QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2018

《BUCK在大规模iOS开发中的应用实践》

演讲者 / 陈坤

主办方 Geekbang. 极客邦科技 InfoQ

Airbnb iOS Stats

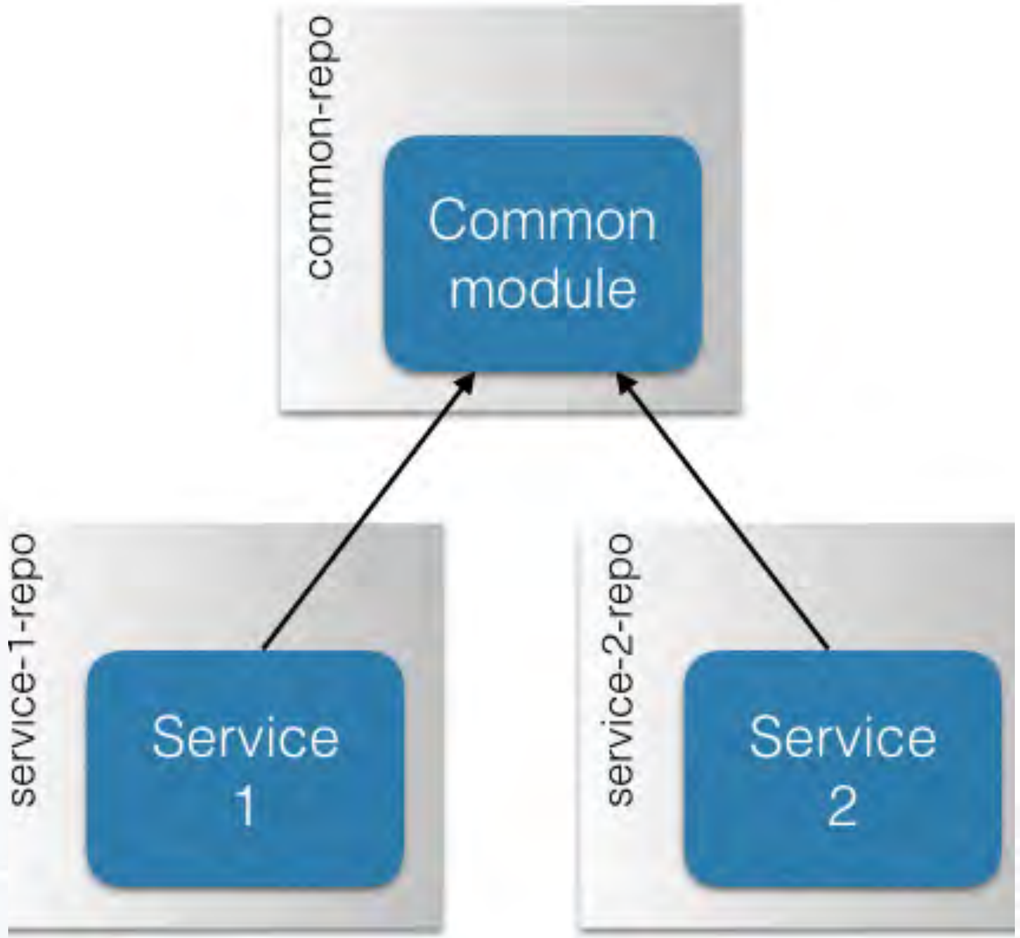
- 80 commits/day to master (PRs are squashed)
- 500+ builds/week
- 71 iOS contributors last month

Airbnb iOS Stats

- Multi apps
- Monorepo
- 900k lines of code
  - 67% Swift
  - 25% Javascript
  - 8% Objective-C
- 80 internal frameworks + 25 Cocoapods + 2 Carthage

Main Challenges

- Multi-Repos
  - Hard to refactor
  - Non-atomic changes, complex versioning
  - Cocopods encourages multi repos.

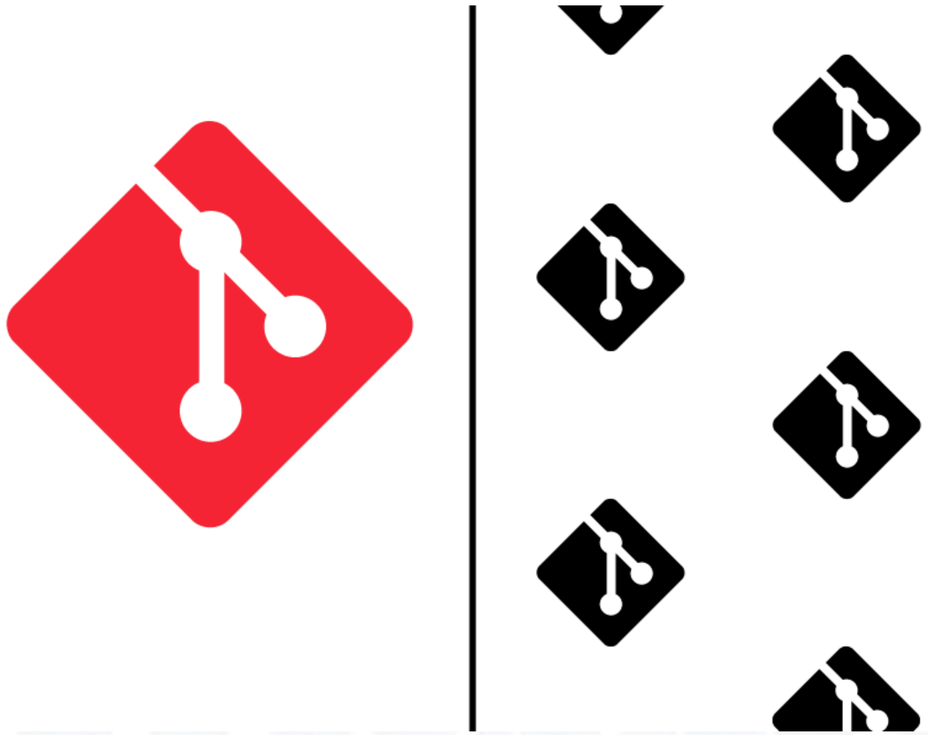


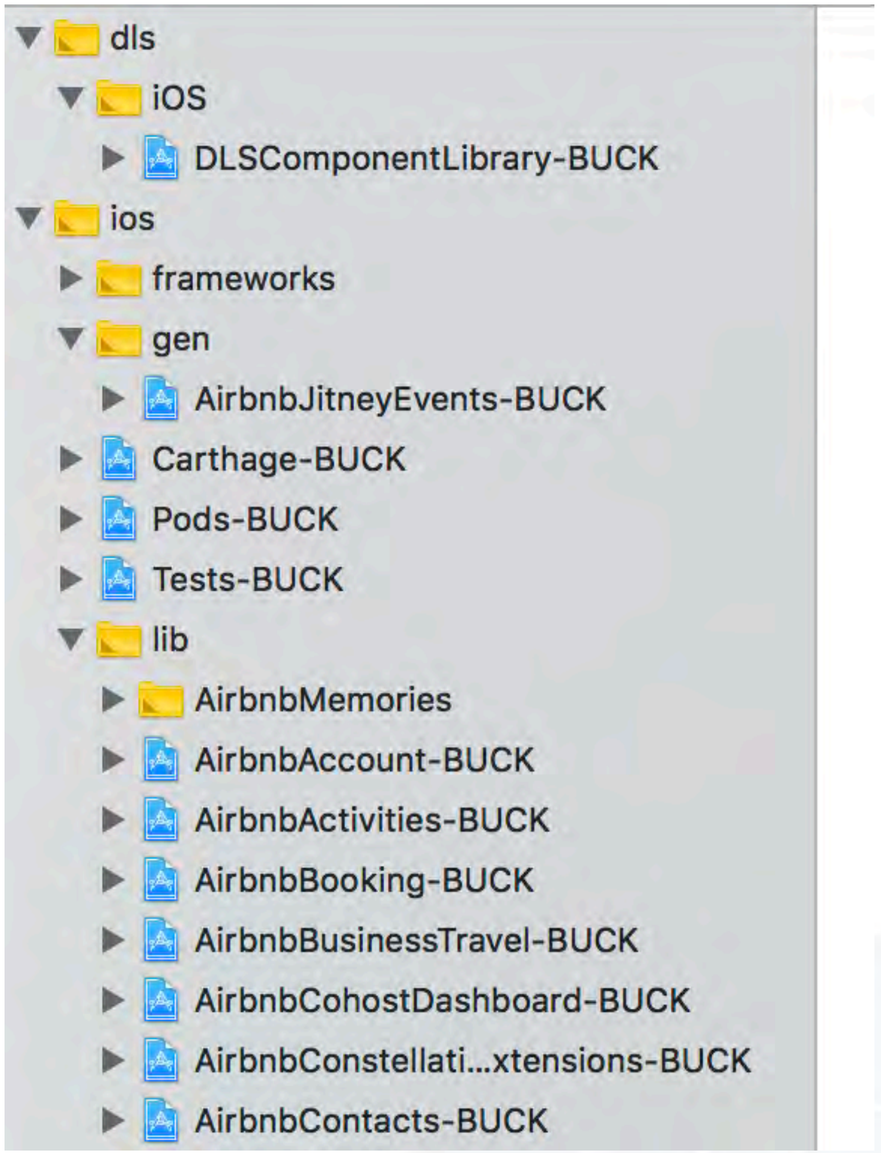
Main Challenges

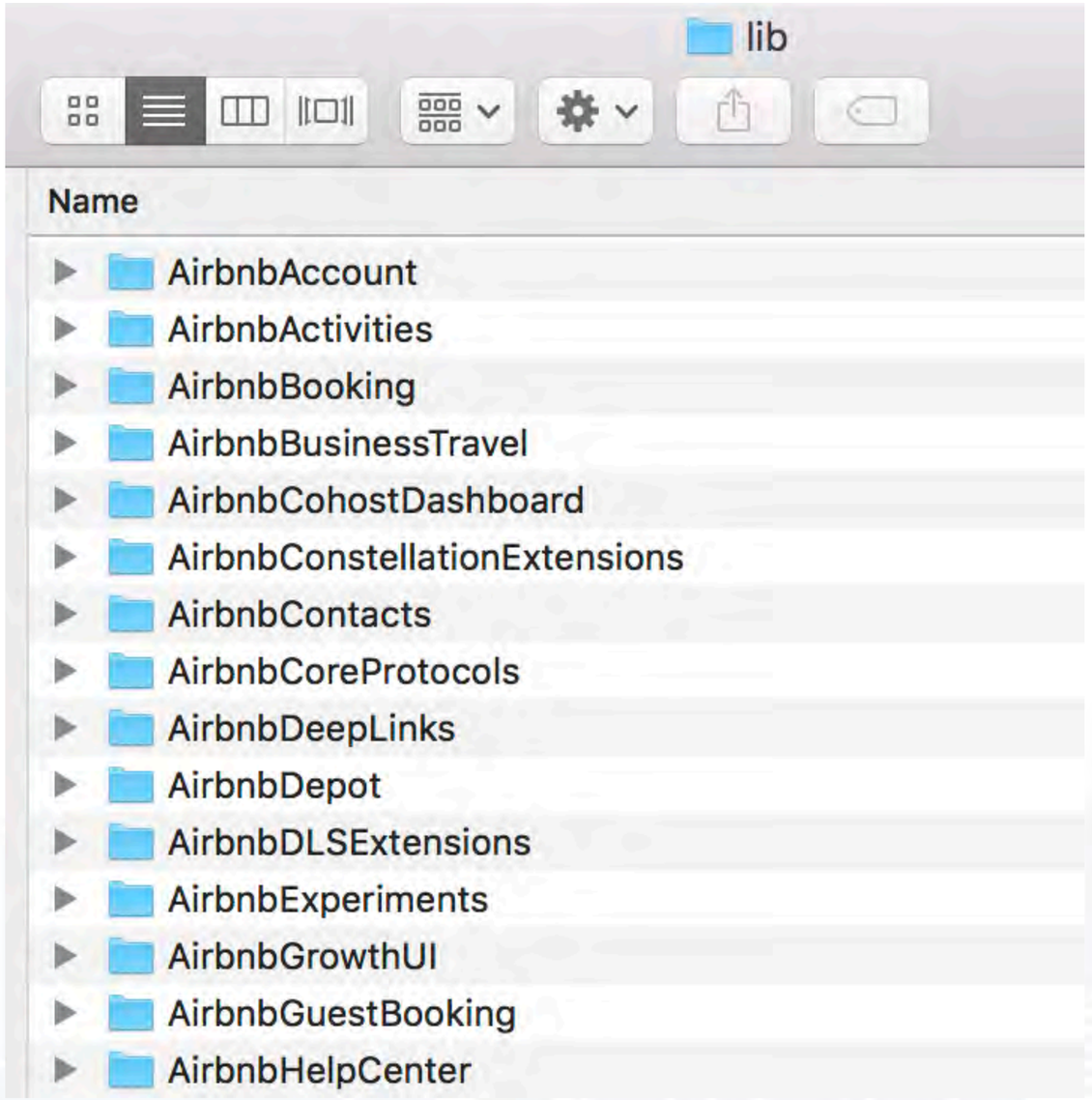
- Xcode project management
  - Resolving .xcproj conflicts.
  - Hard to maintain dependencies.
  - Hard for modernization.
  - Hard to change build settings.
- CI build/test performance

Airbnb change to use Monorepo

- Unified versioning, one source of truth.
- Atomic changes.
- Flexible team boundaries and code ownership.
- Large scale refactoring, codebase modernization.
- **Works for BUCK.**




dls
iOS
DLSComponentLibrary-BUCK
ios
frameworks
gen
AirbnbJitneyEvents-BUCK
Carthage-BUCK
Pods-BUCK
Tests-BUCK
lib
AirbnbMemories
AirbnbAccount-BUCK
AirbnbActivities-BUCK
AirbnbBooking-BUCK
AirbnbBusinessTravel-BUCK
AirbnbCohostDashboard-BUCK
AirbnbConstellati...xtensions-BUCK
AirbnbContacts-BUCK


lib
Name
AirbnbAccount
AirbnbActivities
AirbnbBooking
AirbnbBusinessTravel
AirbnbCohostDashboard
AirbnbConstellationExtensions
AirbnbContacts
AirbnbCoreProtocols
AirbnbDeepLinks
AirbnbDepot
AirbnbDLSExtensions
AirbnbExperiments
AirbnbGrowthUI
AirbnbGuestBooking
AirbnbHelpCenter

Use Buck as the build system

Build Tool Landscape

- Cocoapods
- Carthage
- BUCK
- xctool
- xcodebuild
- Swift Package Manager
- …

Cocoapods

- Airbnb now only uses Cocoapods for 3rd party libraries.
  - We used it for internal UI frameworks.
- Why it doesn't work us:
  - Slow to resolve, complex interdependencies.
  - Not designed for monorepo.

BUCK helps

- Designed for monorepo.
- Easy and clean dependency management.
- Encouraging modernization.
- Generate project for Xcode.
- More important: Faster build/test.

Why using BUCK

- Build Time
  - Building with multi-threading
  - Cache aggressively
- App Size
  - Merging modules into single binary
  - Saving spaces in keep multiple copies of same resource
- Dev Efficiency
  - Reducing effort in maintaining Cocoapods
  - Creating new module easier by adding BUCK file
  - Running tests with one line command
  - Review diffs in Xcode project changes

BUCK File

- Every framework/app has a BUCK file.
  - Buck tool is written in Java, but BUCK file is **python script.**
- Define build rules and build targets.
- Build targets
  - What and how to compile.
  - Specify dependencies (transitive)

Example of BUCK file

BUCK

```
apple_library(
  name = 'AirbnbStories',
  swift_version = "4",
  visibility = ['PUBLIC'],
  exported_headers = glob([
    'AirbnbStories/**/*.h',
  ]),
  srcs = glob([
    'AirbnbStories/**/*.m',
    'AirbnbStories/**/*.swift',
  ]),
  deps = [
    '//ios/lib/AirbnbAccount:AirbnbAccount',
    '//ios/lib/AirbnbDLSExtensions:AirbnbDLSExtensions',
    '//ios/lib/AirbnbExperiments:AirbnbExperiments',
    '//ios/lib/AirbnbModelExtensions:AirbnbModelExtensions',
    '//ios/lib/AirbnbModels:AirbnbModels',
    '//ios/lib/AirbnbNavigation:AirbnbNavigation',
    '//ios/lib/AirbnbObjectStore:AirbnbObjectStore',
```

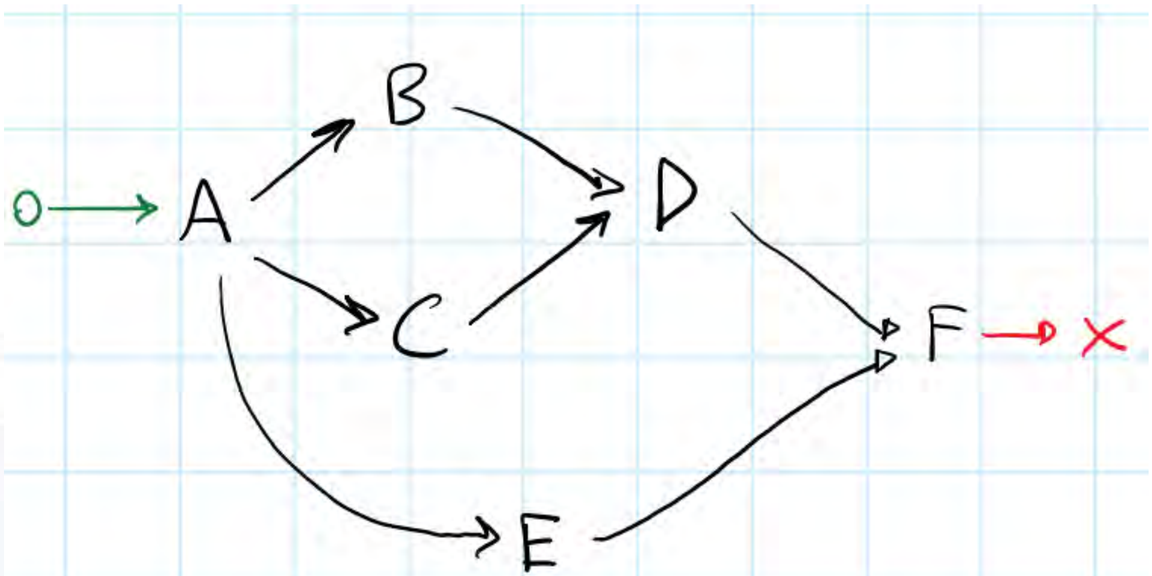
Power of python script

```
def AirbnbApp_rule(build):
  non_empty_build_type = build or 'Debug'
  apple_binary_name = get_name_with_build_type('AirbnbBinary', build)
  apple_bundle_name = get_name_with_build_type('AirbnbApp', build)
  apple_package_name = get_name_with_build_type('AirbnbPackage', build)

  apple_binary(
    name = apple_binary_name,
    visibility = ['PUBLIC'],
    configs = airbnb_binary_config,
    entitlements_file = build_configs[non_empty_build_type]['AIRBNB_ENTITLEMENTS'],
    srcs = glob([
      'BuckSupportFiles/main.m',
    ]),
```

BUCK DAG

- Explicit dependency, no hidden state.
- Dependencies are transitive.
- DAG powers the build speed and cache.



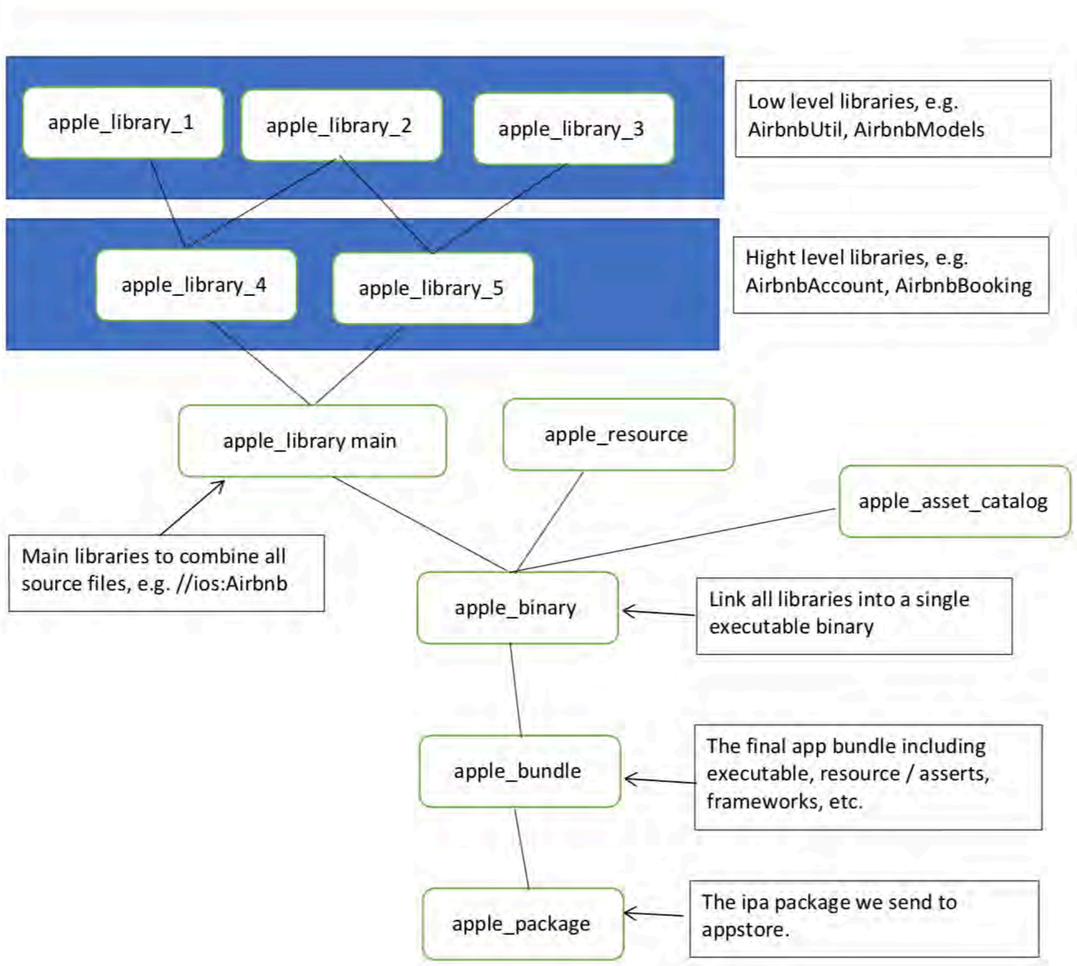
Faster Build

- BUCK is deterministic, output only determined by declared input.
- Cache
  - Disk cache.
  - Http cache.
    - https://github.com/uber/buck-http-cache
- Parallelize build/test.

Buck Build Demo

```
Running Buck build.
/Users/kun_chen/.cache/buck/binary/a25f5bab92ccef97482e3109075c7c7353b40763/bin/buck build //ios:Airbnb --config cxx.cflags='-fmodules -fobjc-arc -D BU
 --config cxx.cxxflags='-fobjc-arc -std=c++14 -D DEBUG -g' --config swift.compiler_flags='-DBUCK -whole-module-optimization -DDEBUG -enable-testing -g
.default_debug_info_format_for_binaries=DWARF
Building... 03:06.7 min (82%) 522/631 jobs, 522 updated, 8.2% cache miss
 - //ios/lib/Walle:Walle#apple-swift-compile,iphonesimulator-x86_64... 6.3 sec (running swift compile[6.3 sec])
 - //ios/lib/FeatureGarden:FeatureGarden#apple-swift-compile,iphonesimulator-x86_64... 1.6 sec (running bash[0.0 sec])
 - IDLE
 - //ios/lib/AirbnbDeepLinks:AirbnbDeepLinks#apple-swift-compile,iphonesimulator-x86_64... 0.2 sec (running swift compile[0.1 sec])
 - //ios/lib/ConstellationHomesGuestTeam:ConstellationHomesGuestTeam#apple-swift-compile,iphonesimulator-x86_64... 6.8 sec (running swift compile[6.8 s
 - //ios/lib/AirbnbSunset:AirbnbSunset#apple-swift-compile,iphonesimulator-x86_64... 6.8 sec (running artifact_compress[0.1 sec])
 - IDLE
 - //ios/lib/ConstellationMagicalTripsTeam:ConstellationMagicalTripsTeam#apple-swift-compile,iphonesimulator-x86_64... 7.5 sec (running swift compile[7
```

How Buck builds iOS app



How apple_library works

1. Input: *.h. Output: Module.hmap.

2. Input: *.swift, Module.hmap. Output: Module.o, Module.swiftmodule, Module-Swift.h

3. Input: *.m, Module.hmap, Module-Swift.h. Output: *.o for each *.m.

4. Input: *.o, Module.o. Output: libModule.a.

BUCK Commands

- build
  - *buck build //ios:AirbnbPackage_Alpha*
  - *buck build AirbnbAlpha*
- Generate project
  - *buck project //ios:Airbnb*
- Query
  - *buck query “deps(//ios:Airbnb)”*

Airbnb Dependency Graph

TODO… Attach an image here

Challenges in Airbnb

Mix Languages 1

- Auto import objective-c to Swift
  - -import-underlying-module doesn’t work
  - Explicitly pass bridging headers.

Mix Languages 2

- @import doesn't work in objective-C due to no *module.modulemap*.
  - Use #import in our own codebase.
  - For generated **-Swift.h*, use customize script to do transform.
  - https://github.com/airbnb/buck/commit/93453a5730445c0dd799872b6e8bba49233430d7

Mix Languages 3

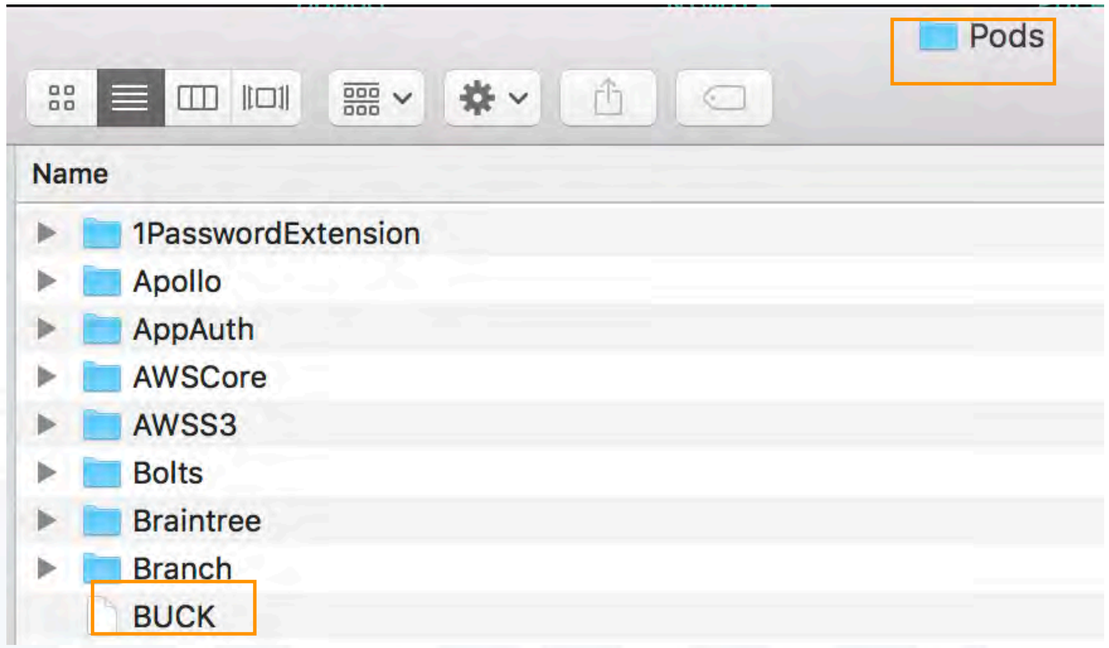
- Bridging header can’t import headers under current module.
  - Update BUCK to include the generated header map for current library.
  - https://github.com/facebook/buck/pull/1164

```
apple_library(
    name = 'Greeter',
    bridging_header = 'bridging-header.h',
    headers = glob([
        '*.h',
    ]),
    srcs = glob([
        '*.m',
        '*.swift',
    ]),
    frameworks = [
        '$SDKROOT/System/Library/Frameworks/Foundation.framework'
    ],
)
```

Can’t #import <Greeter/Greeter.h>
in the bridging header

Work with Cocopods

- Still use Cocopods to install packages.
- Manually create and maintain a BUCK file.
- We can do better:
  - Hook into Podfile to generate BUCK file automatically.
  - But we are moving off from Cocoapods.



Work with Cocopods

BUCK

```
177     ]),
178   )
179
180   apple_library(
181     name = 'libPhoneNumber_iOS',
182     visibility = ['PUBLIC'],
183     exported_headers = glob([
184       'libPhoneNumber-iOS/**/*.h',
185     ]),
186     srcs = glob([
187       'libPhoneNumber-iOS/**/*.m',
188     ]),
189   )
```

Untitled-1

```
21    srcs = glob([
22      'AirbnbUtil/**/*.m',
23      'AirbnbUtil/**/*.swift',
24    ]),
25    deps = [
26      '//ios/lib/AirbnbNetworking:AirbnbNetworking',
27      '//ios/lib/AirbnbPhrases:AirbnbPhrases',
28      '//ios/lib/AirbnbSwiftExtensions:AirbnbSwiftExtensions',
29      '//ios/Pods:libPhoneNumber_iOS',
30      '//ios/Pods:PromiseKit',
31      '//ios/Pods:SDWebImage',
32    ],
33    tests = [
34      ':AirbnbUtilTests',
35    ],
```

Others

- Support *prebuilt_cxx_library* in BUCK project generation
- Support umbrella headers
  - Cocoapod generate umbrella headers named as *Module-umbrella.h*
  - Buck assuming all module header named as *Module.h*
  - *swiftc* compliant if a public header file is not included in module header

Resource

- https://github.com/airbnb/buck/tree/airbnb-modular-external-use
- Sample: https://github.com/airbnb/BuckSample

Migration Process

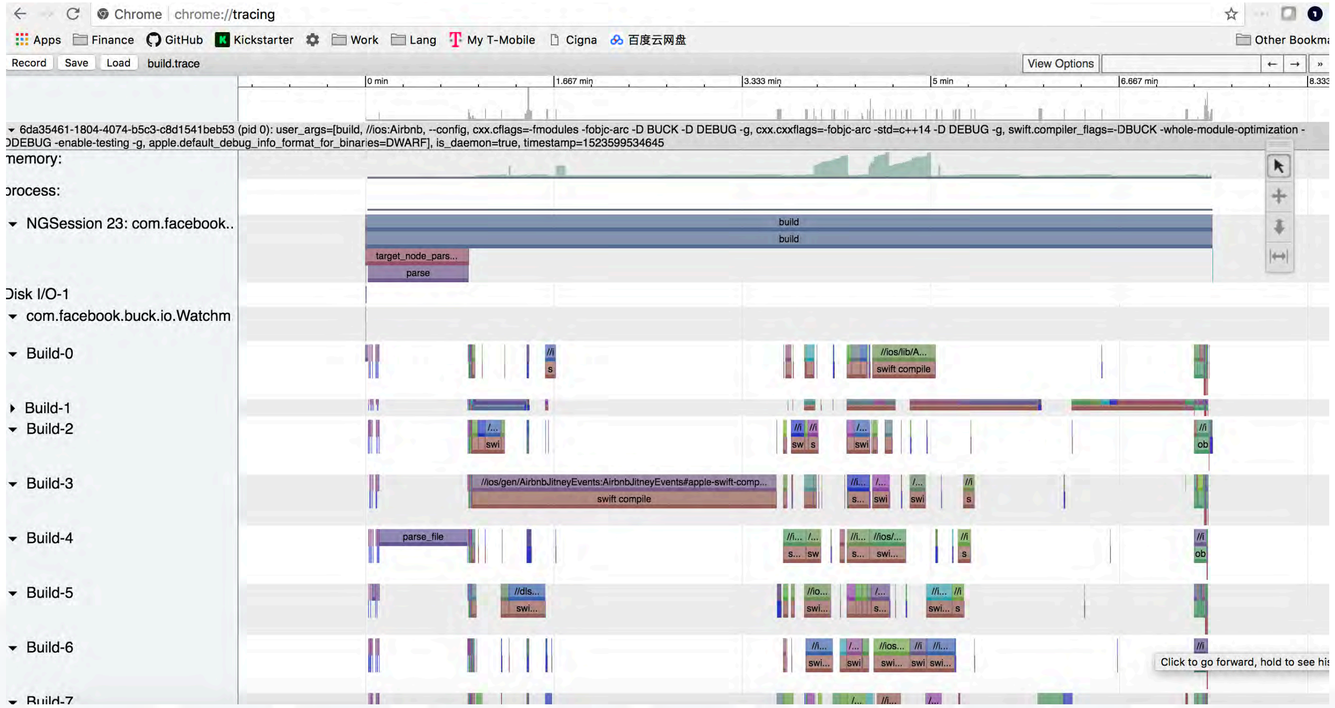
- Make Cocoapods support BUCK
- Create BUCK file for each library/module one by one.
- Dual build system in CI
  - Use both xcodebuild and BUCK in branch build (including tests) and Alpha build.
- Dual local development workflow
  - Xcode workflow work as normal
  - Use *buck project* to generate Xcode projects.
  - In appropriate timing, remove Xcode projects and stick to Buck workflow

Results

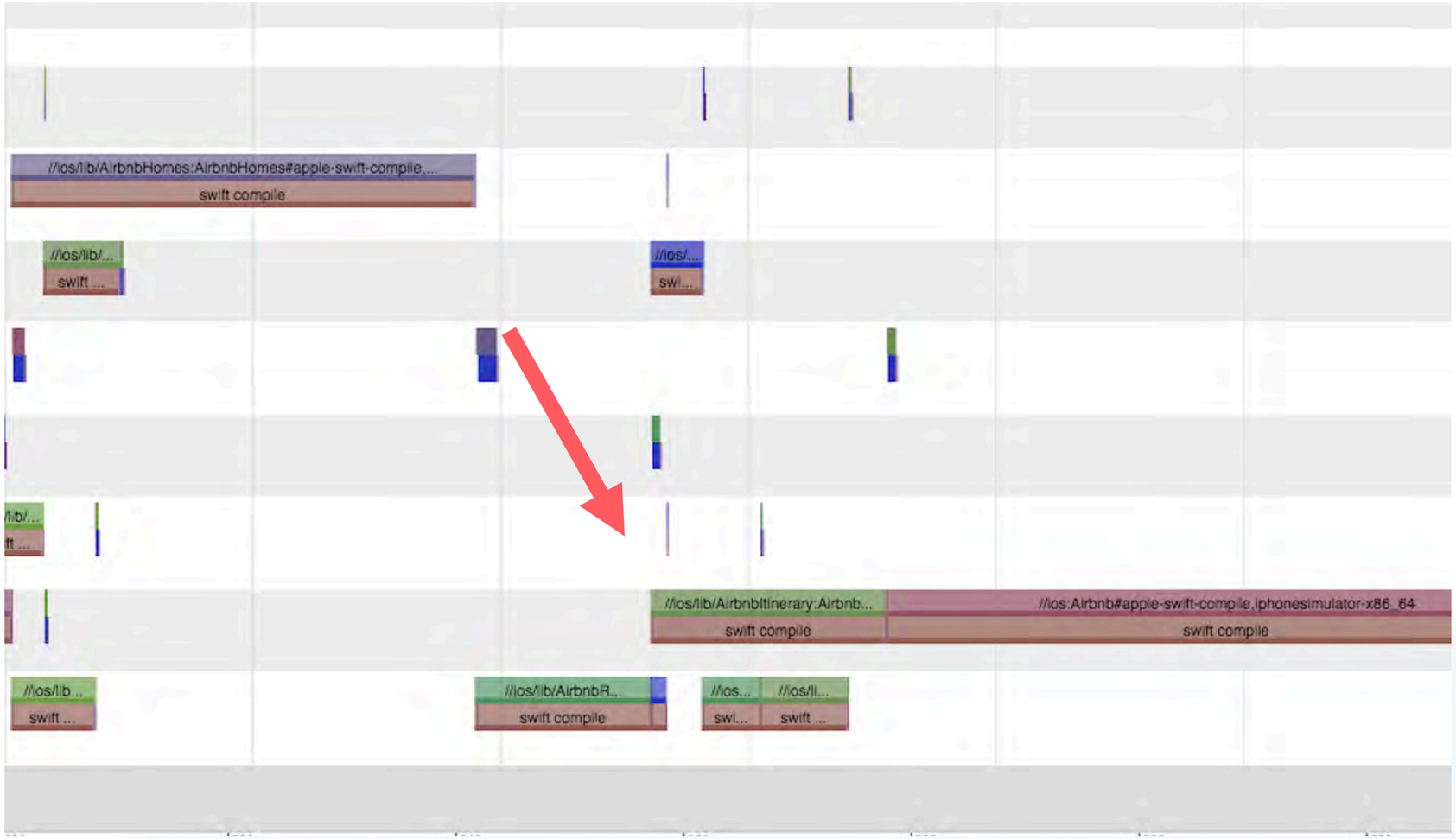
- The whole app can be built using Buck.
- Buck generated Xcode workspace checked into codebase and can work.
- Use BUCK in Alpha build for employee testing.
- **50%** faster CI builds, **30%** smaller app size.

Visualize Build Process

- After buck build, tracking logs are in *buck-out/log/traces/*
- Chrome tracing tool: *chrome://tracing*

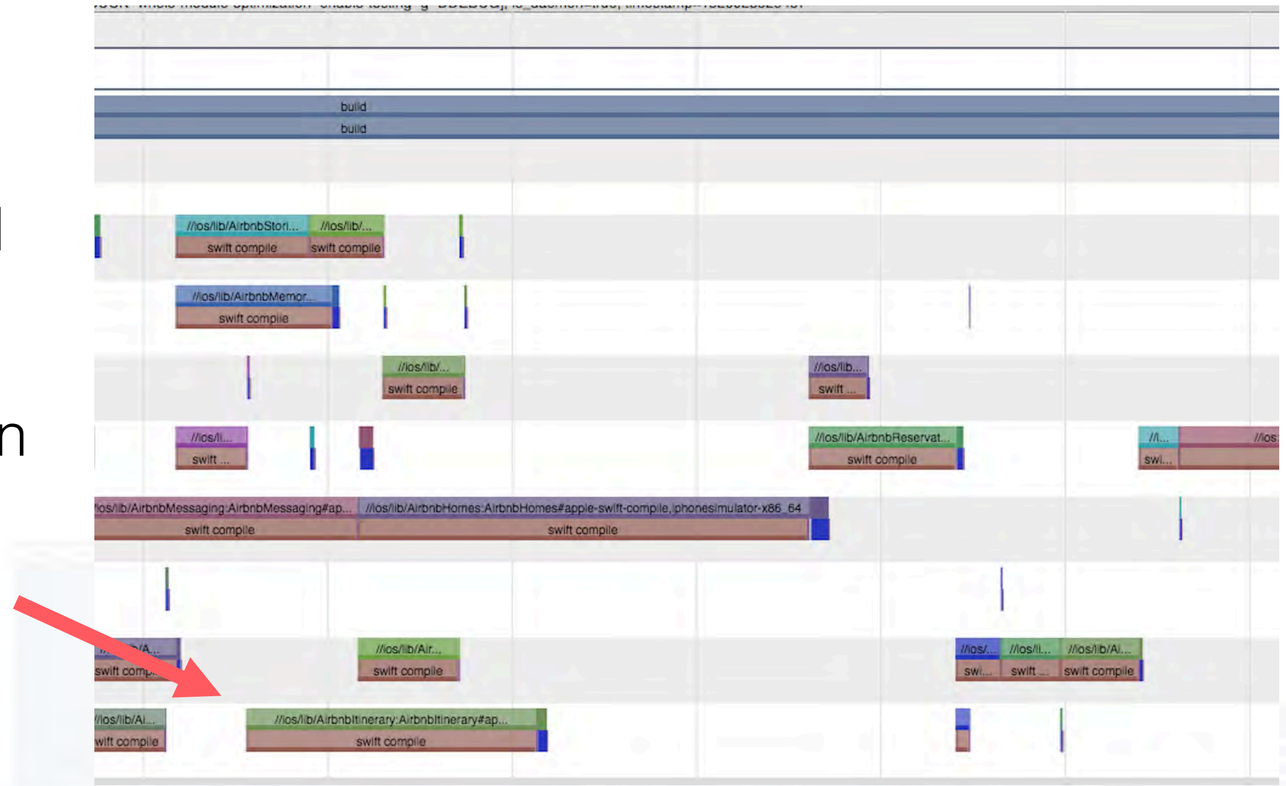


Improve build time show case

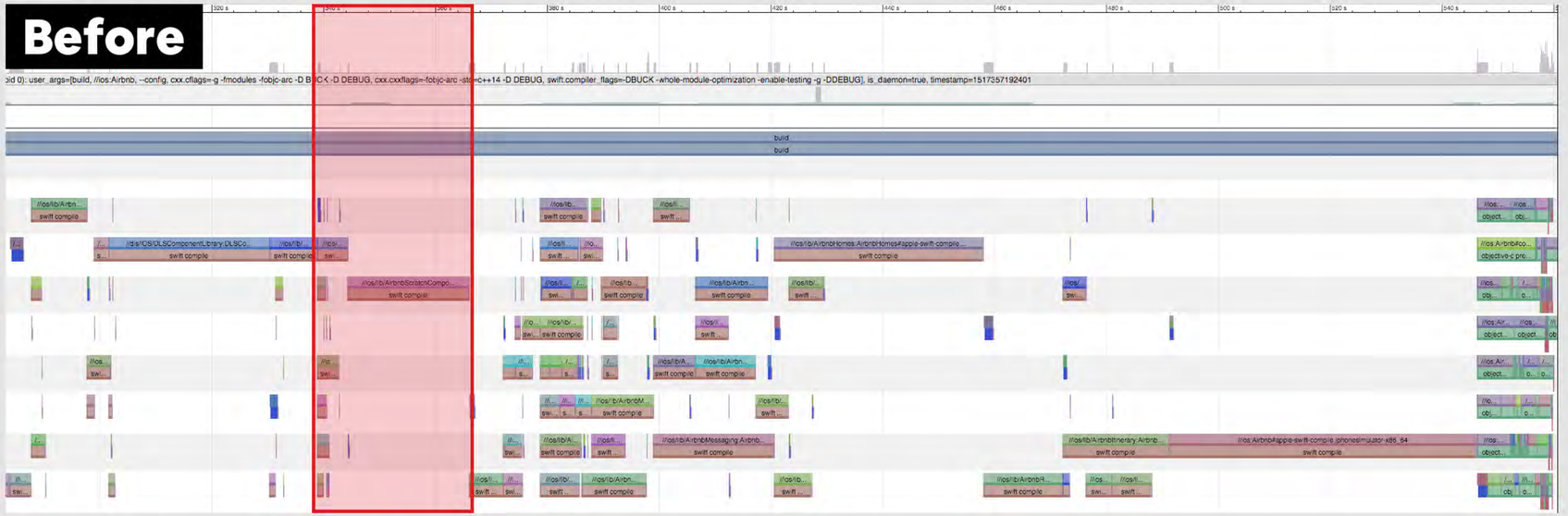


Solution

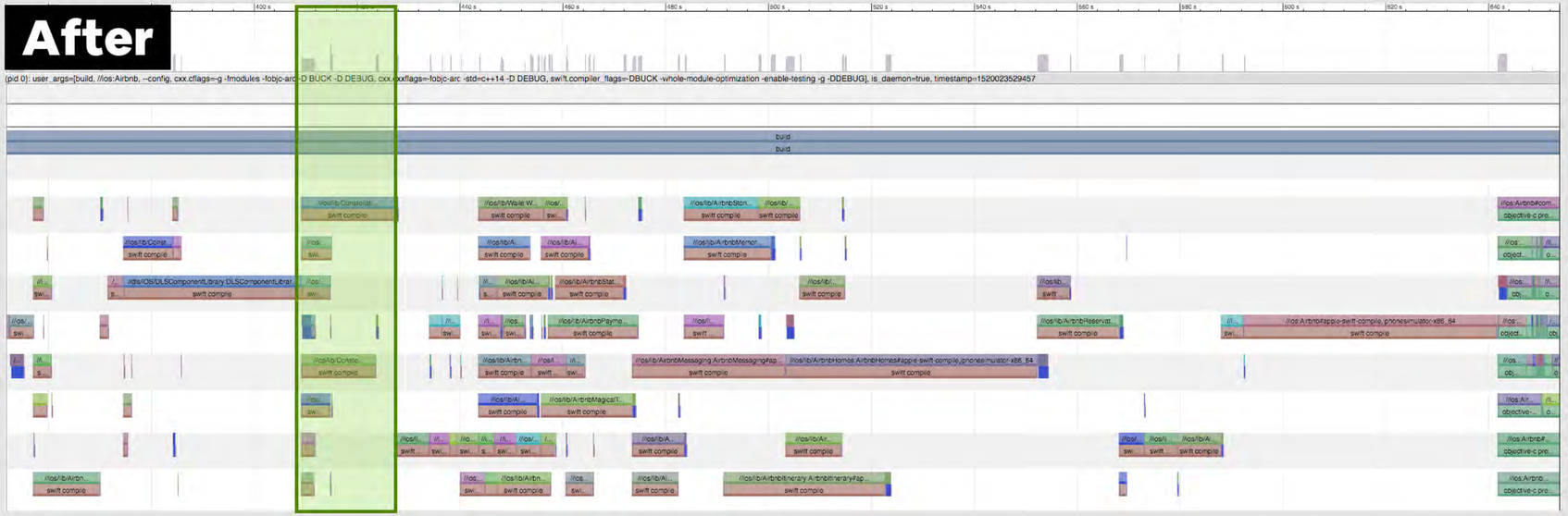
- Issue: AirbnbItinerary is depended by a lot of other modules.
- Solution: Use dependency injection to break up the dependency.



Before



After



关注QCon微信公众号，
获得更多干货！

Thanks!

INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

主办方 Geekbang. 极客邦科技 InfoQ