Exploring Descriptive Metrics of Build Performance

A Study of GitHub Actions in Continuous Integration Projects

Author
Radu Stefan Constantinescu
Contact: Constantinescu-3@tudelft.nl

Responsible Professor
Sebastian Proksch

Supervisor
Shujun Huang

01 Introduction

- **Continuous Integration (CI)**
  - practice in software development, widely recognised and adopted
  - involves frequent merging into central repository [1].
  - CI implementation != "one size fits all solution" <- constraints and contexts.
- **Performance of CI Build Stage**
  - "heart of software development ecosystem" [2]
- **Github Actions - Compelling Dev Option [3]**
  - flexibility
  - robustness
  - tight GitHub integration

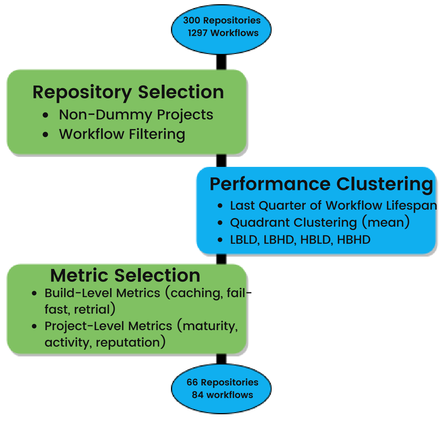
02 Research Question

What are the most descriptive metrics for identifying build performance?

- **RQ1:** What are the key **build level metrics** that significantly contribute to the evaluation of build performance?
- **RQ2:** What are the essential **project level metrics** that play a significant role in the assessment of build performance?

03 Methodology

- **GitHub projects** analysis
- Focus on **GitHub Actions** as CI tool. (Travis CI - extensively studied) [4]
- **Gap:** Isolation study of performance aspects [4]
- Performance metrics: **build breakage** & **build duration**



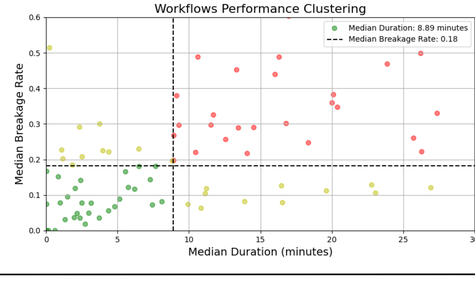

```yaml
name: Java CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 17
        uses: actions/setup-java@v3
        with:
          java-version: '17'
          distribution: 'temurin'
      - name: Build with Maven
        run: mvn --batch-mode --update-snapshots package
```

Java Gradle Build and Test Workflow Template [5]



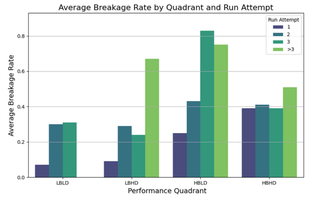
04 Results

1. **Principle: "If it's not broken don't fix it"**
2. **Keep the main clean, experiment on other branches**

Table 3: Build-Level Metrics on different branches

Quadrant	branch type	breakage rate mean	resolution time median	consecutive fails mean	job churn mean	run count mean)
LBLD	main	0.05	47.10	0.25	-0.00	312.21
LBLD	others	0.15	0.00	0.45	0.00	9.60
LBHD	main	0.06	135.39	0.92	-0.00	235.44
LBHD	others	0.12	0.00	0.48	0.06	11.00
HBLD	main	0.25	92.38	6.53	-0.05	141.42
HBLD	others	0.31	5.76	1.00	-0.01	11.43
HBHD	main	0.33	219.98	13.91	0.00	235.64
HBHD	others	0.34	31.78	1.06	0.27	10.38

3. **Retrying failed builds won't fix them**

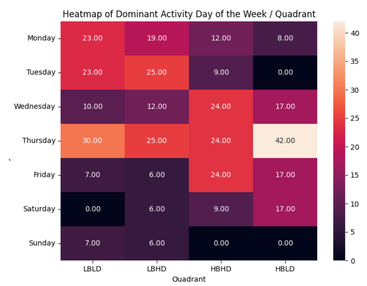


4. **Strong Performance indicators: Caching, Fail-Fast, Skipping**

Table 6: Fail-fast (FF) and No Cache and Skip Usage in Performance Clusters

Quadrant	Fail Fast Disabled %	Cache Disabled %	Skip Usage %
LBLD	6.67%	6.67%	13.33%
LBHD	6.25%	12.50%	25.00%
HBLD	16.67%	0.00%	0.00%
HBHD	26.47%	14.71%	2.94%

5. **CI Build Activity - Project context dependent**

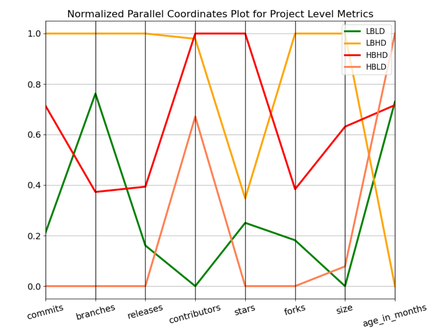


6. **Keep configuration simple, solve failures fast**

Table 4: Stats of # Jobs Configured / Workflow

Quadrant	Average	Median	p80	p95	p99
LBLD	1.50	1.00	2.00	3.55	5.42
LBHD	1.94	1.00	2.00	5.00	7.4
HBLD	1.08	1.00	1.00	1.45	1.89
HBHD	6.91	5.50	9.00	18.05	29.03

7. **Keeping team size low and having a small project, practices of good performance**
8. **Adopt CI best practices fast, before it's too late**
9. **Maturity: Influence of Build Performance**



05 Limitations

- **Limited #** repositories studied.
- Possibility of **bias** in repository selection.
- Looking at **restricted history** of builds.
- Workflow **filtering**.
- Observations based on **cluster properties.**

06 Conclusions

- New CI tools, like GitHub Actions, show **similar patterns** in terms of best practices to already studied CI technologies.
- Build Level Metrics: **Job Churn**, **Caching**, **Fail-Fast** configurations, **Skipping** usage, show clear relation to build performance.
- Project Metrics: **maturity** and **project context**, strong pre-requisites needed for a holistic understanding of performance

07 References

[1] Omar Elazhary, Colin Werner, Ze Shi Li, Derek Lowlind, Neil A Ernst, and Margaret-Anne Storey. Uncovering the benefits and challenges of continuous integration practices. IEEE Transactions on Software Engineering, 48(7):2570–2583, 2021.
[2] Shane McIntosh, Bram Adams, Thanh HD Nguyen, Yasutaka Kamei, and Ahmed E Hassan. An empirical study of build maintenance effort. In Proceedings of the
33rd international conference on software engineering, pages 141–150, 2011
[3]Alexandre Decan, Tom Mens, Pooya Rostami Mazrae, and Mehdi Golzadeh. On the use of github actions in software development repositories. In 2022 IEEE International Conference on Software Maintenance andEvolution (ICSME), pages 235–245, 2022.
[4] Taher Ghaleb, Safwat Hassan, and Ying Zou. Studying the interplay between the durations and breakages of continuous integration builds. IEEE Transactions on Software Engineering, pages 1–21, 11 2022
[5] Github. Build and test java with gradle. https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-java-with-gradle#using-the-gradle-starter-workflow, 2023. Accessed June 21, 2023