEFAULT NULL` for primary key column to support MySQL 5.7.3		
Since MySQL 5.7.3 m13 does now allow primary key column is null.		
(cherry picked from commit b6655885ef13cf8d1705dc9b5232846f0207febd)	() Total time 9 hrs 26 min 35 sec	
 ↓ Commit e17e25c ↓ Branch 3-2-stable 		
🚭 Yasuo Honda authored 🛛 👷 Andrew White committed		

Build Jobs

× # 43033.1	🗞 Ruby: 1.8.7	GEM=railties	() 2 min 24 sec
✓ # 43033.2	🖓 > Ruby: 1.9.2	GEM=railties	() 25 min 37 sec
✓ # 43033.3	🖓 > Ruby: 1.9.3	GEM=railties	() 21 min 39 sec
✓ # 43033.4	🗞 Ruby: 2.0.0	GEM=railties	() 13 min 46 sec
✓ # 43033.5	🗞 Ruby: 2.1.8	GEM=railties	() 19 min 8 sec
✓ # 43033.6	🖓 Ruby: 2.2.6	GEM=railties	() 11 min 12 sec
✓ # 43033.7	🖓 > Ruby: 2.3.3	GEM=railties	() 11 min 11 sec
✓ # 43033.8	🖓 Ruby: 1.8.7	GEM=ap,am,amo,ares,as	() 7 min 45 sec
✓ # 43033.9	🗟 Ruby: 1.9.2	GEM=ap,am,amo,ares,as	() 9 min 57 sec
✓ # 43033.10	🗞 Ruby: 1.9.3	GEM=ap,am,amo,ares,as	() 8 min 49 sec
✓ # 43033.11	🗞 > Ruby: 2.0.0	GEM=ap,am,amo,ares,as	() 6 min 27 sec
•			



Travis CI Build Environment

rails / rails ($\ensuremath{\wp}$ build passing

Curre	Branches Build History Pull Requests		More options	\equiv
×	3-2-stable CRON Remove `DEFAULT NULL` for primary key column to support MySQL 5.7.3	ඌ #43033 failed		
	Since MySQL 5.7.3 m13 does now allow primary key column is null.	් Ran for 25 min 37 sec (Total time 9 hrs 26 min 35 sec		
	Image: Commit e17e25c Image: Branch 3-2-stable	🗊 about an hour ago		
	🚳 Yasuo Honda authored 🛛 👰 Andrew White committed			

Build Jobs

× # 43033.1		GEM=railties	() 2 min 24 sec
✓ # 43033.2		GEM=railties	(25 min 37 sec
✓ # 43033.3		GEM=railties	() 21 min 39 sec
✓ # 43033.4		GEM=railties	(13 min 46 sec
✓ # 43033.5		GEM=railties	() 19 min 8 sec
✓ # 43033.6) GEM=railties	(§ 11 min 12 sec
✓ # 43033.7		GEM=railties	() 11 min 11 sec
✓ # 43033.8	🛞 > Ruby: 1.8.7	GEM=ap,am,amo,ares,as	(§ 7 min 45 sec
✓ # 43033.9		D GEM=ap,am,amo,ares,as	() 9 min 57 sec
✓ # 43033.10	Ruby: 1.9.3	GEM=ap,am,amo,ares,as	(8 min 49 sec
✓ # 43033.11	🛞 > Ruby: 2.0.0	GEM=ap,am,amo,ares,as	() 6 min 27 sec

Travis CI build environment

rails / rails	Build History	Id passing Of Pull P squests	peratin System	Run-time Environment		More options 🗮
X 3-2-stable CR Since MySQL 5 Cohenny picke	ON Factore	<pre>> DEFAULT NULL` for primar > Ruby: 1.8.7</pre>	y key column to support MySQL column is null.	53 CS #43033 failed © Ran for 25 min 37 so © Total time 9 hrs 26 r	sec min 35 sec	
Commit e17e		> Ruby: 1.9.2		GEM=railties		
Yasuo Honda		> Ruby: 1.9.3		GEM=railties		
	ß	> Ruby: 2.0.0		GEM=railties		
× # 43033.1	æ	() Puby 2.1.9		GEM=railties	0 0	2 min 24 sec 25 min 37 sec
✓ # 43033.3	<i>\$</i> 3	W RUDY: 2.1.8		GEM=railties	0	21 min 39 sec
✓ # 43033.4	- Co	> Ruby: 2.2.6		ℜ GEM=railties	<u> </u>	13 min 46 sec
✓ # 43033.5✓ # 43033.6	\$	> Ruby: 2.3.3			C.	11 min 12 sec
✓ # 43033.7	0			GEM=railties	0	11 min 11 sec
✓ # 43033.8		> Ruby: 1.8.7		🗇 GEM=ap,am,amo,ares,as	0	7 min 45 sec
✓ # 43033.9✓ # 43033.10	\$	> Ruby: 1.9.2		GEM=ap,am,amo,ares,as	© 0	9 min 57 sec 8 min 49 sec
✓ # 43033.11	A	() Dubu 102		9.051	0	6 min 27 sec
	8	W Ruby: 1.9.3		U GEM=ap,am,amo,ares,as		
		> Ruby: 2.0.0		🗇 GEM=ap,am,amo,ares,as		7

Not just Travis CI How to Interpret Build Results for cparDifferment: Geometingtions of OS/env.?!

Submit

Distribution (e.g. DBI, CPAN-Reporter, YAML-Syck): ALL cygwin darwin dragonfly freebsd gnukfreebsd mswin32 netbsd linux openbsd solaris 5.19.3 5.19.2 5.18.1 **5.18.0** 5.16.3 5.16.2 5.16.1 5.16.0 5.14.4 5.14.3 5.14.2 5.14.1 5.14.0 5.12.5 5.12.4 5.12.3 5.12.2 5.12.1 5.12.0 5.10.1 5.10.0 5.8.9 5.8.8 5.10.0 5.8.9 8 5.8.8



The impact of OSes and environments on build results to analyze the phenomenon of "build inflation"

Difficult to interpret build results and detect





Practitioners and researchers should take it into consideration when analyzing build results



Hilton et al. ASE 2016:

- Analyzed ~35K open-source projects of GitHub
- ***** Over 40% of these projects use CI
- * Analyzed ~1.5M builds of Travis CI

* CI is widely adopted in popular projects and helped to decrease the time between releases

Merzazi et al. SANER 2016:

* Studied releases of a large e-commerce web app

* Source code is not the main cause of problems, but defective configurations are the main issues

Seo et al. ACM 2014

* Assessed 26.6 million builds in C and Java at Google

* Focused on build failures due to compilation problems



* Dependencies are the main source of compilation errors

Kerzazi et al. ICSME 2014:

* Analyzed ~3K builds to study build failures

*~18% of the build failures have a potential cost of ~2K man-hours (1h for each build to succeed)



Build inflation is an actual phenomenon in open-source projects that impacts the interpretation of build results



Does build frequency/failure differ across OS/env.?



What are the impact of OSes on build failure types?



To what extent do environments impact build failures?



To what extent do environments impact build failure types?

Microsoft

nython" 🥐

What percentage of build failures are OS independent?

CP

Case Study Setup

Empirical study of 30M builds

- perity
- Study the whole history of CI build packages on CPAN
- ✤ 39K packages, 10 Oses, 103 Perl run-time environments



Filtering CPAN Packages Based on Distribution of the Number of Builds and versions





Does build frequency/failure differ across OS/env.?







Does build frequency/failure differ across OS/env.?

Different OSes/env. have different frequency & failureproneness



To what extent do environments impact build failures?

How Often Environments Break the Build And to What Degree a Failing Build can Recover

CPAN Testers Matrix: List-Objects-Types 0.004002





To What Degree a Failing Build Can Recover

- Our observation shows that 12% of changes to new environment have
- 11% were unable to recover



Does build frequency/failure differ across OS/env.?

Different OSes/env. have different frequency & failureproneness



To what extent do environments impact build failures?

77\% and 6\% of the environment vectors consistently succeed or fail



What percentage of build failures are OS independent?

Do. Builds Sacese Ad Tail Spesificently eacross all OSSes?...





100

Median of 86.5% of Builds Succeed/Fail Consistently



Number of OSes on which packages were built





Median of 30% of Inconsistent Successes in Windows vs. 7% in Linux





Does build frequency/failure differ across OS/env.?





To what extent do environments impact build failures?

77\% and 6\% of the environment vectors consistently succeed or fail



What percentage of build failures are OS independent?

86.5% of build results were consistent → evidence of build inflation



What are the impact of OSes on build failure types?

Comprehending build failures and their relationships with

environments/OSes is helpful to solve future failures.

Manually analyzed the build logs and categorized the failures according

to the reason of their occurrence.



Missing modules is the main reason of failures overall Majority failures: programming faults Minority failures: environment faults







Windows and Cygwin are the most fault-prone OSes and experience a wide range of fault types in minority failures Linux 66.7% is responsible for the "data format" failures and 33.3% for the "configuration"





Does build frequency/failure differ across OS/env.?





To what extent do environments impact build failures?

77\% and 6\% of the environment vectors consistently succeed or fail



What are the impact of OSes on build failure types?

Windows and Cygwin are the most fault-prone OSes and experience a wide range of fault types



Sun Cobalt What percentage of build failures are OS independent?

86.5% of build results were consistent → evidence of build inflation



To what extent do environments impact build failure types?



0-0 and 1-1 patterns confirm the occurrence of build inflation

Dependency faults for majority failures are more difficult to resolve than for minority

failures.



Minority Failures

Majority Failures



Does build frequency/failure differ across OS/env.?



What are the impact of OSes on build failure types?





To what extent do environments impact build failures?

77\% and 6\% of the environment vectors consistently succeed or fail





What percentage of build failures are OS independent?

86.5% of build results were consistent → evidence of build inflation



To what extent do environments impact build failure types?



10-fold Increase in Average #builds per Release



Build Inflation: More Builds Finding Less Failures



THREATS TO VALIDITY



Only one ecosystem and programming language





Filtered out builds without corresponding failure

OS version/architecture were ignored

Future Work

How to **interpret**

build results correctly?

Replicating this study on other ecosystems and programming languages



Ways to prioritize the

most valuable OS/env.

to test on

Publications

☑ "Do not trust build results at face value: an empirical study of 30 million CPAN builds." *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE Press, 2017

Solution Extension of MSR paper is submitted to the Springer journal on Empirical Software Engineering