Automated Large-scale Multi-language Dynamic Program Analysis in the Wild

Alex Villazón[1], Haiyang Sun[2], Andrea Rosà[2], Eduardo Rosales[2], Daniele Bonetta[3], Isabella Defilippis[1], Sergio Oporto[1], Walter Binder[2]

[1]Universidad Privada Bolivia (UPB), Bolivia
[2]Università della Svizzera italiana (USI), Switzerland
[3]Oracle Labs, United States

UPB
Garantía de Excelencia

USI
Università
della
Svizzera
italiana

ECOOP * Artifact * AEC
Consistent * Complete * Well Documented * Easy to Reuse
* Evaluated *

Our Work

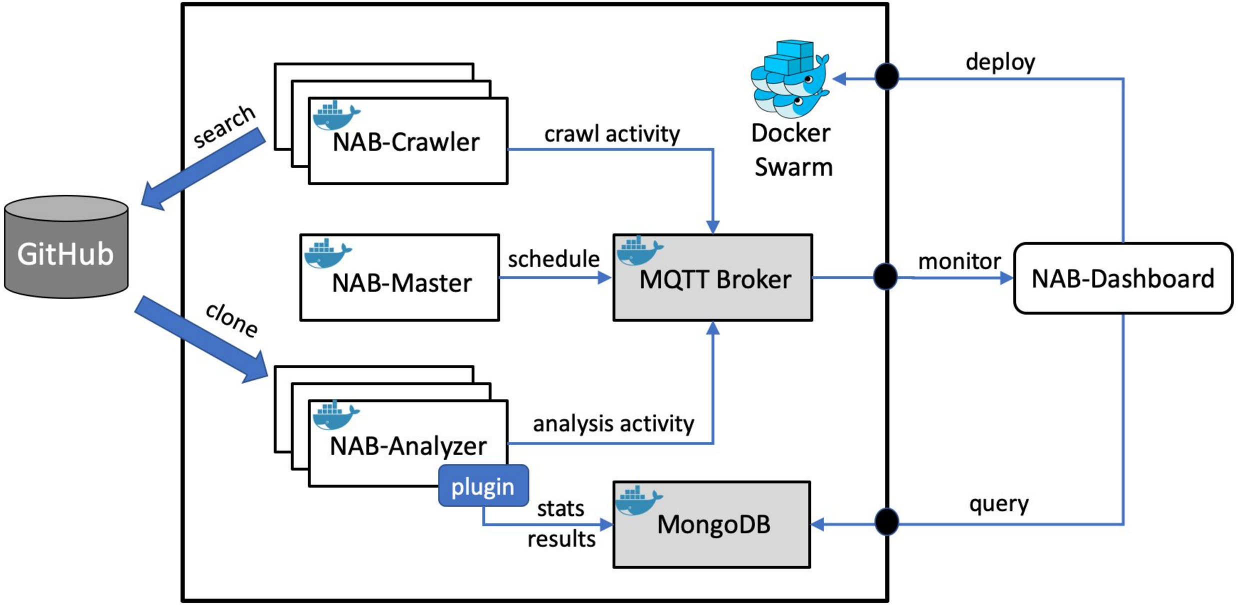
- **Goal:** Propose a methodology for **automatically** applying **Dynamic Program Analysis (DPA)** at a **large-scale** on projects hosted in public **open-source repositories**

- Motivation:
  - Applying DPA in large code repositories is increasingly important
  - Existing infrastructures focus mainly on static analysis

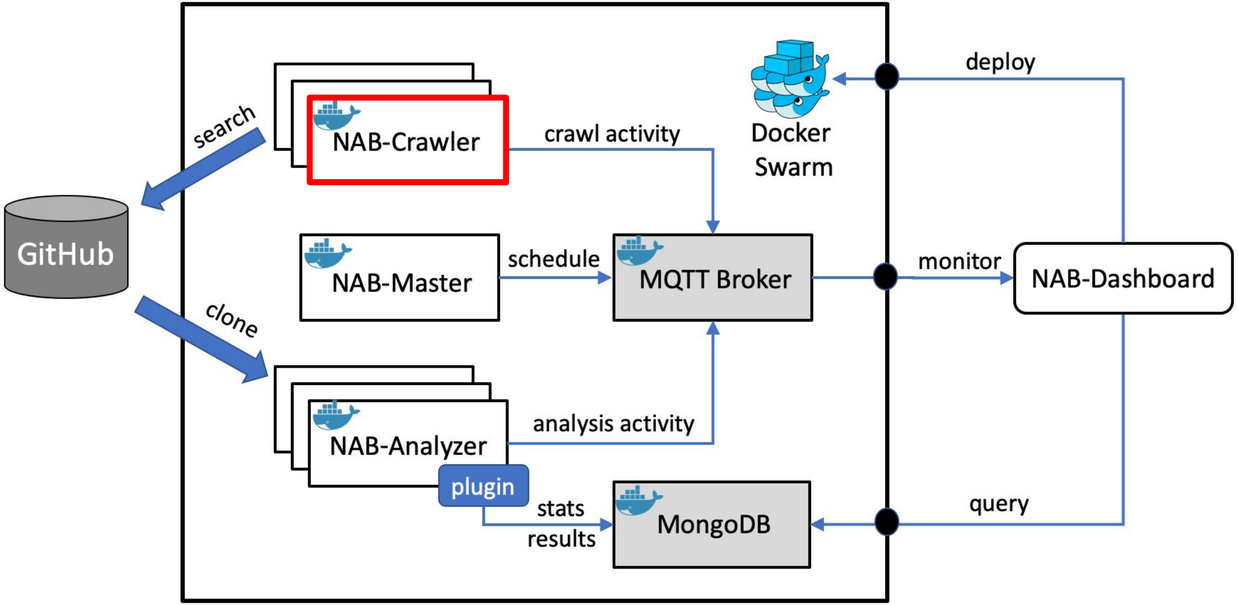
NAB: A Distributed Infrastructure for Automated DPA at Large Scale

- Automatically looks for executable code in public repositories
  - E.g., GitHub
- Filters out projects according to user-defined criteria
  - E.g., programming language, date of last commit, # contributors
- Attempts to apply DPA on workloads that can be automatically executed
  - E.g., tests (via build systems such as Maven, NPM, SBT)
- Uses containerization (Docker)
  - Simplified distributed deployment to increase scalability
  - Easy to integrate different runtimes; support for multiple languages
  - Natural and efficient sandboxing to protect from buggy or malicious code

NAB Architecture

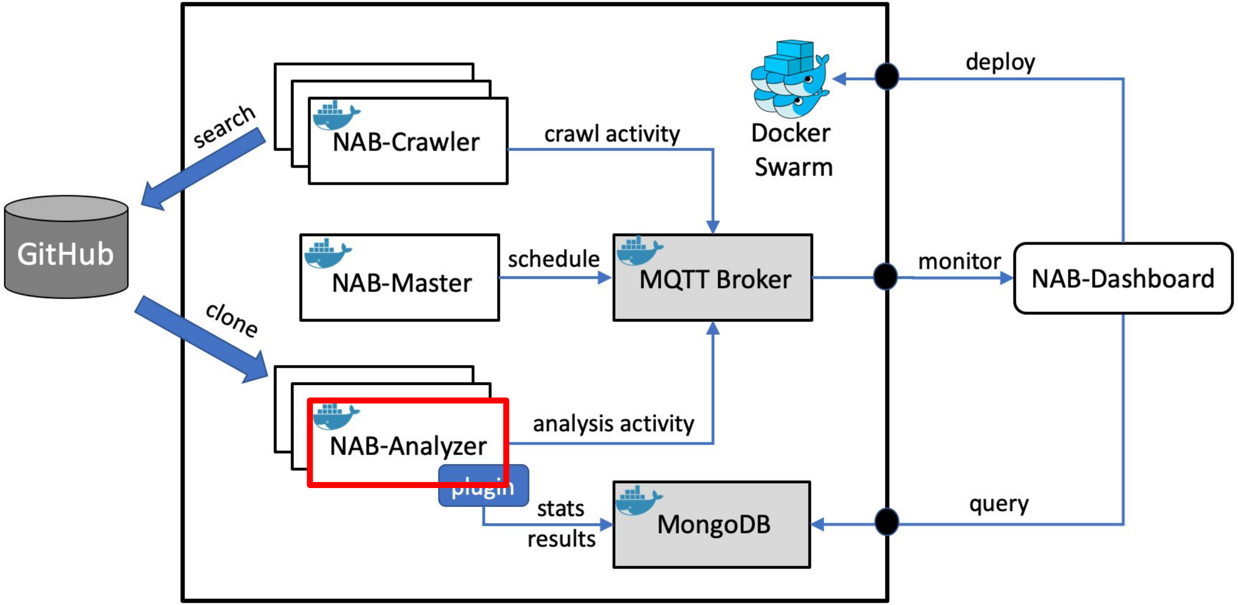


NAB Architecture



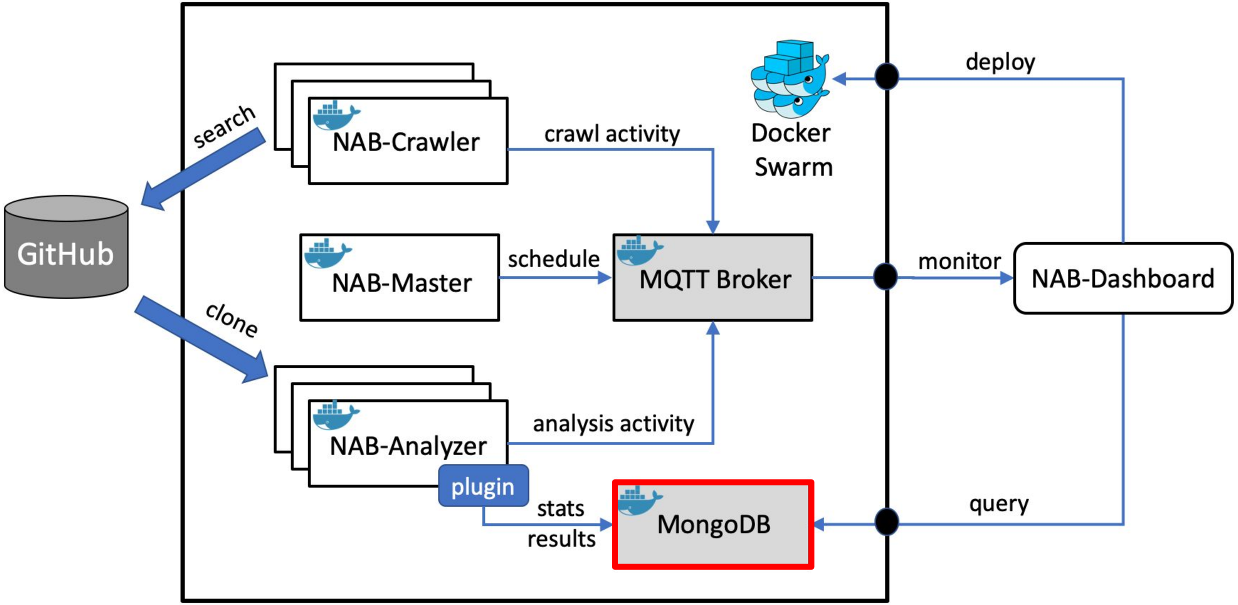
NAB-Crawler: crawls and mines code repositories,
determine projects to analyze (according to user-defined criteria)

NAB Architecture



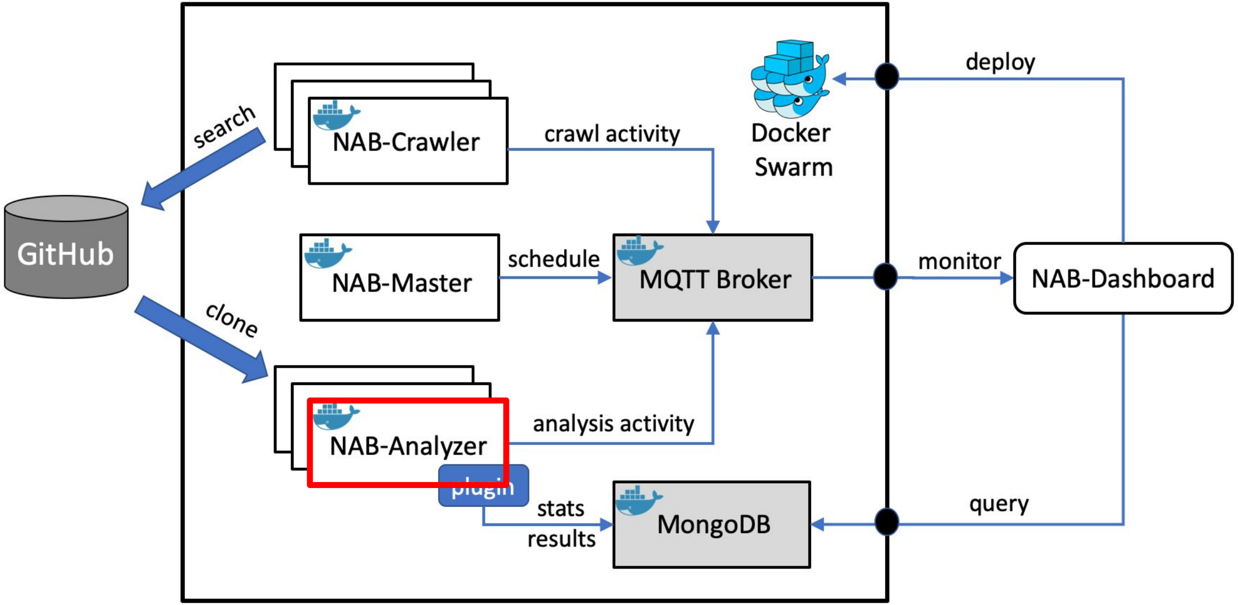
NAB-Analyzer: clones code from repositories, builds code, runs DPA on executable workloads

NAB Architecture



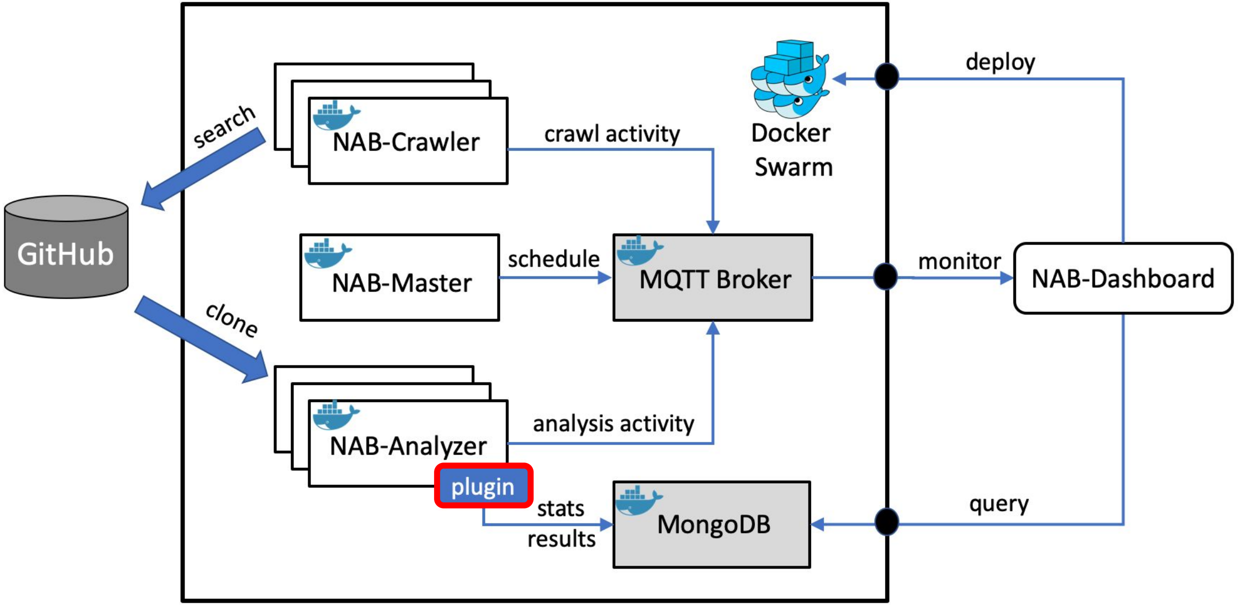
MongoDB: stores DPA results, metrics, and execution statistics

NAB Architecture



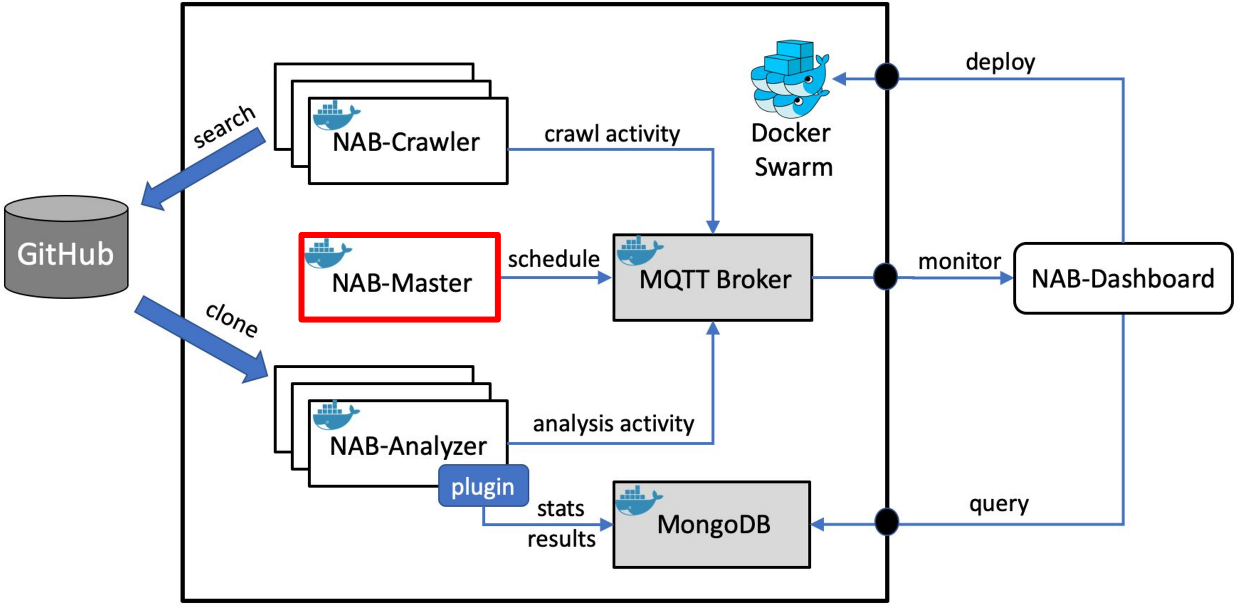
- Reports reasons of failures
- Configurable analysis timeout (default: 1 hour)

NAB Architecture



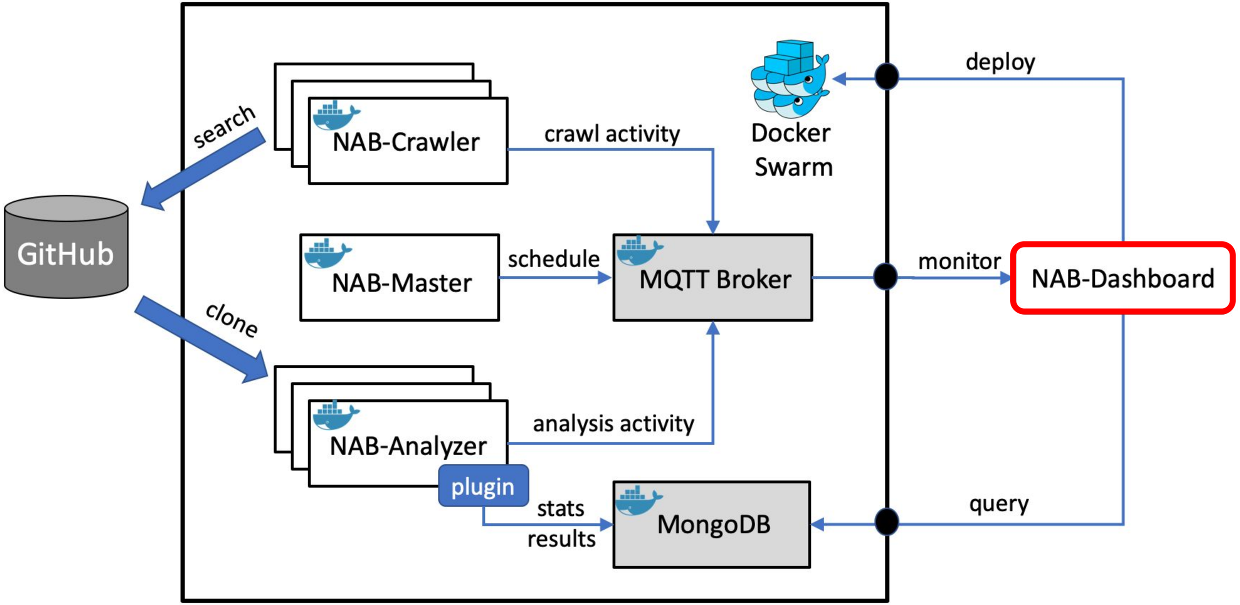
Plugin: mechanism to integrate *existing* DPA

NAB Architecture



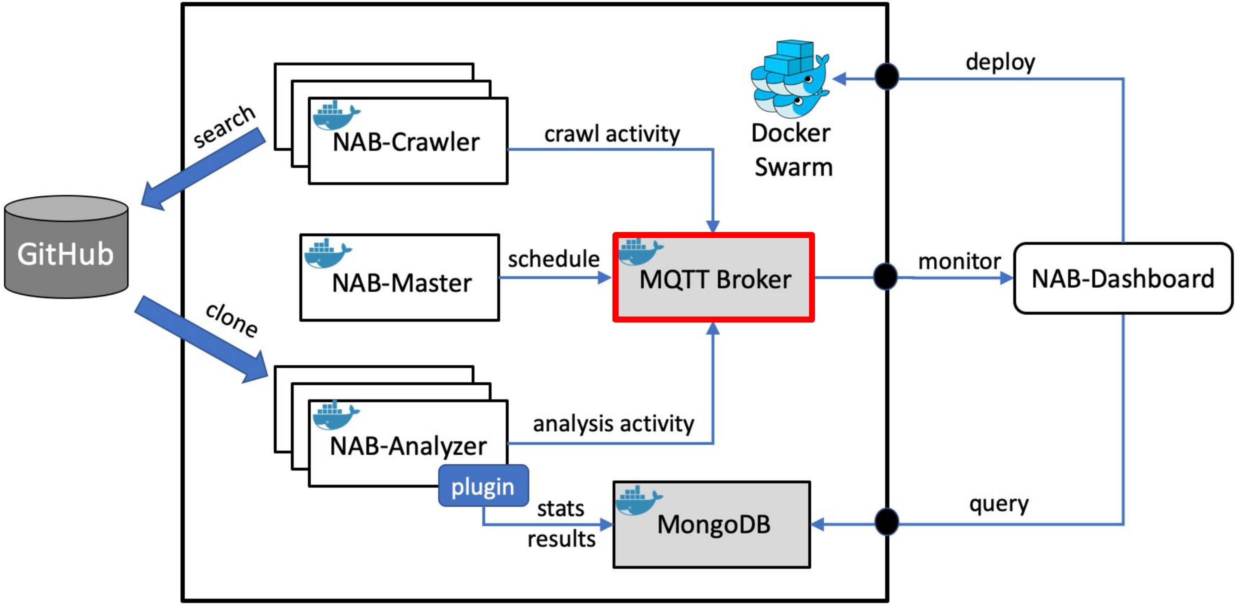
NAB-Master: orchestrates the distribution of crawling and DPA activities

NAB Architecture



NAB-Dashboard: handles deployment of NAB services (using Docker Swarm), allows users to monitor DPA progress

NAB Architecture



MQTT Broker: handles asynchronous communication through events (publish-subscribe communication protocol)

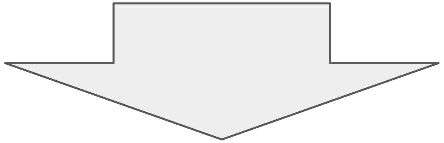
Case Studies

I Use of promises in Node.js applications

II JIT-unfriendly code patterns in Node.js applications

III Discovering Java and Scala task-parallel workloads for domain-specific benchmarking

- Codebase:
  - 5 years (2013-2017) of Node.js, Java, and Scala projects from GitHub



7.6 M projects

56 K projects

Case Study III: Discovering Task-parallel Workloads for Java and Scala

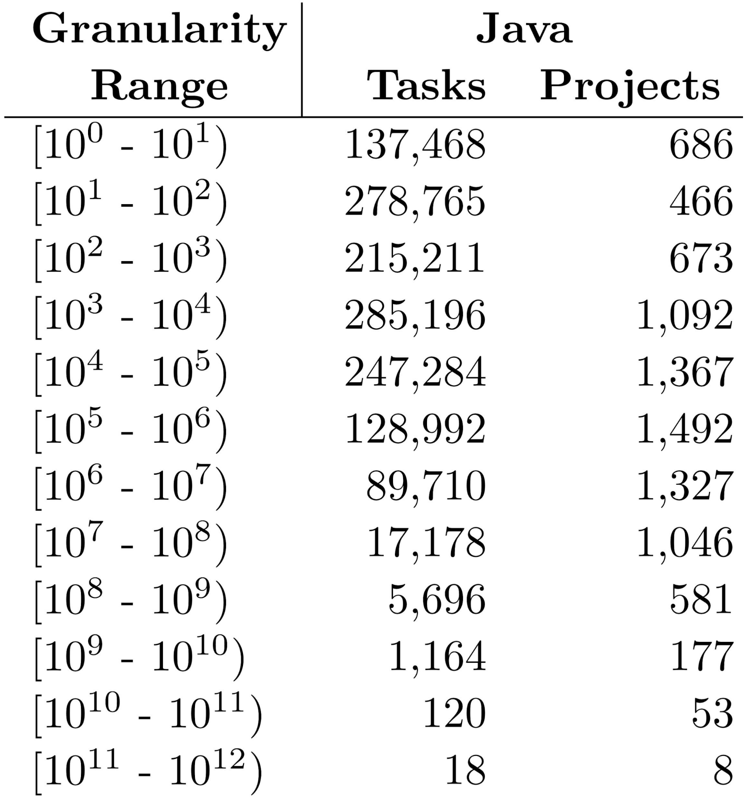
- **Goal:** Discover Java and Scala task-parallel workloads with diverse task granularity to analyze concurrency-related aspects
  - Granularity: number of bytecode instructions executed by a parallel task
- **DPA: tgp [1]** task granularity profiler
  - Collects granularity of all spawned tasks
    - Task = subtypes of **`Runnable`**, **`Callable`**, **`ForkJoinTask`**

[1] Rosà et al., Analyzing and Optimizing Task Granularity on the JVM. CGO 2018

Case Study III: Results (1/2)

Java workloads

- 1,769 projects successfully analyzed with task-parallel workloads
- Two workloads with granularities spanning all ranges
  - `https://github.com/rolfl/MicroBench`
  - `https://github.com/47Billion/netty-http`
- **Good candidates for benchmarking task execution in Java workloads**

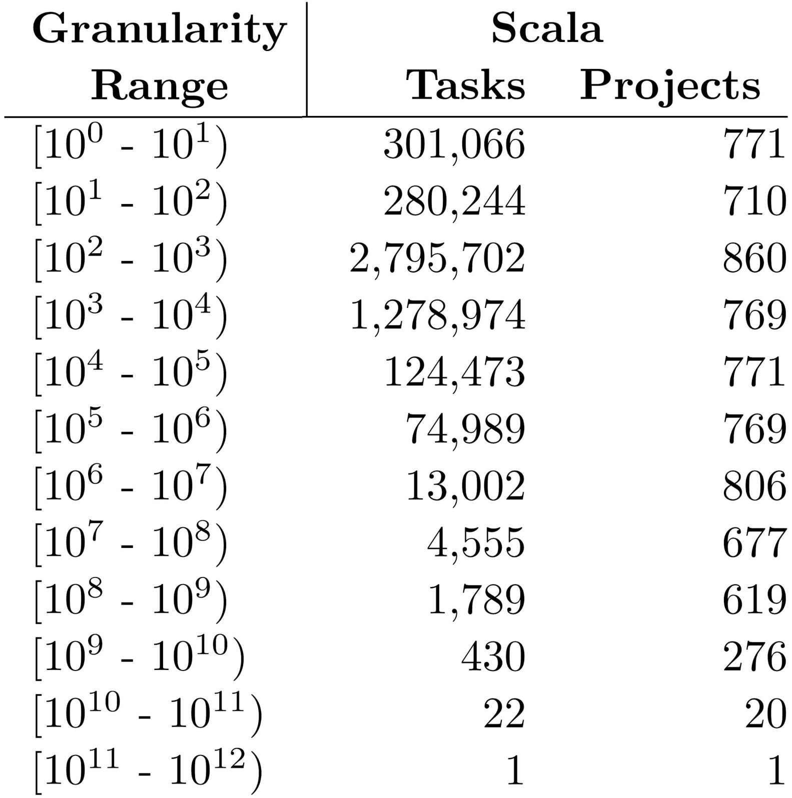


Granularity	**Java**	
Range	**Tasks**	**Projects**
$[10^0$ - $10^1)$	137,468	686
$[10^1$ - $10^2)$	278,765	466
$[10^2$ - $10^3)$	215,211	673
$[10^3$ - $10^4)$	285,196	1,092
$[10^4$ - $10^5)$	247,284	1,367
$[10^5$ - $10^6)$	128,992	1,492
$[10^6$ - $10^7)$	89,710	1,327
$[10^7$ - $10^8)$	17,178	1,046
$[10^8$ - $10^9)$	5,696	581
$[10^9$ - $10^{10})$	1,164	177
$[10^{10}$ - $10^{11})$	120	53
$[10^{11}$ - $10^{12})$	18	8

Case Study III: Results (2/2)

Scala workloads

- 860 projects successfully analyzed with task-parallel workloads
- Three workloads with granularities spanning all ranges
  - `https://github.com/iheartradio/asobu`
  - `https://github.com/TiarkRompf/virtualization-lms-core`
  - `https://github.com/ryanlsg/gbf-raidfinder`
- **Good candidates for benchmarking task execution in Scala workloads**



Granularity	**Scala**	
Range	**Tasks**	**Projects**
$[10^0 - 10^1)$	301,066	771
$[10^1 - 10^2)$	280,244	710
$[10^2 - 10^3)$	2,795,702	860
$[10^3 - 10^4)$	1,278,974	769
$[10^4 - 10^5)$	124,473	771
$[10^5 - 10^6)$	74,989	769
$[10^6 - 10^7)$	13,002	806
$[10^7 - 10^8)$	4,555	677
$[10^8 - 10^9)$	1,789	619
$[10^9 - 10^{10})$	430	276
$[10^{10} - 10^{11})$	22	20
$[10^{11} - 10^{12})$	1	1

Conclusions

- NAB: novel, distributed infrastructure
  for executing massive custom DPA on open-source code repositories
- Scalability, supporting Cloud-based deployment
- Fault-tolerance mechanisms
- Safety thanks to sandboxing of unknown code

- Presented case study:
  Discovering task-parallel workloads for Java and Scala
  - We identified five candidate workloads to benchmarking
    task parallelism on the JVM

NAB

- Evaluation version at
  http://research.upb.edu/NAB/nab-artifact.tgz

- Contact:
  Walter Binder
  walter.binder@usi.ch

Thanks for your attention