Parallel UI testing in Android

Jean

Growth Session #26 - July 25-26 2019

Previously... on Nimble Growth

Automating the release process; repetitive clerical tasks

- Creating proper gitflow branches
- Updating version code
- Phraseapp pull
- Localization diff, etc.
- Sending release email

Challenges

The CI/CD Jenkins pipeline for Android Red Planet

- Pipeline execution - 2hrs on average
- 2 pipelines execution per pull request
- Server only executes max of 2 in parallel, **but not** within the same pipeline
- Waiting time of up to 8 hrs is not uncommon

UI Tests amount for ¾ of the time taken

Goals

Deliver much faster

- Running tests in parallel on multiple devices **within** the same pipeline
- Load balance UI tests between ***n*** number of emulators/devices, i.e. the more emulators/devices, the faster the execution

Test Sharding

The traditional (naive) approach:

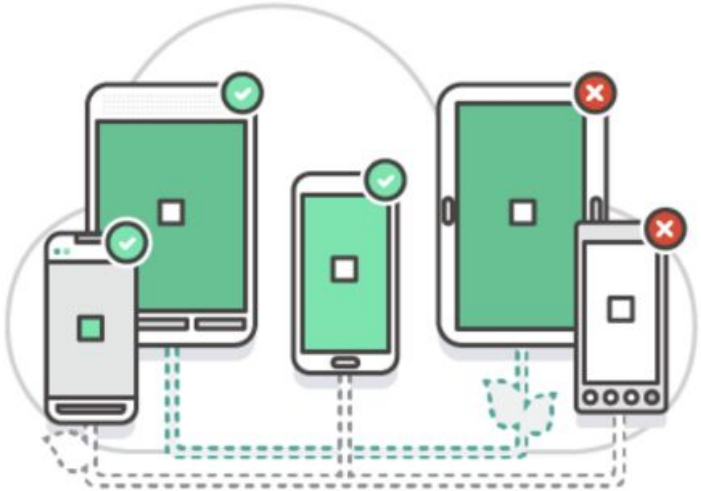
- Sharding allows to evenly divide up your tests into groups
- Doesn't take into account how long a particular test takes

```
adb -s {device} shell am instrument -w -e numShards 4 -e shardIndex 1
```

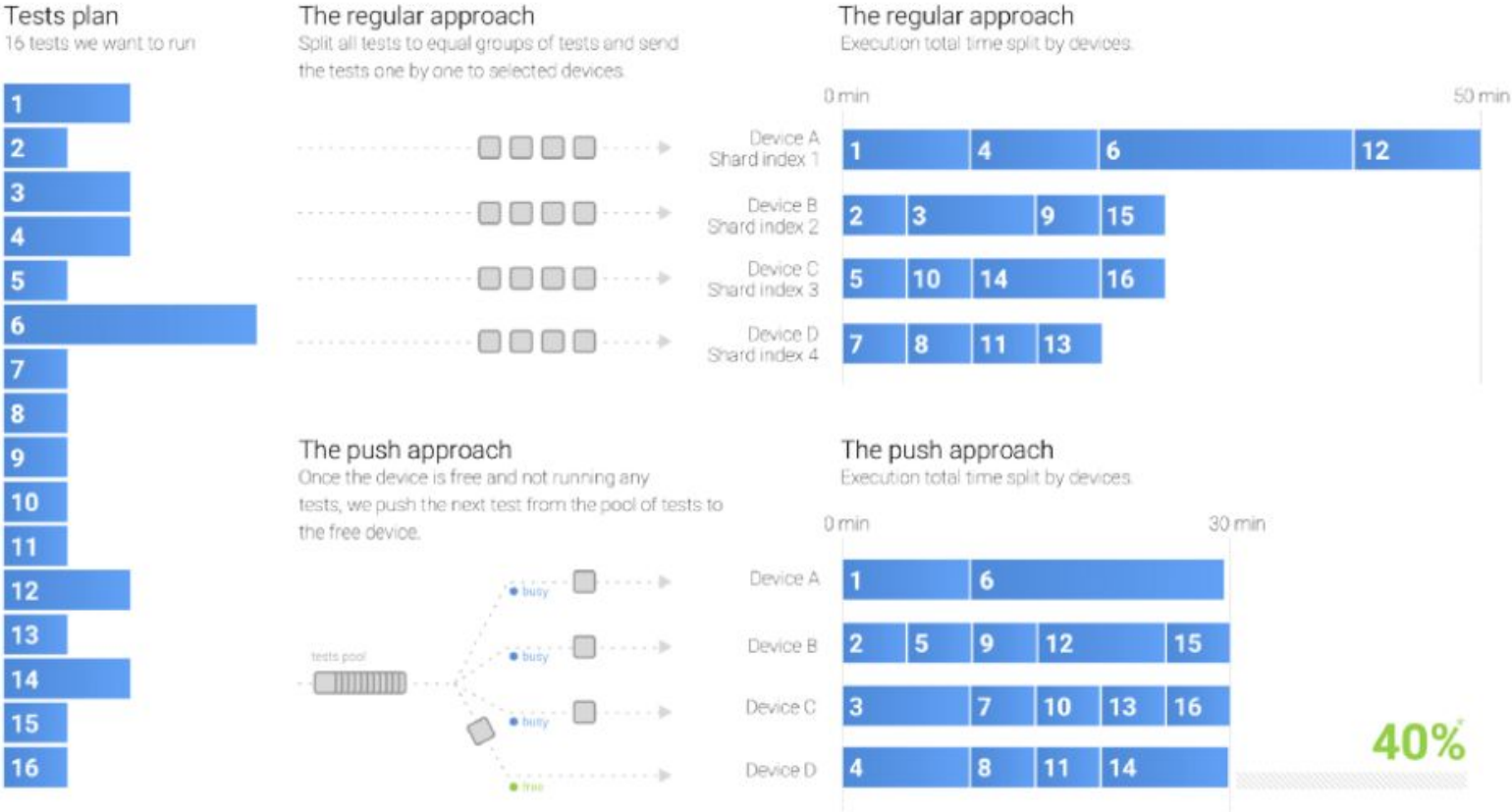
- **numShards** = number of groups we ask Android to divide all our tests
- **shardIndex** = the group index we want to run on specific {**device**}

Load Balancing (as simple implementation)

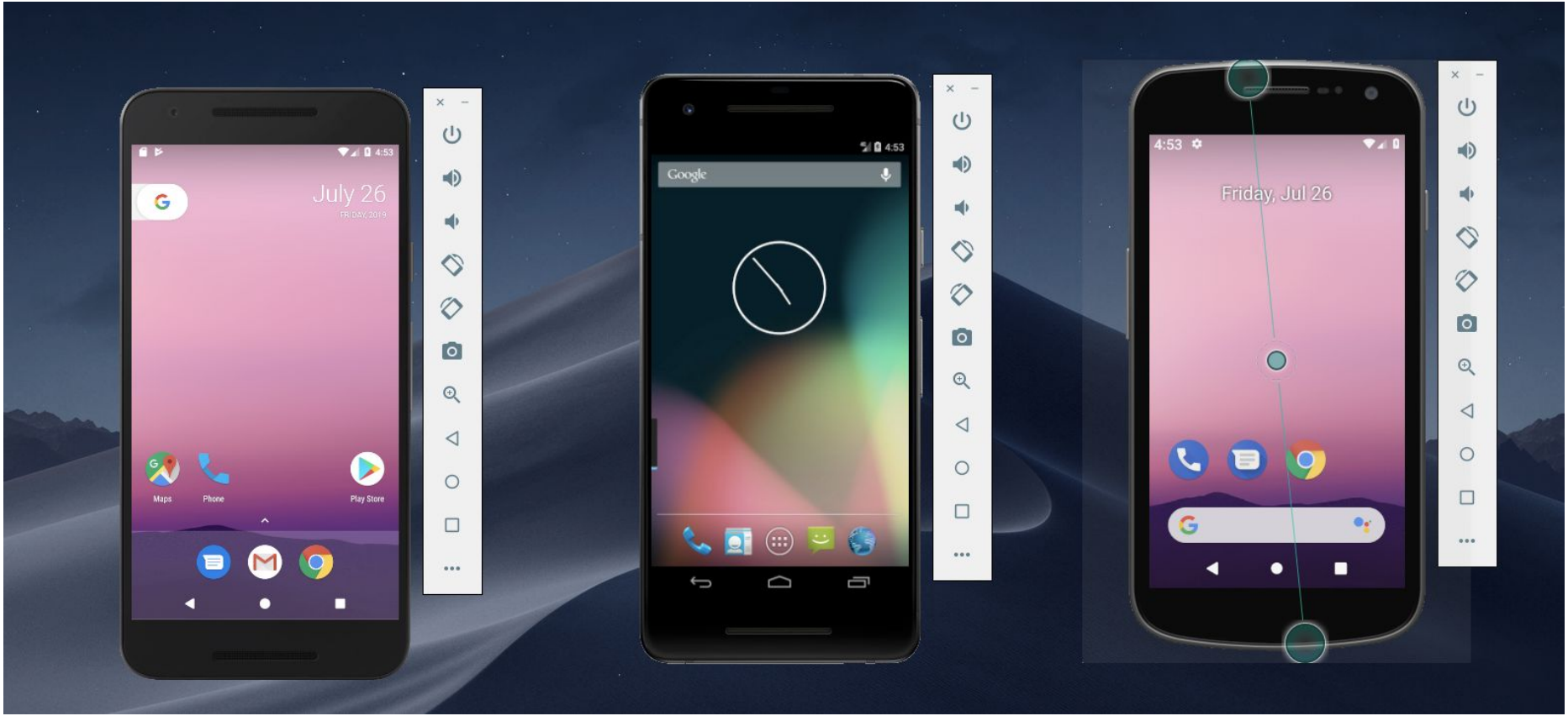
- Build and install on all connected devices; app and test apk
- Get list of tests from test apk
- For each test, run next on available/idle device



Traditional Sharding VS Load balancing

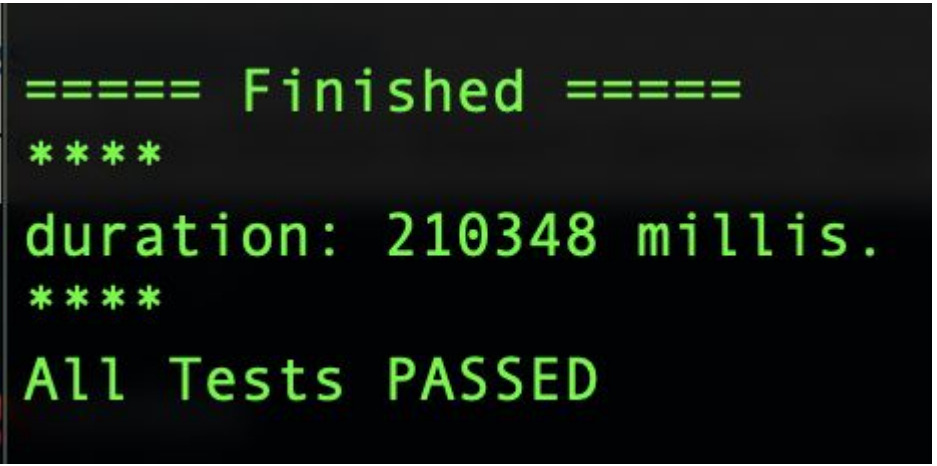


Demo

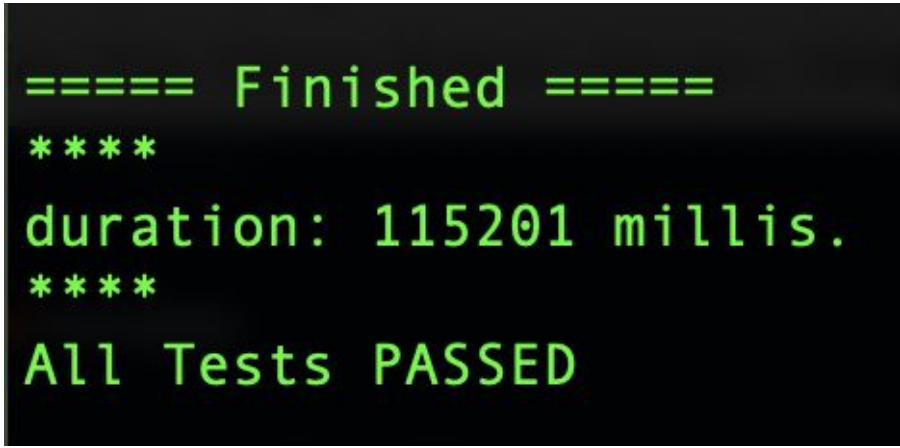

4:53
July 26
FRIDAY, 2019
Maps
Phone
Play Store
4:53
Google
4:53
Friday, Jul 26

Performance

In sequence



In parallel



When ran using load balancing, the UI tests ran almost **twice as fast** as those in parallel.

Achievement and Progress

- Successfully achieved load balancing
  - Faster pipeline
  - Scalable solution, i.e. the more devices, the faster
  - but...
- This approach requires native UI tests, i.e. espresso, UIAutomator
- Red Planet Android tests are not native

Next Steps

- Ongoing work required to slowly convert from rspec tests to native Android tests
- Implement control functionality such as running only a subset of UI tests, e.g. critical VS non-critical
- Experiment with connecting real devices to CI server
- Management of emulators delegated out of the pipeline

Conclusion

- Automating the clerical tasks for a release in previous Nimble Growth
- Cutting down UI tests execution time
- Keep identifying ways to optimize the CI pipeline

The CI pipeline requires constant optimization

Thanks!

Contact Nimble

nimblehq.co

hello@nimblehq.co

Bangkok

399 Interchange 21 Sukhumvit Road, Unit #2402-03, Klong Toei, Wattana, Bangkok 10110, Thailand

Singapore

28C Stanley St, Singapore 068737

Hong Kong

20th Floor, Central Tower

28 Queen's Road, Central, Hong Kong

nimble