Fixr App Builder Farm

CU Boulder Muse Team

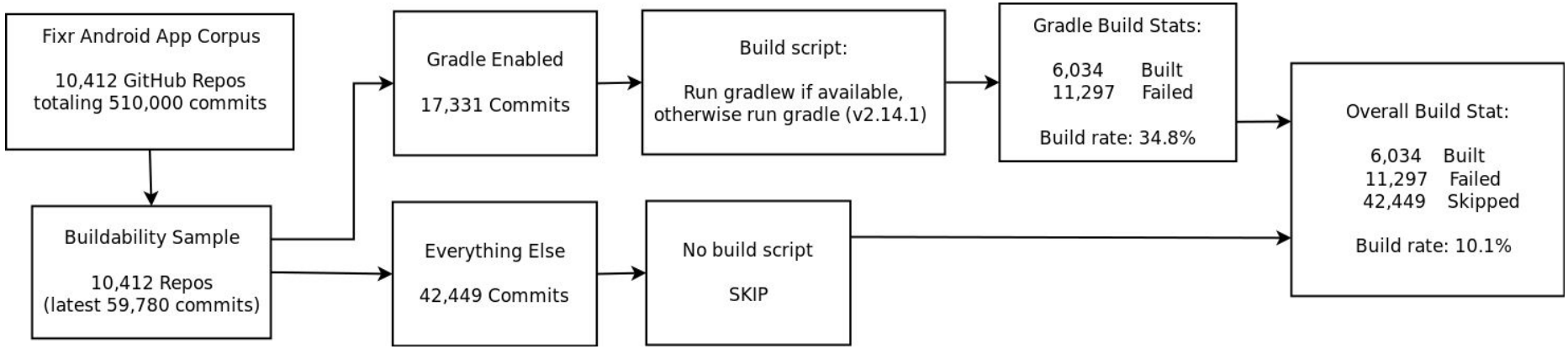
App Building: Prelude to Understanding Android Bugs

- Before we analyze Android apps for bugs….
  - Generating dynamic traces
  - Analyzing/instrumenting JVM bytecodes
- We've got to build them. Our corpus:
  - 510,000k commits from 10,412 GitHub **Android** Repositories
  - Extracted from GitHub search API: "Android" in name, description, readme
  - Plus some post filtering (e.g., > 5 watchers, > 40 kbytes)

Building Android Apps En Masse

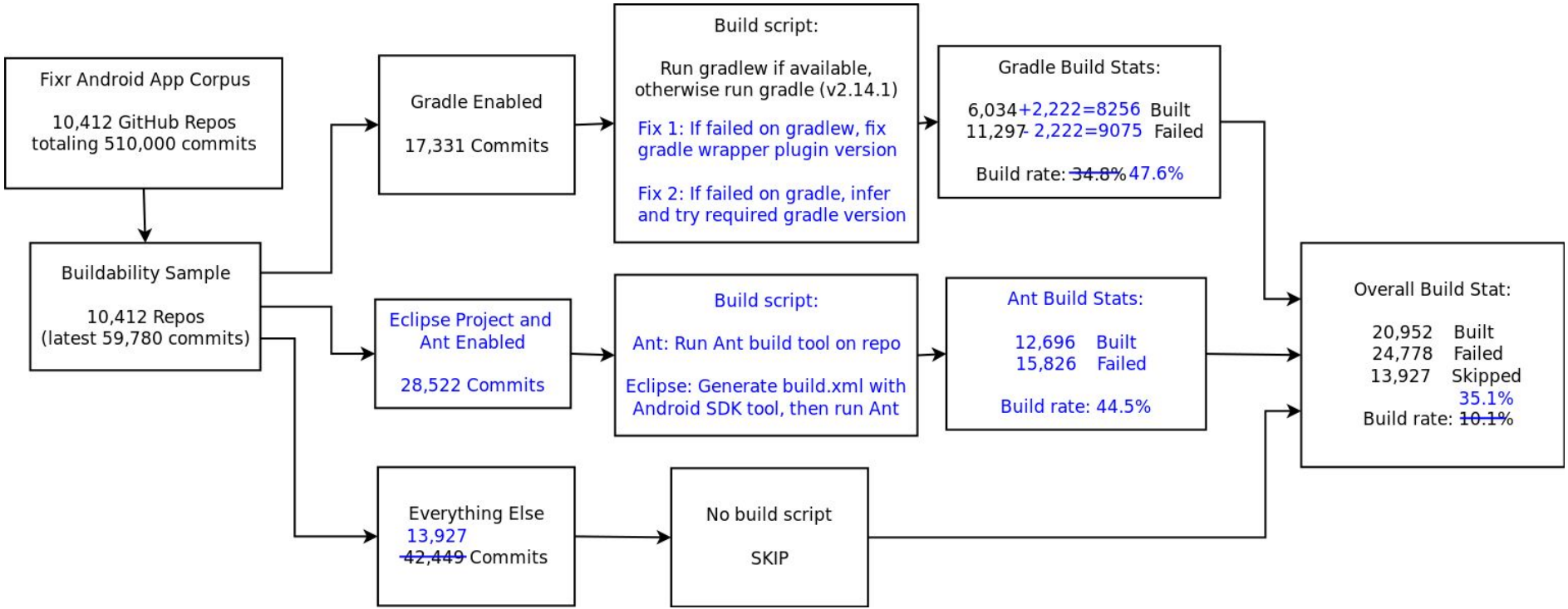
- A challenging engineering problem:
  - Diversity (Gradle, Ant, Eclipse projects, even plain garbage..)
  - Bad practices are rampant (e.g., hard coded configs, no gradle wrapper)
  - Long build / download times (up to 4-5 minutes per commit)
  - We want to analyze **all** commits! Not just “head”.
- Our ongoing efforts to mitigate this problem:
  - Enhancing our Android app building automation script.
  - From standalone scripts, to full-stack cloud compute and Web API service

Our App Building Script v1.0 (May 2016)



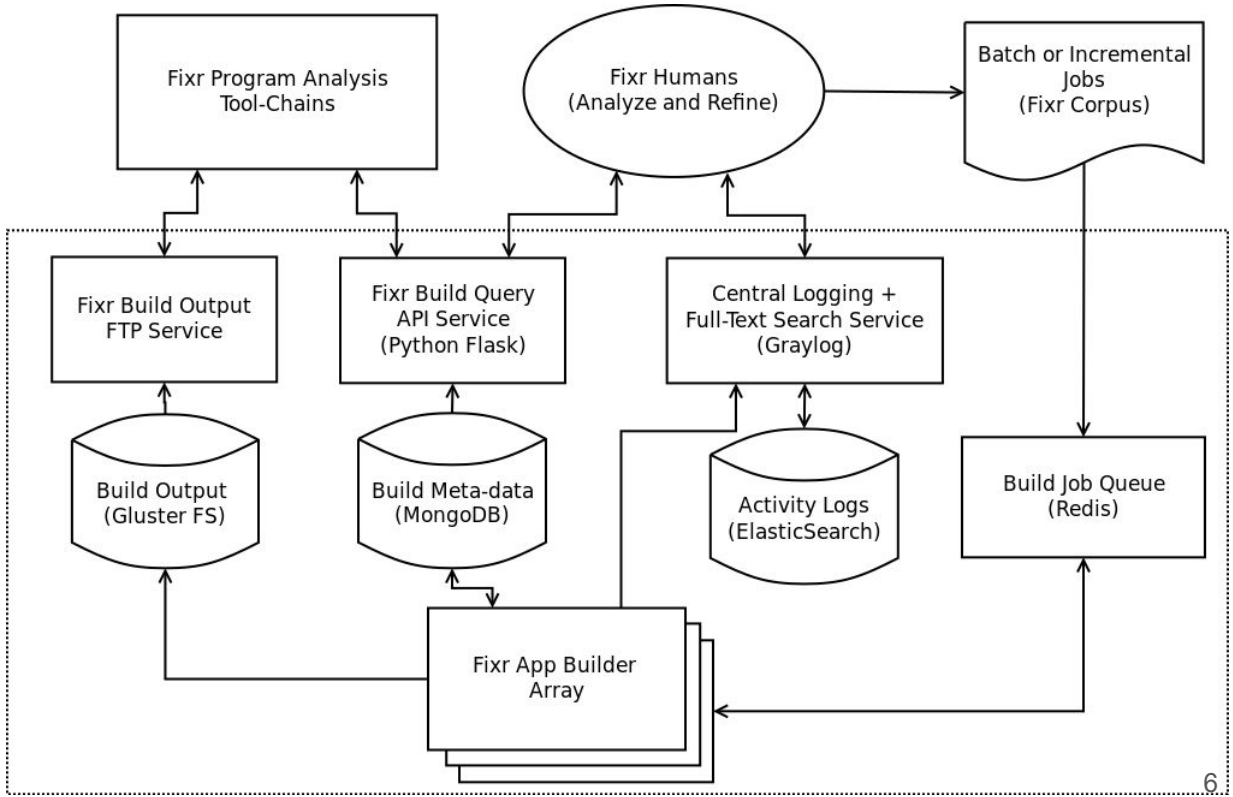
- Buildability sample: 59,780 commits = Latest 8 (up to) commits of each 10,412 repositories.
- To understand why so many failed we...
  - Manually ‘eye-balled’ some error logs.
  - Applied K-means clustering on relevant fragments build error logs.

Our App Building Script v2.0 (Today)



The Fixr App Builder Farm

- Addresses a number of issues:
  - Scale with cloud compute
  - Manage BIG build data
  - Support data query and extraction
  - Tool-chain integration
- Current Running on:
  - CU OpenStack
  - Front-end Services: 1 VM
  - Builder Array: 2 VMs
  - Quad-Core 32GB Instances



Summing it up

- Increased build rates: 10.1% => 35.1%

- Scaling up: Standalone scripts => Full-stack cloud service

Our Next Steps

- Integrating our analysis tools with our app builder service.
- Increase build rates:
  - Understand why Gradle and Ant still failed 55% (roughly) of the time.
  - Propose and implement heuristics to ‘fix’ broken/bad Git repos.
- Beyond our corpus:
  - Integrating with Leidos’ corpus.
  - Active focused crawl on GitHub for more Android repositories.
- Scaling cloud services:
  - Elastic computing with OpenStack Nova and Docker swarm
  - Scaling build data warehouse with object storage (OpenStack Swift)

Fin

Questions? Ask away now or offline!

Facing similar challenges? Let’s exchange notes & cheat sheets!