NAB: Automated Large-Scale Multi-language Dynamic Program Analysis in Public Code Repositories

Alex Villazón
Universidad Privada Boliviana
Bolivia
avillazon@upb.edu

Haiyang Sun
Università della Svizzera italiana
Switzerland
haiyang.sun@usi.ch

Andrea Rosà
Università della Svizzera italiana
Switzerland
andrea.rosa@usi.ch

Eduardo Rosales
Università della Svizzera italiana
Switzerland
rosale@usi.ch

Daniele Bonetta
Oracle Labs
United States
daniele.bonetta@oracle.com

Isabella Defilippis
Universidad Privada Boliviana
Bolivia
isabelladefilippis@upb.edu

Sergio Oporto
Universidad Privada Boliviana
Bolivia
sergiooporto@upb.edu

Walter Binder
Università della Svizzera italiana
Switzerland
walter.binder@usi.ch

Abstract

This paper describes NAB, a novel framework to execute custom dynamic analysis on open-source software hosted in public repositories. NAB is fully-automatic, language-agnostic and scalable. We present NAB's key features and its architecture. We also discuss three large-scale case studies enabled by NAB on more than 56K Node.js, Java, and Scala projects.

CCS Concepts • **Software and its engineering → Dynamic analysis**.

Keywords Dynamic program analysis; code repositories; GitHub; Node.js; Java; Scala; promises; JIT-unfriendly code; task granularity

ACM Reference Format:
Alex Villazón, Haiyang Sun, Andrea Rosà, Eduardo Rosales, Daniele Bonetta, Isabella Defilippis, Sergio Oporto, and Walter Binder. 2019. NAB: Automated Large-Scale Multi-language Dynamic Program Analysis in Public Code Repositories. In *Proceedings of the 2019 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH Companion '19), October 20–25, 2019, Athens, Greece.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3359061.3362777

1 Introduction

Analyzing today's large code repositories has become an important research area for understanding and improving different aspects of modern software systems. Despite the presence of a large body of work on mining code repositories through static analysis, studies applying dynamic analysis to open-source projects are scarce and of limited scale. Nonetheless, being able to apply dynamic analysis to the projects hosted in public code repositories is fundamental for large-scale studies on the runtime behavior of applications, which can greatly benefit the programming-language and software-engineering communities.

To enable large-scale studies on the wild requiring dynamic analysis, we propose NAB [4], a novel, distributed, container-based infrastructure for massive dynamic analysis on code repositories hosting open-source projects, which may be implemented in different programming languages. NAB automatically looks for available executable code in a repository, instruments it according to a user-defined dynamic analysis, and runs the instrumented code. Such executable code could correspond to existing benchmarks (e.g., workloads defined by the developers via the Java Microbenchmark Harness (JMH)) or software tests (e.g., defined in the default test entry of a Node.js project managed by Node Package Manager (NPM), or based on popular testing frameworks such as JUnit).

NAB resorts to containerization for efficient sandboxing, for the parallelization of dynamic analysis execution, and for simplifying the deployment on clusters or in the Cloud. Sandboxing is important to isolate the underlying execution environment and operating system, since NAB executes unverified projects that may contain buggy or even harmful

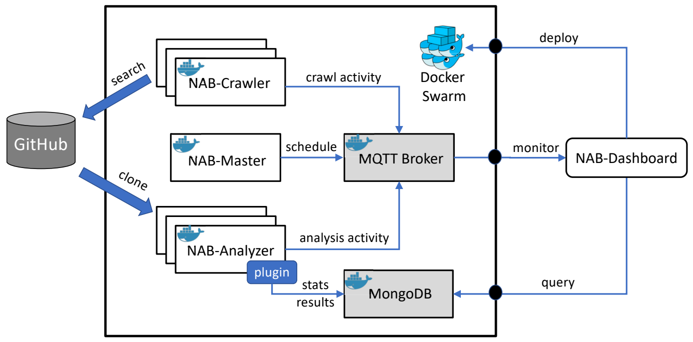

Figure 1. Overview of NAB. New components are shown as white boxes, whereas existing containerized services are marked in gray.

code. Also, parallelizing dynamic analysis execution is an important feature for massive analysis, as sequential analysis of massive code repository would take prohibitive time.

Figure 1 depicts the overall NAB architecture based on Docker containers. At its core, NAB features a microservice architecture based on a master-worker pattern relying on a publish-subscribe communication layer using the MQTT protocol, allowing asynchronous events to be exchanged between its internal components. NAB uses existing containerized services and introduces four new components, three of them running in containers: NAB-Crawler, NAB-Analyzer, and NAB-Master; as well as one external service, NAB-Dashboard. The NAB-Crawler instances are responsible for mining and crawling code repositories, collecting metadata that allows making a decision on which projects to analyze. The NAB-Analyzer instances are responsible for downloading the code, applying some filtering based on user-defined criteria and eventually running the dynamic analysis. The results generated by the dynamic analysis (such as profiles containing various dynamic metrics) are stored in a NoSQL MongoDB database. NAB provides a plugin mechanism to integrate different dynamic analyses in NAB-Analyzer instances.

NAB-Master orchestrates the distribution of crawling and dynamic analysis activities with NAB-Crawler and NAB-Analyzer instances. NAB-Dashboard is responsible for the deployment of NAB components through the Docker Swarm orchestration service and monitors the progress of an ongoing dynamic analysis. Finally, NAB supports different build systems, testing frameworks, and runtimes (see Table 1), thus enabling multi-language support. Moreover, it can easily integrate existing dynamic analyses.

We used NAB to conduct three large-scale case studies applying dynamic analysis on more than 56K open-source projects hosted on GitHub, leveraging unit tests that can be automatically executed and analyzed. We performed a novel analysis that sheds light on the usage of the Promise API in open-source Node.js projects. We found many projects

Table 1. NAB supported languages, build systems, analysis frameworks, dynamic analyses, and runtimes.

Language	Build System	Analysis Framework	Dynamic Analysis	Runtime
JavaScript	NPM	NodeProf [3]	Deep-Promise [4] JITProf	GraalVM
Java	MVN	DiSL [1] AspectJ	tgp [2] JavaMOP	HotSpot VM GraalVM
Scala	SBT MVN	DiSL	tgp	HotSpot VM GraalVM

with long promise chains, which can potentially be considered for benchmarking promises on Node.js. Moreover, the results of our analysis could be useful for Node.js developers to find projects and popular modules that use promises for asynchronous executions, whose optimization could be beneficial to several existing applications. We also conducted a large-scale study on the presence of JIT-unfriendly code on Node.js projects. Our study revealed that Node.js developers frequently use code patterns that could prevent or jeopardize dynamic optimizations and have a potential negative impact on applications performance. Finally, we performed a large-scale analysis on Java and Scala projects, searching for task-parallel workloads suitable for inclusion in a benchmark suite. We identified five candidate workloads (two in Java and three in Scala) that may be used for benchmarking task parallelism on the JVM. Overall, our case studies confirm that NAB can be used for applying dynamic analysis massively on public code repositories, and that the large-scale analyses enabled by NAB provide insights that are of practical interest.

A preliminary version of NAB can be downloaded at http://dag.inf.usi.ch/software/nab/. More information on NAB and on the described use cases can be found in our previous publication [4]. We are actively working on an open-source release of NAB.

Acknowledgments

This work has been supported by Oracle (ERO project 1332), the Swiss National Science Foundation (scientific exchange project IZSEZ0_177215), the Hasler Foundation (project 18012), and by a Bridging Grant with Japan (BG 04-122017).

References

[1] L. Marek, A. Villazón, Y. Zheng, D. Ansaloni, W. Binder, and Z. Qi. 2012. DiSL: A Domain-specific Language for Bytecode Instrumentation. In *AOSD*. 239–250.
[2] A. Rosà, E. Rosales, and W. Binder. 2018. Analyzing and Optimizing Task Granularity on the JVM. In *CGO*. 27–37.
[3] H. Sun, D. Bonetta, C. Humer, and W. Binder. 2018. Efficient Dynamic Analysis for Node.Js. In *CC*. 196–206.
[4] A. Villazón, H. Sun, A. Rosà, E. Rosales, D. Bonetta, I. Defilippis, S. Oporto, and W. Binder. 2019. Automated Large-scale Multi-language Dynamic Program Analysis in the Wild. In *ECOOP*. 20:1–20:27.