ByteTech

超级App构建性能提升40%

JoJo，字节自研iOS构建系统

舒彪

Jan,2022

CONTENTS

01_自我介绍

舒彪

2016年开始从事iOS开发，加入字节跳动三年中先后负责过OOM治理、流畅度治理、编译系统开发等相关工作。代表作Slardar MemoryGraph、JoJo构建系统。

02

JoJo简介

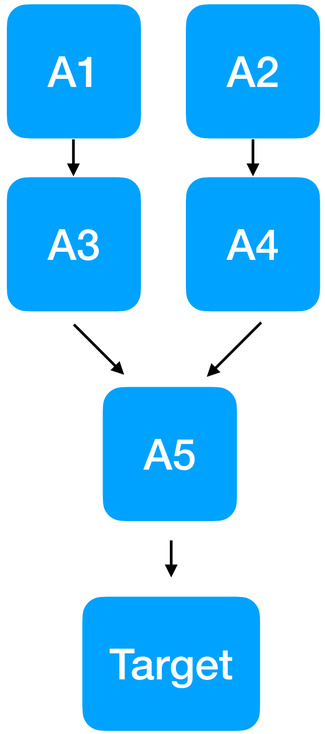
JoJo是一个字节自研的以bazel为核心的iOS构建系统，提供从CI/CD到本地构建开发所需要的一整套解决方案。字节跳动的知名App如抖音、今日头条都使用JoJo进行构建，取得了显著的研发效能提升。

02_JoJo简介

JoJo 与 bazel 的关系

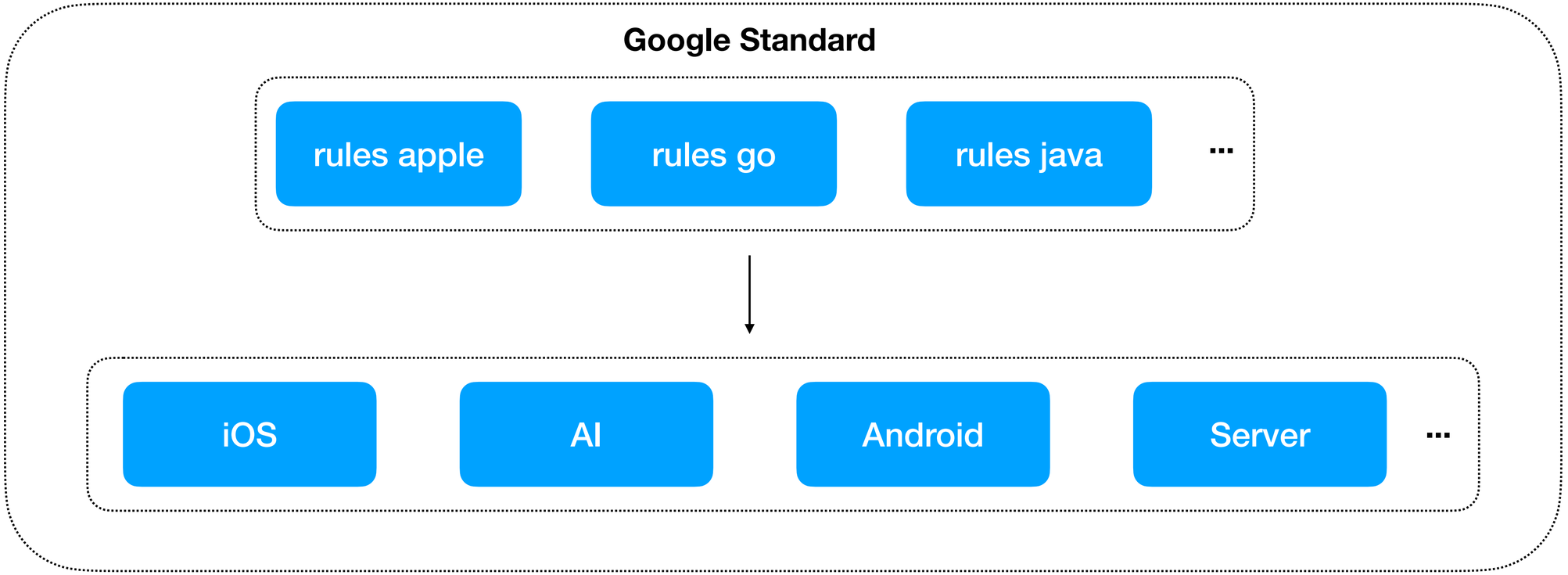
02_JoJo简介

Bazel核心概念



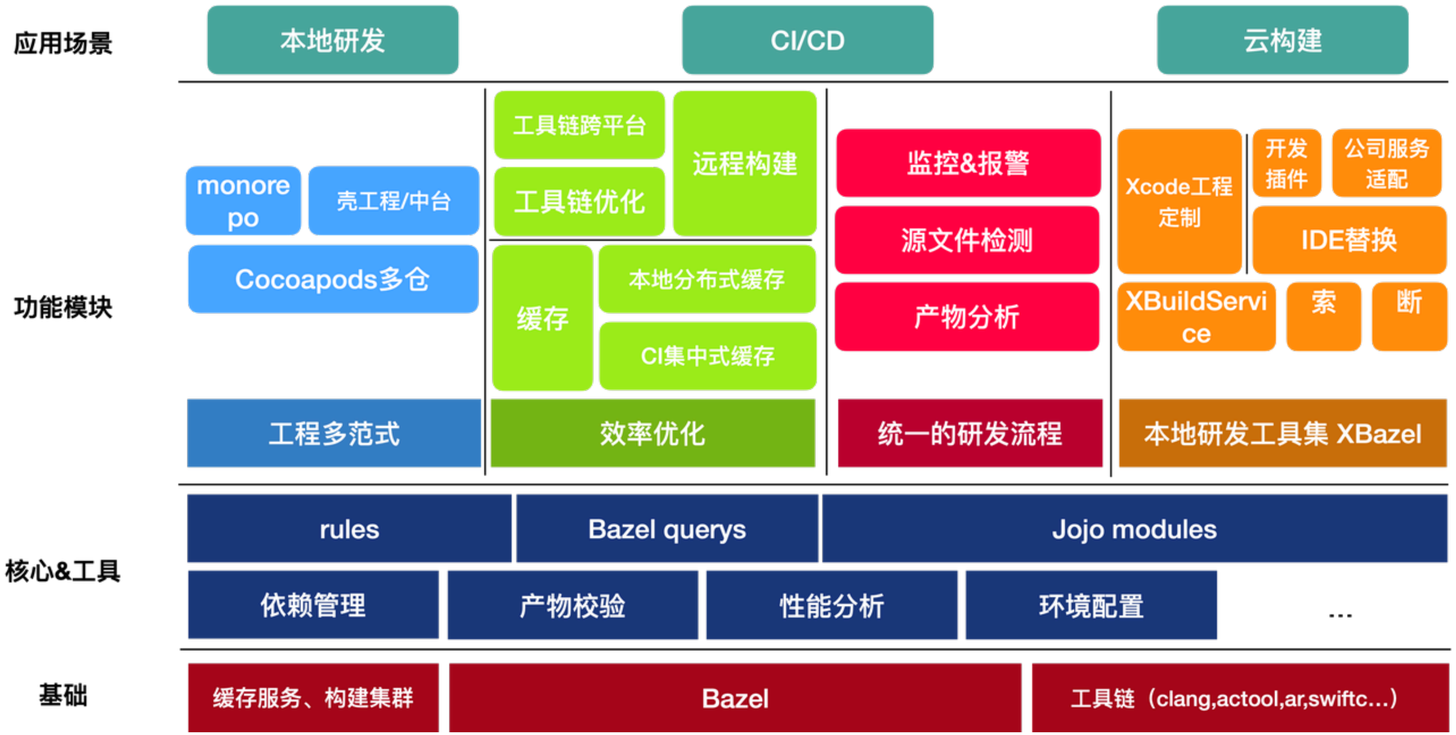
rules & dependencies = workflow

02_JoJo简介


Google Standard
rules apple
rules go
rules java
...
iOS
AI
Android
Server
...

02_JoJo简介

JoJo生态架构图



02_JoJo简介

JoJo的四个特性

高性能

高可扩展性

多工程架构支持

多IDE支持

02_JoJo简介

JoJo的性能表现

25% faster then Xcode + CCache + Pod cache + binary pod

97% cache hit ratio

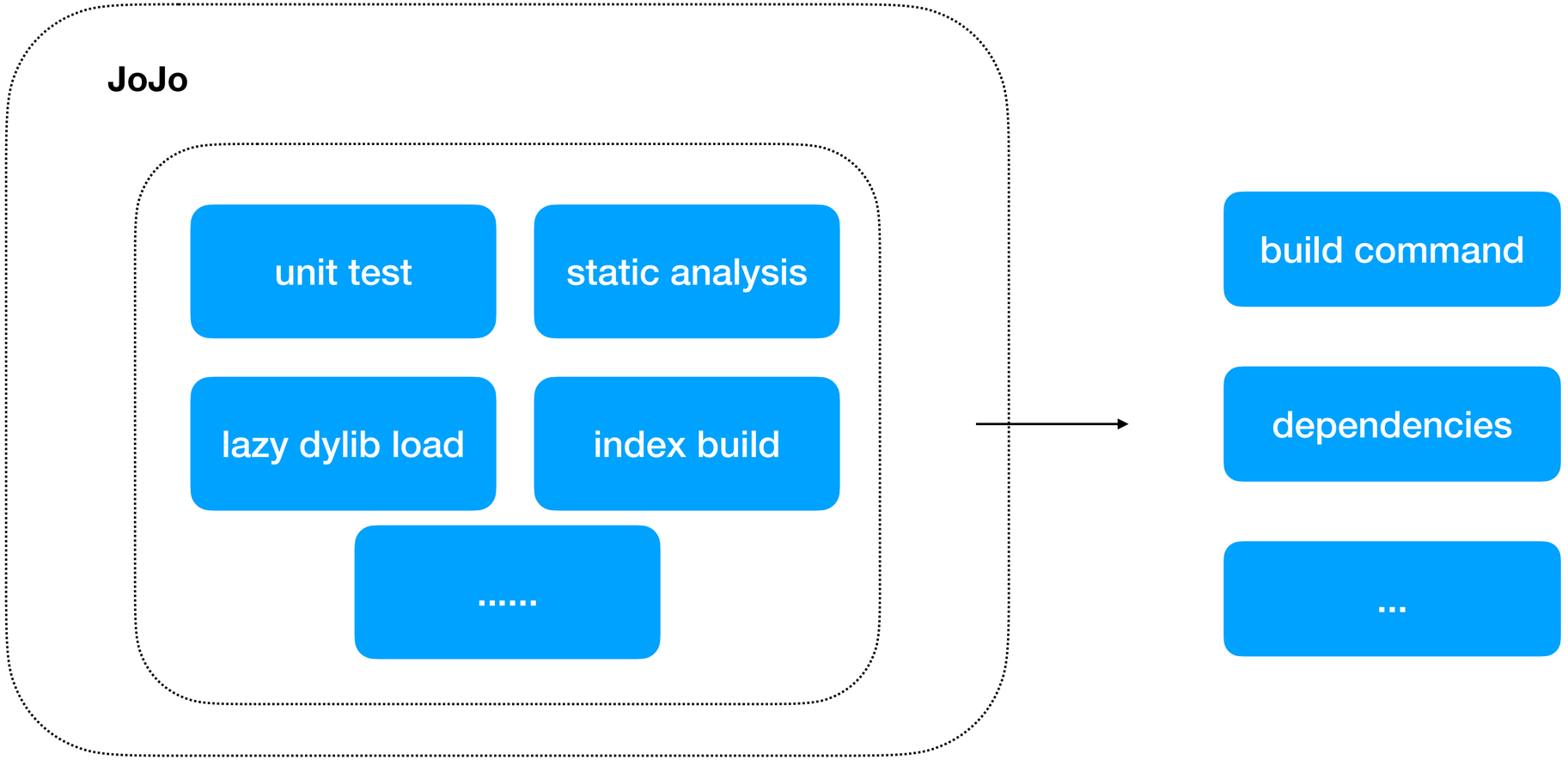
50% faster in local full build

30% faster in local incremental build

and faster pod install、project load、index build ...

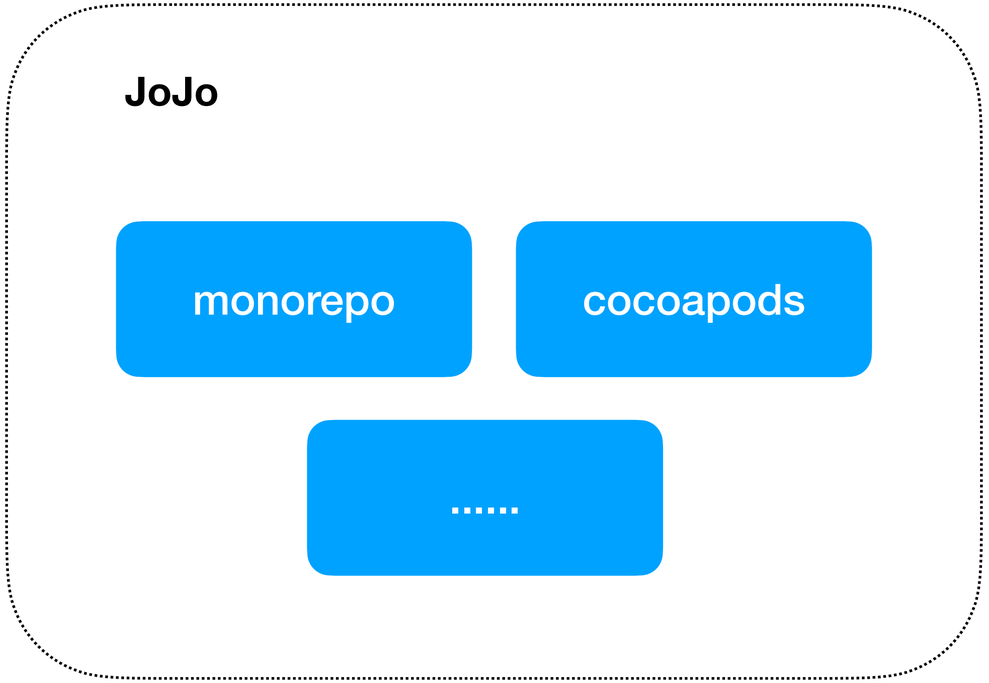
02_JoJo简介

JoJo的可扩展能力



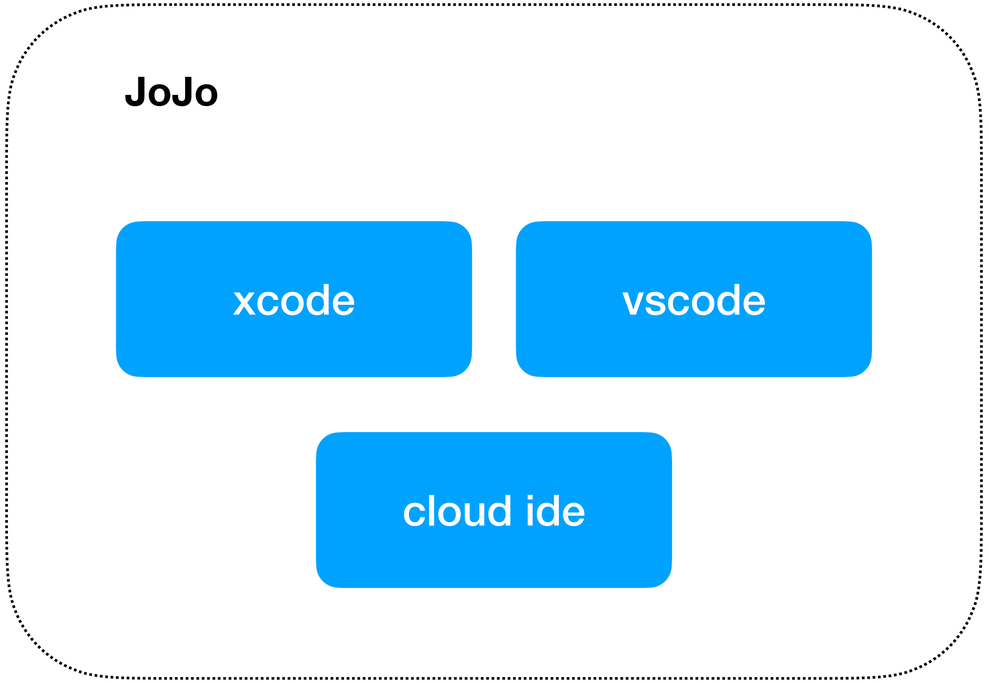
02_JoJo简介

JoJo支持的工程架构



02_JoJo简介

JoJo支持的IDE



03

高性能的基石

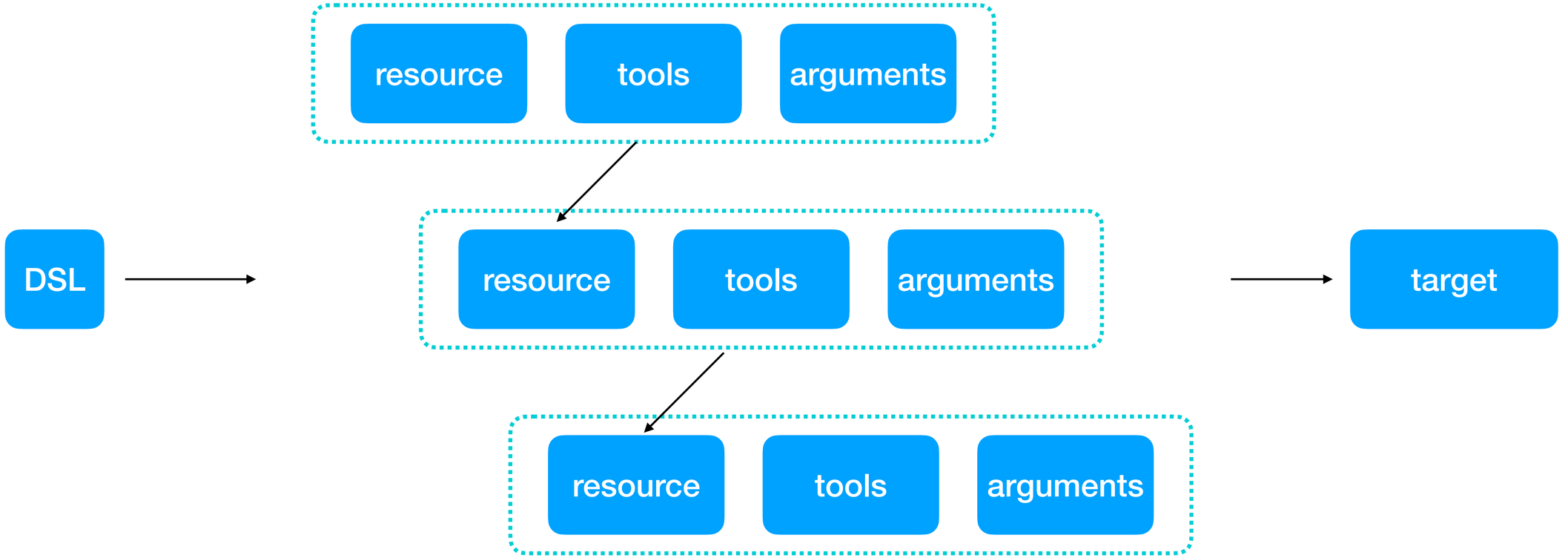
远端缓存

远端执行

依赖计算

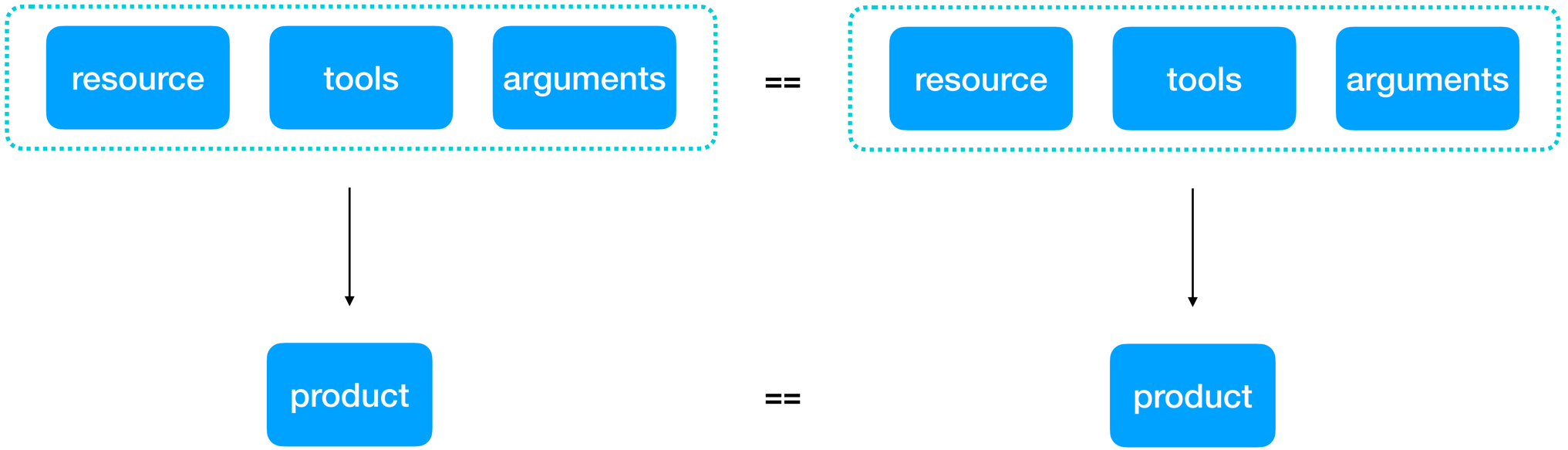
03_高性能的基石

构建的过程



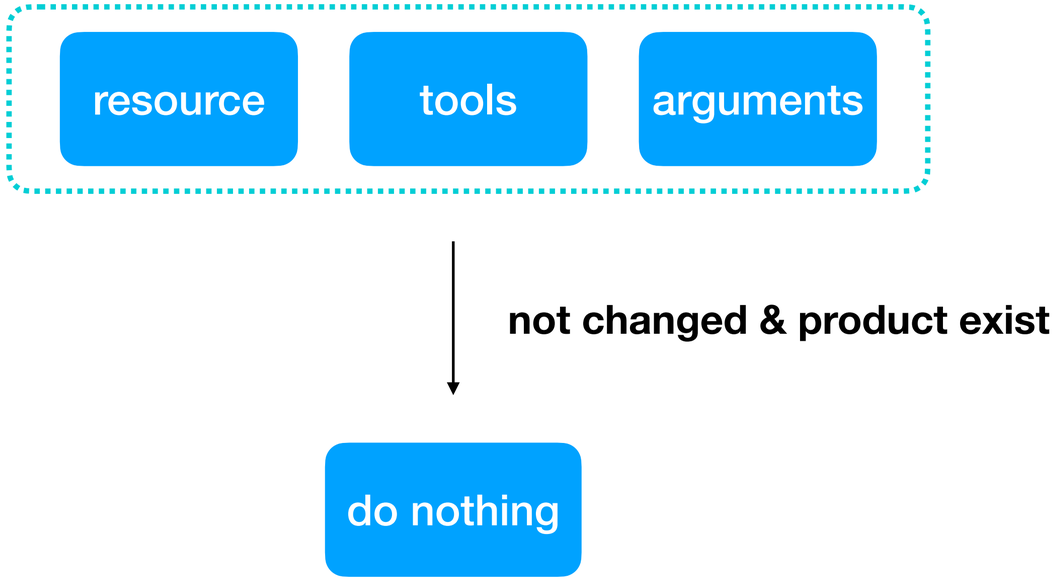
03_高性能的基石

输入决定输出



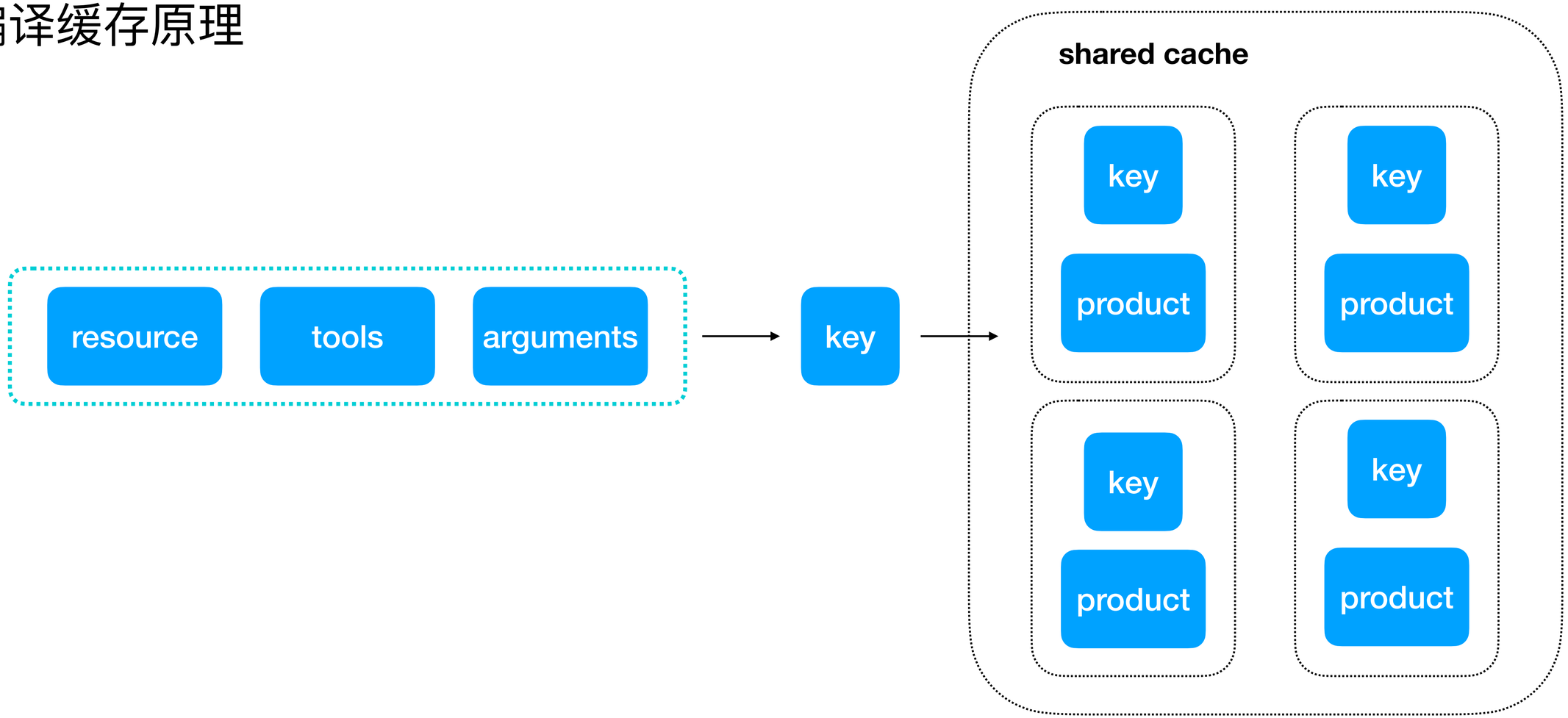
03_高性能的基石

增量构建的产物复用



03_高性能的基石

编译缓存原理



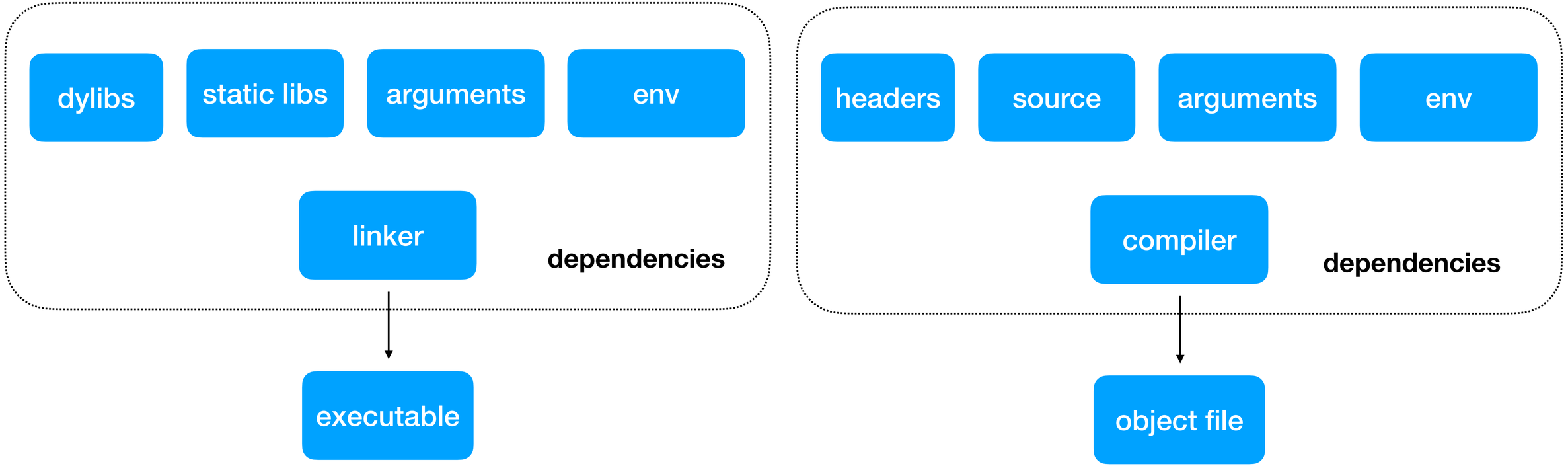
03_高性能的基石

核心问题?

依赖计算

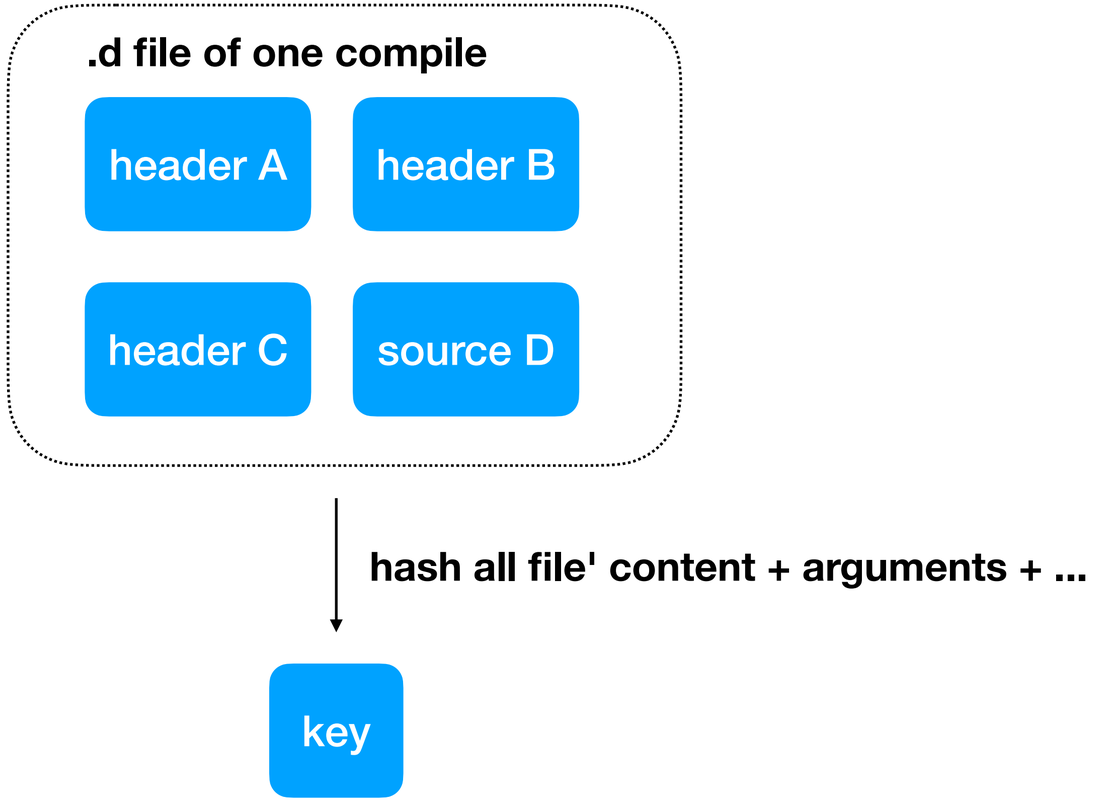
03_高性能的基石

构建任务的依赖



03_高性能的基石

基于.d文件的依赖计算



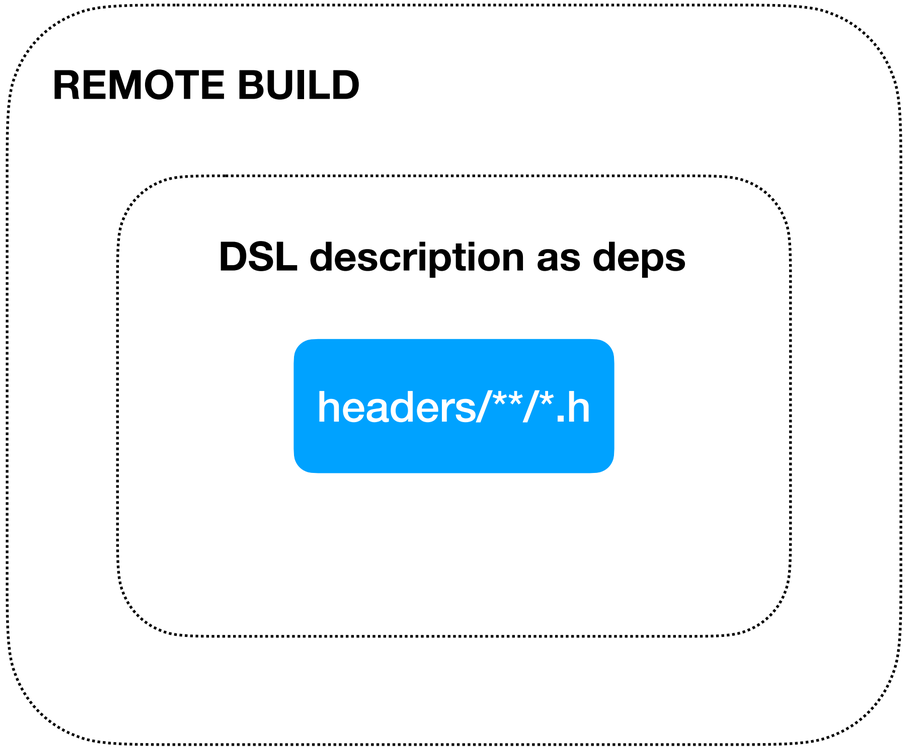
03_高性能的基石

使用上一次构建的.d文件是正确的吗？

.d 不记录头文件上下文

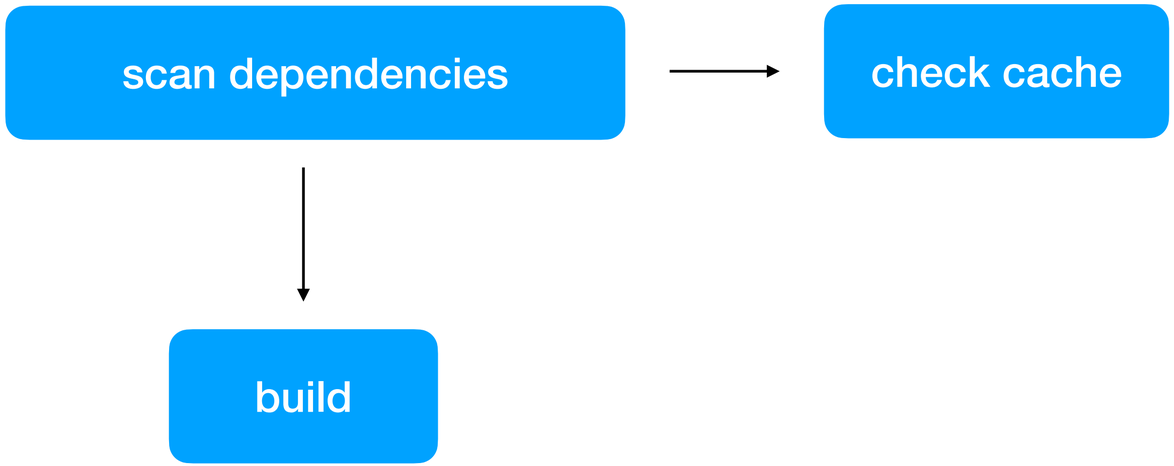
03_高性能的基石

使用用户描述的依赖



03_高性能的基石

即时依赖计算的缓存流程



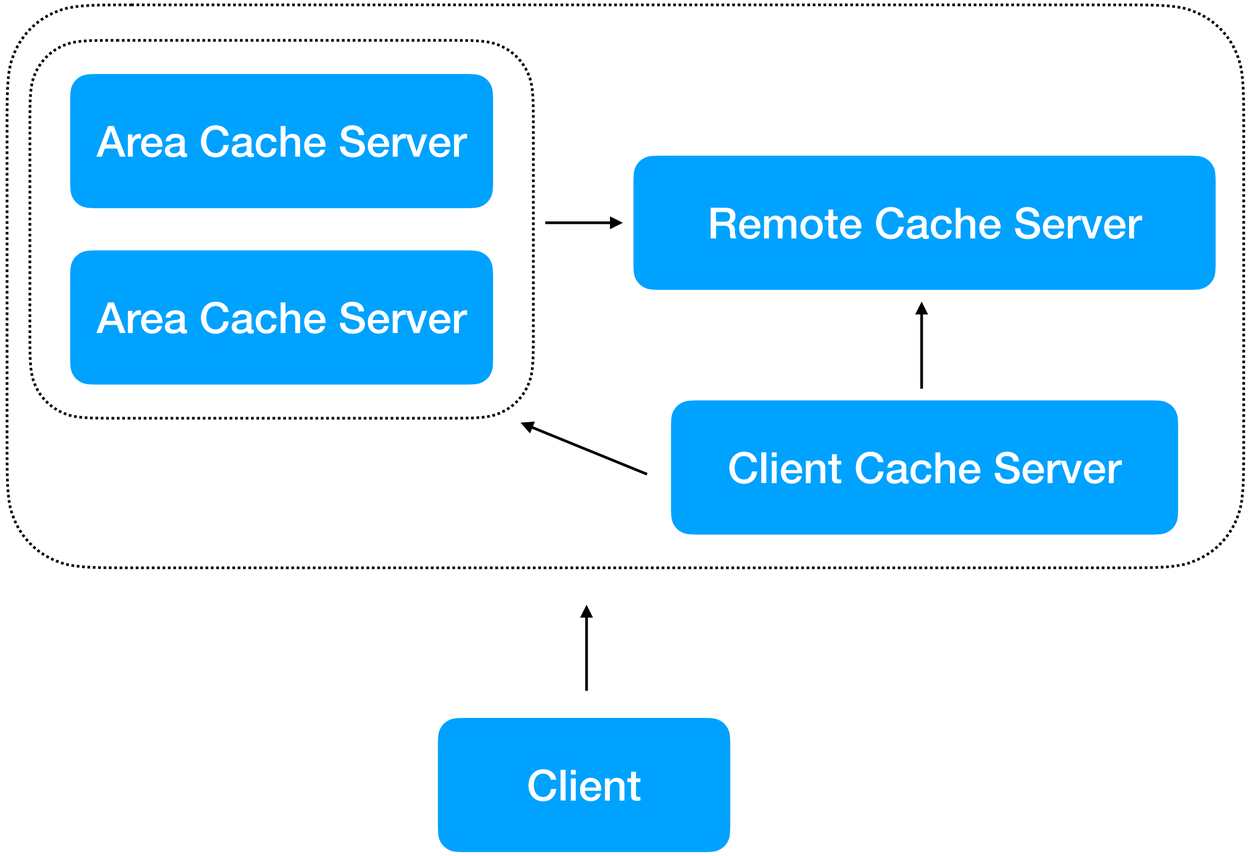
03_高性能的基石

2000+ C系文件扫描数秒内完成

Swift依赖快速计算

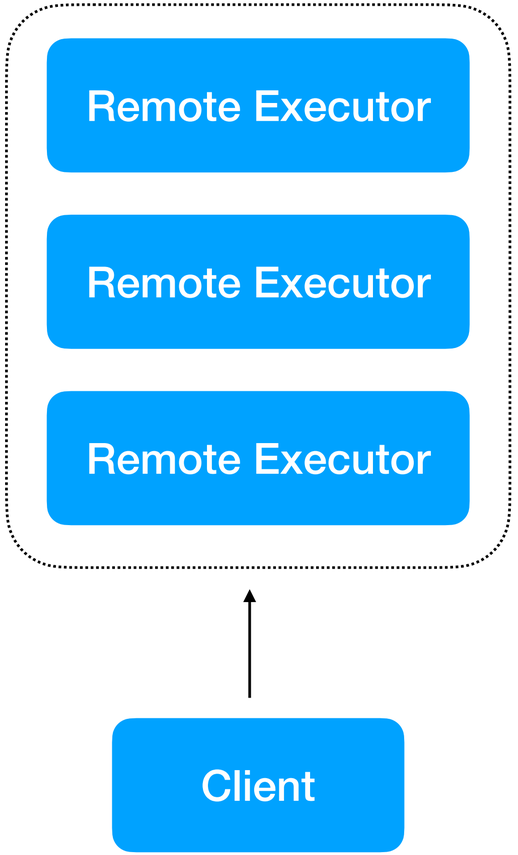
03_高性能的基石

高速缓存网络



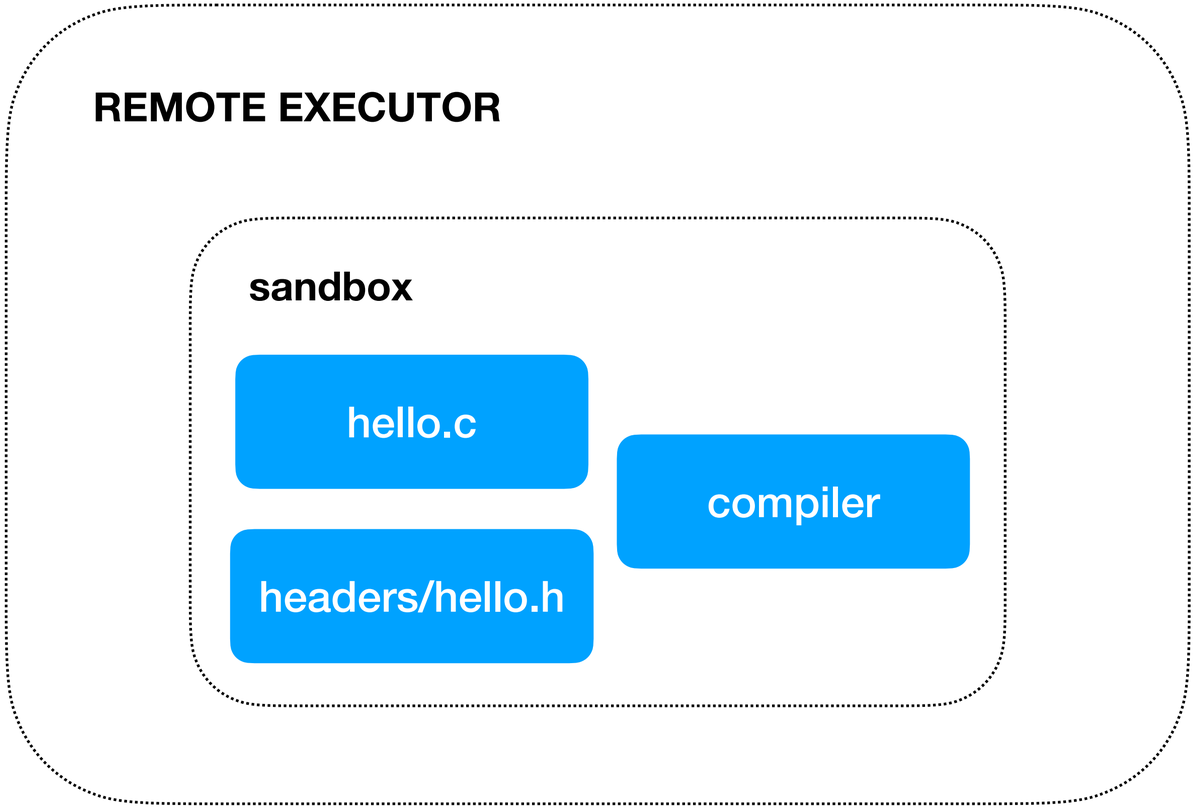
03_高性能的基石

远端执行



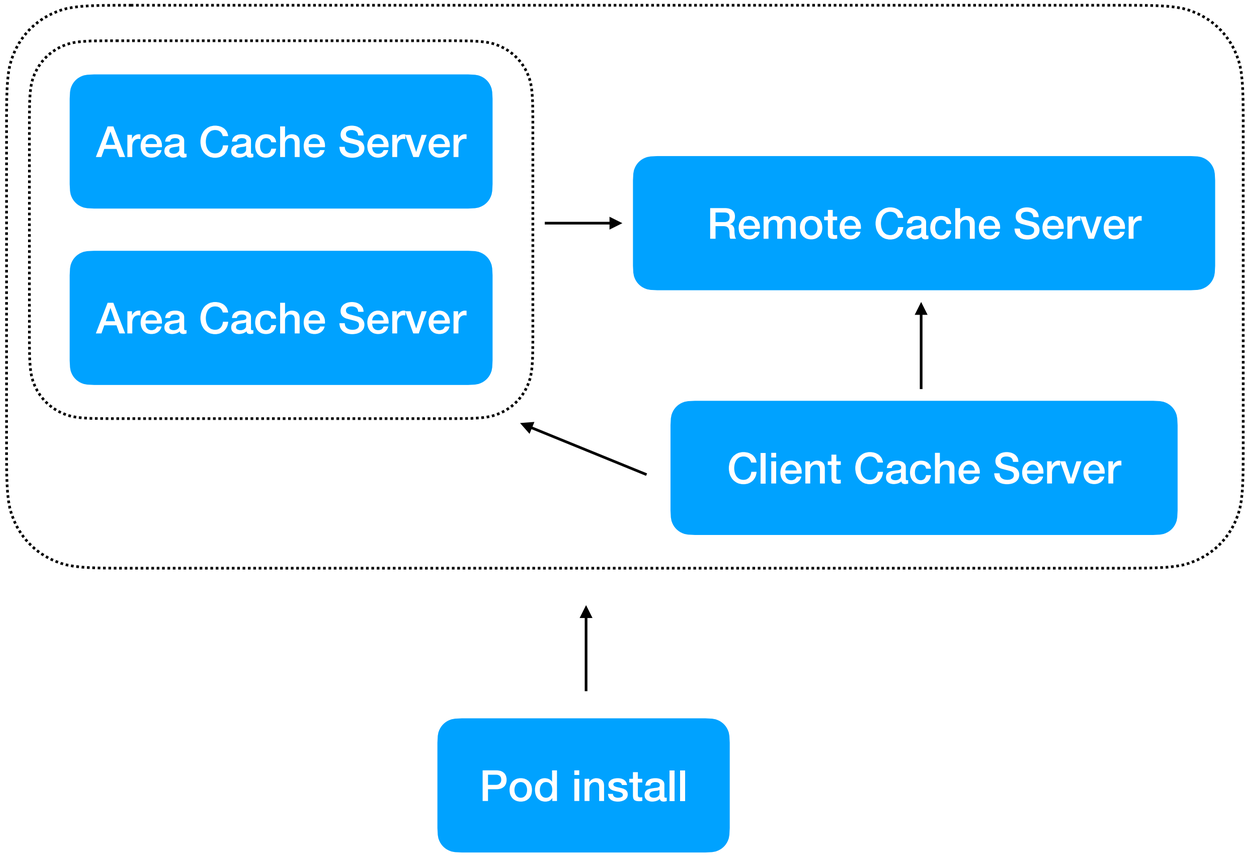
03_高性能的基石

远端执行的原理



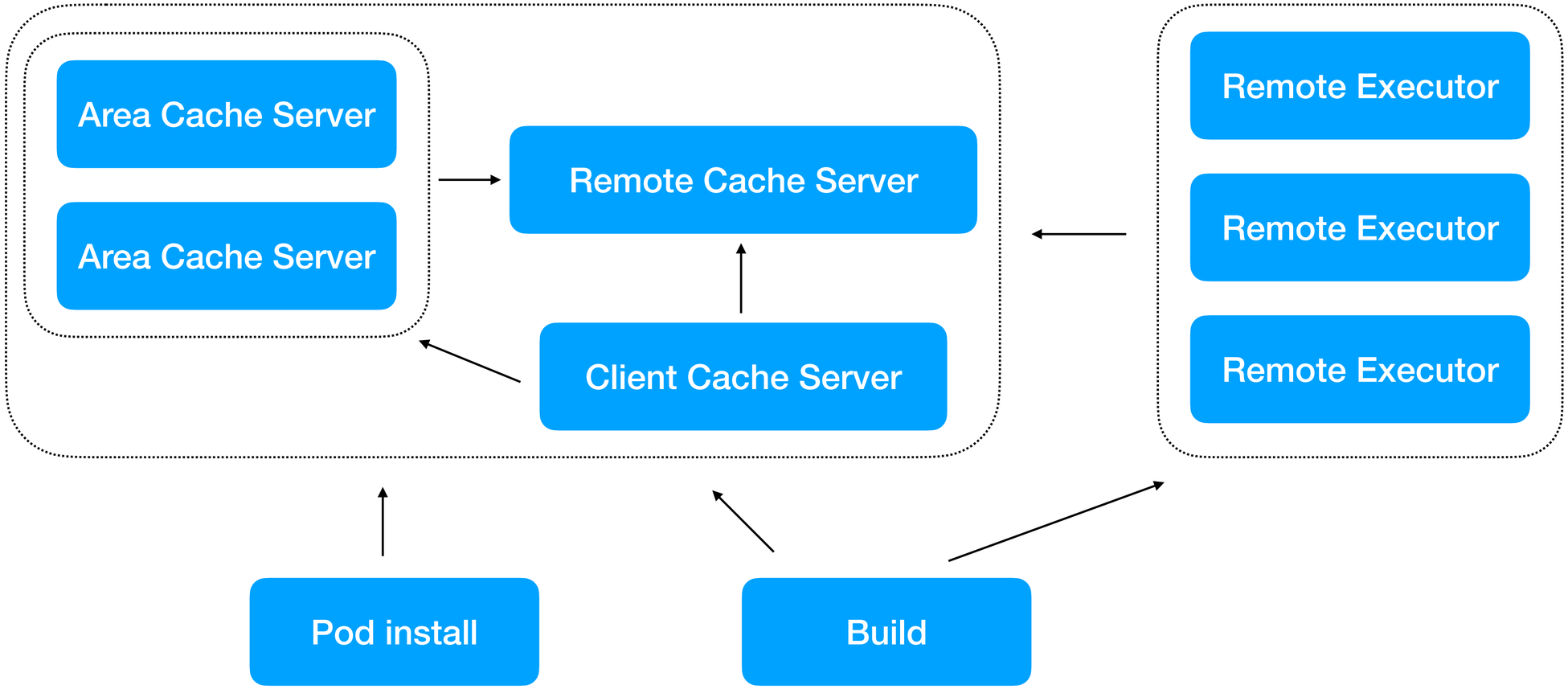
03_高性能的基石

Pod install文件级别的增量仓库下载



03_高性能的基石

分布式构建体系

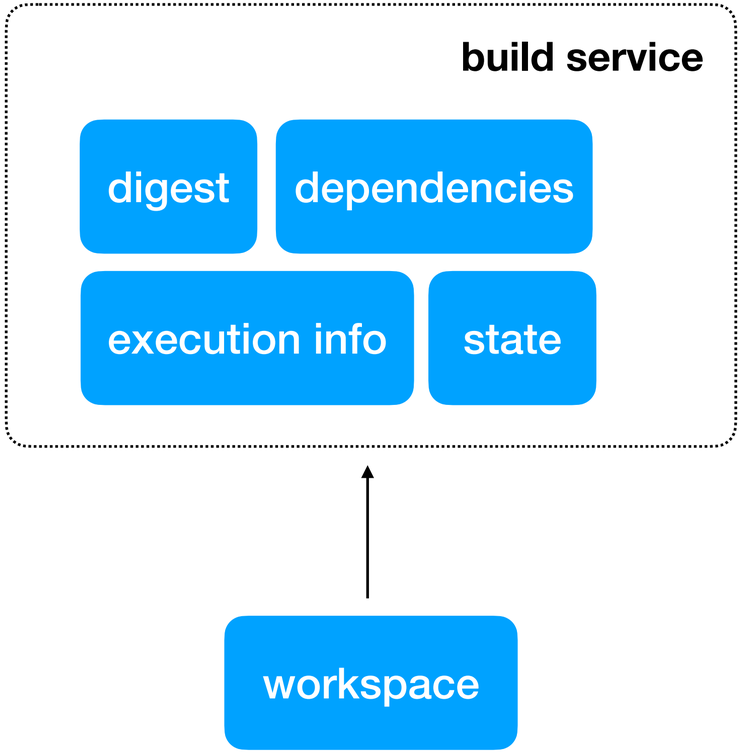


03_高性能的基石

根据资源动态调度

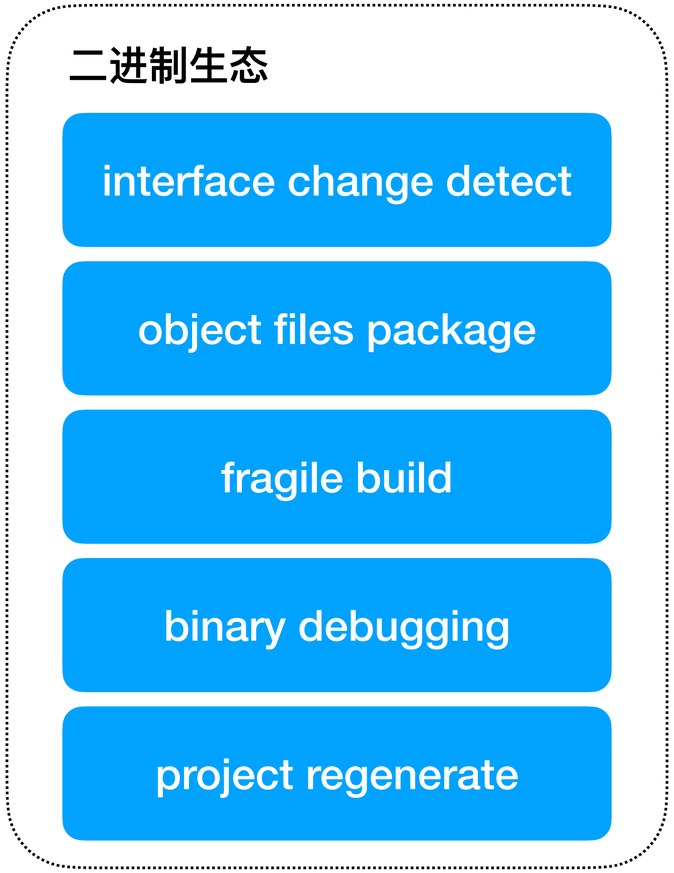
03_高性能的基石

中心式常驻进程使得计算结果共享



03_高性能的基石

业务仓二进制带来的复杂度



03_高性能的基石

抛弃业务仓二进制

03_高性能的基石

支撑Monorepo

04

高可扩展性

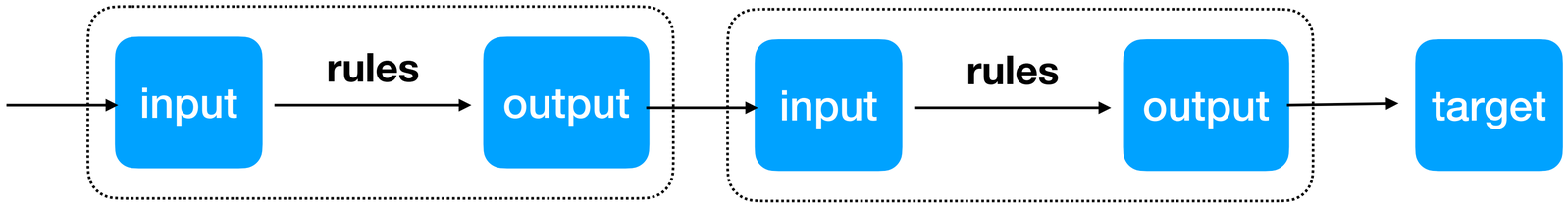
自定义规则

灵活查询

源码可调

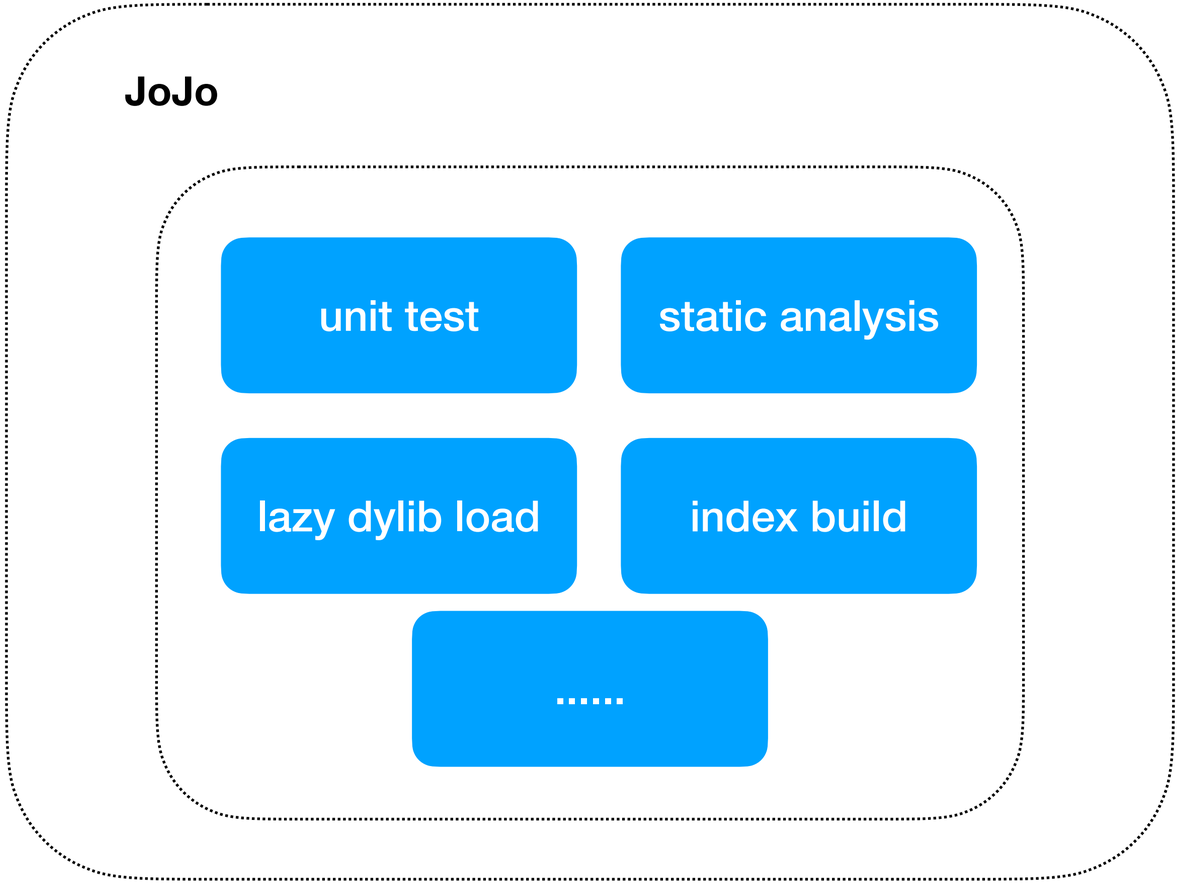
04_高可扩展性

抽象的调度系统



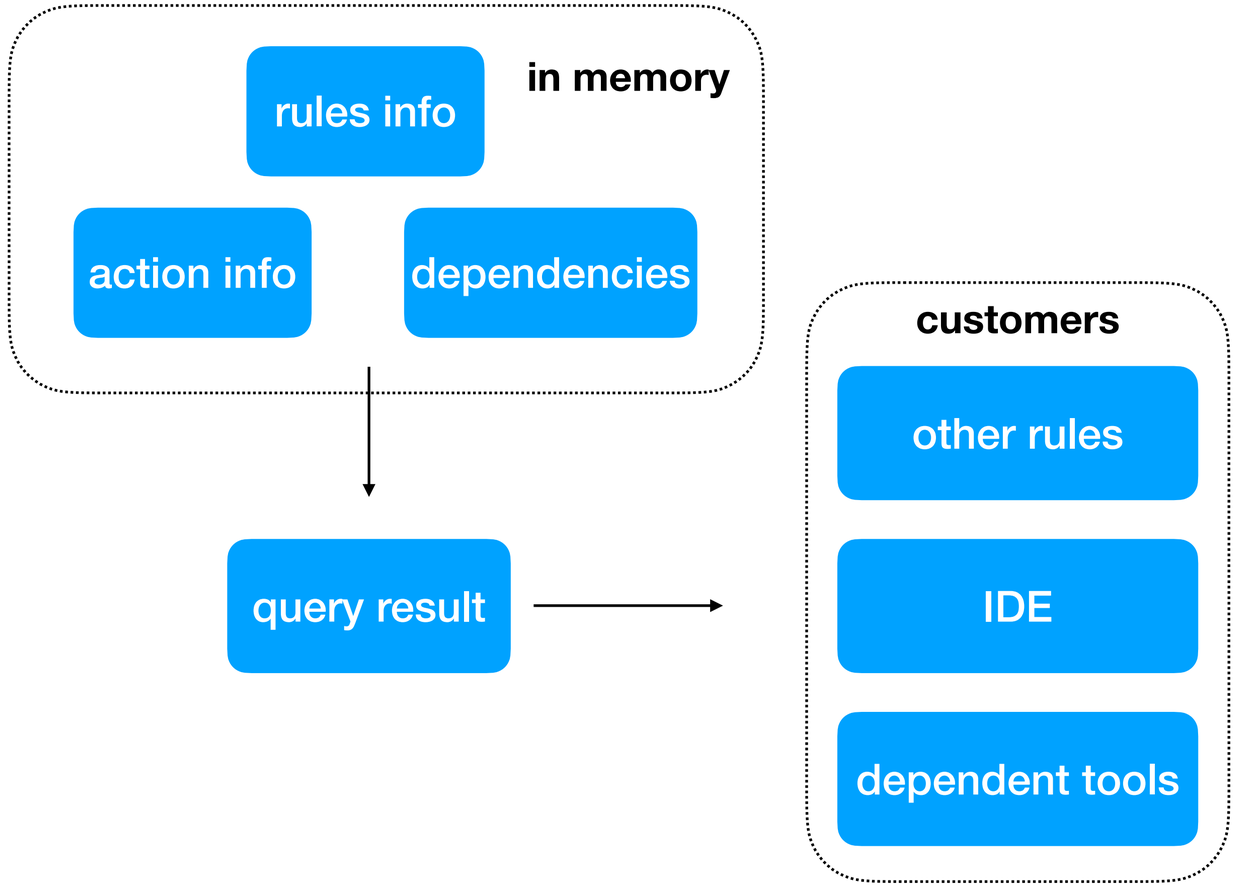
04_高可扩展性

扩展流程的实践



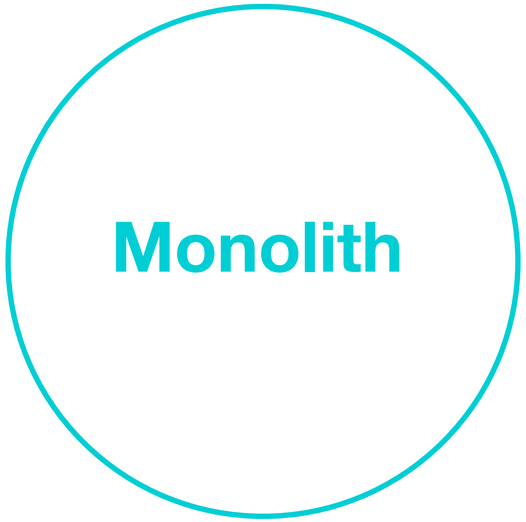
04_高可扩展性

灵活的数据查询



05

多工程架构支持



05_多工程架构支持

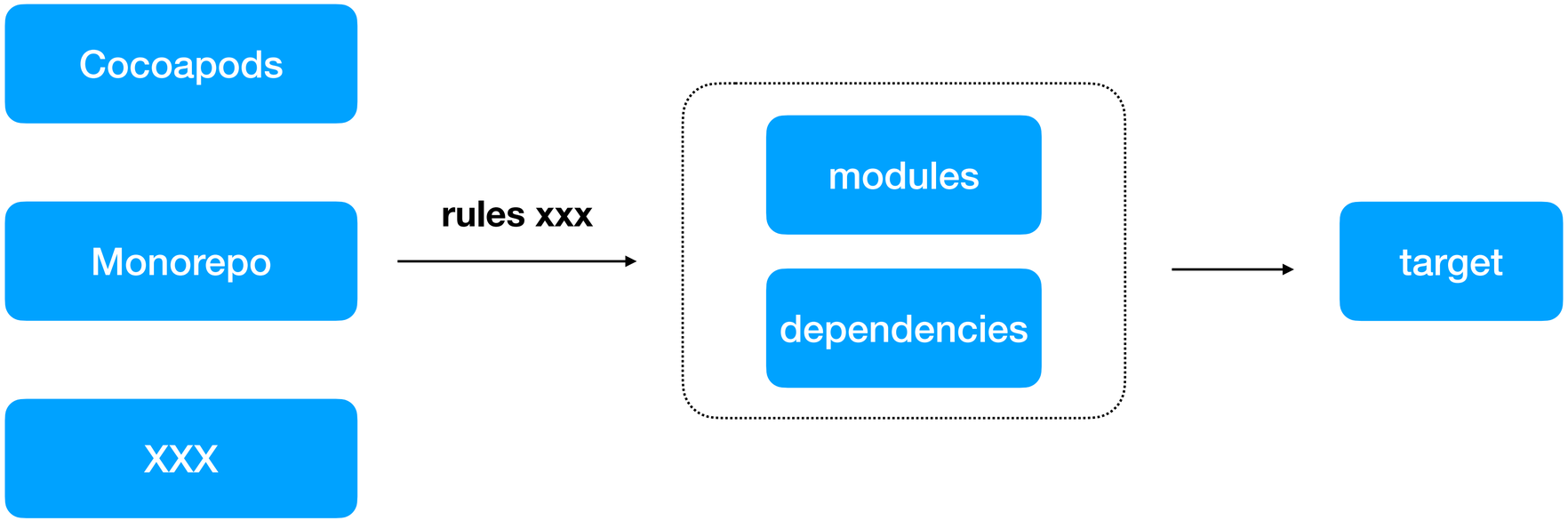
· Cocoapods 工程直接构建

· Cocoapods 、Monorepo混合构建

· Monorepo标准范式

05_多工程架构支持

构建中间层



06

IDE融合

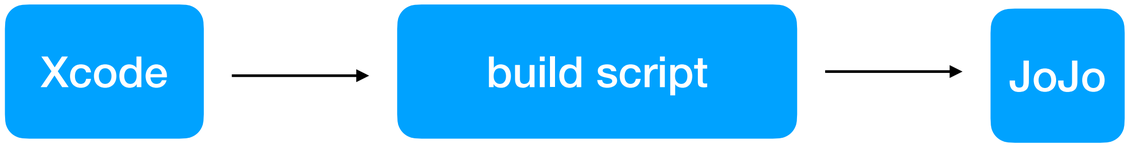
索引

调试

错误和日志

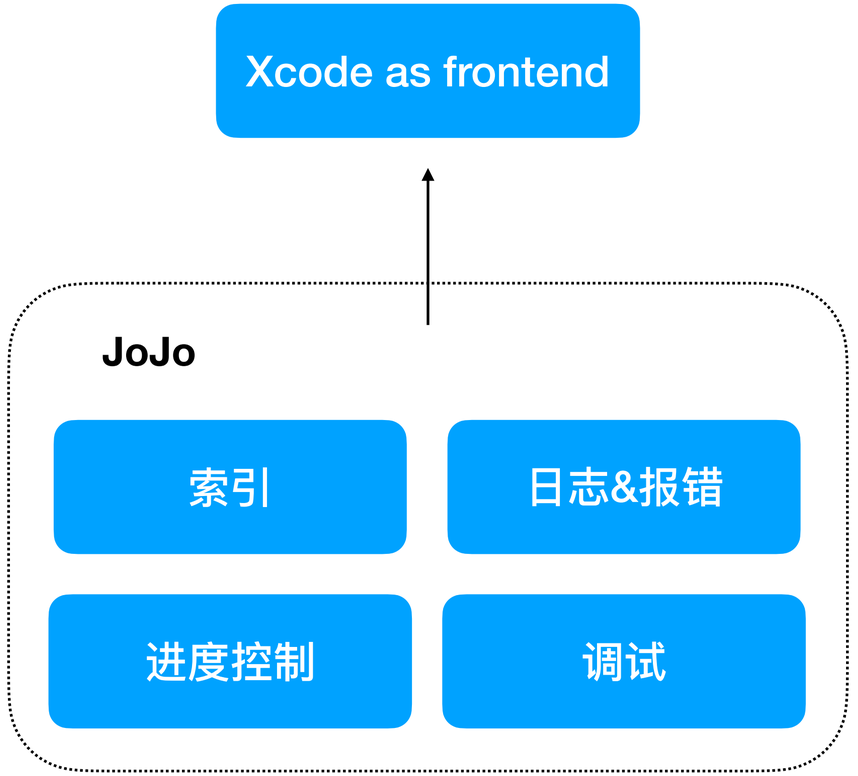
06_IDE融合

Xcode下使用JoJo构建的方式



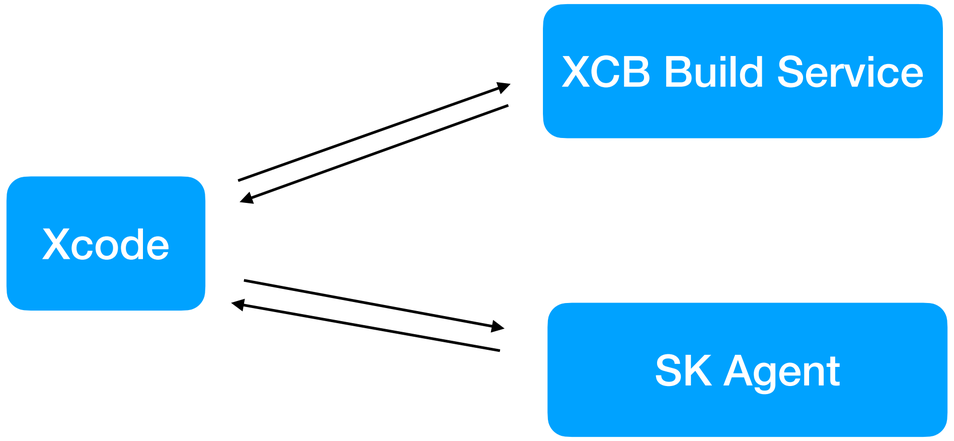
06_IDE融合

Xcode与JoJo的分工



06_IDE融合

Xcode与自带系统的工作流程



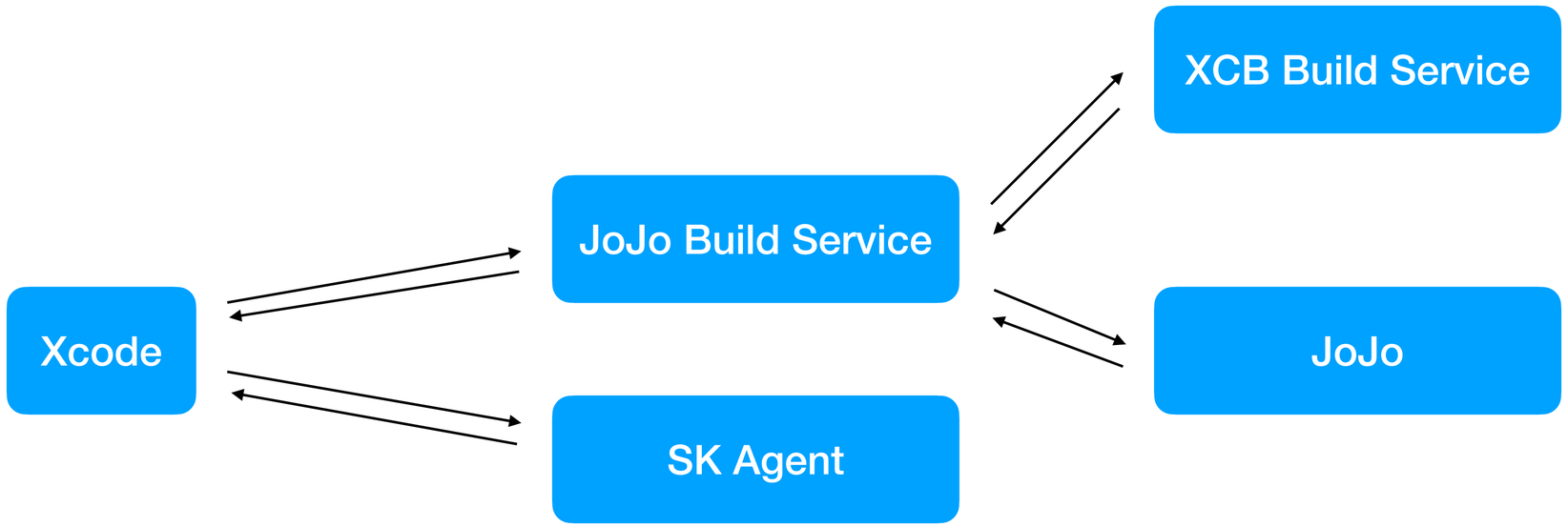
06_IDE融合

Xcode的通信协议

```
➜  ~ /opt/bin/dsdump -sc /Applications/Xcode.app/Contents/SharedFrameworks/
XCBuild.framework/Versions/A/PlugIns/XCBBuildService.bundle/Contents/Framew
orks/XCBProtocol.framework/Versions/A/XCBProtocol -a x86_64
 protocol XCBProtocol.Message // 2 requirements
 protocol XCBProtocol.SessionMessage // 2 requirements
 protocol XCBProtocol.ClientExchangeMessage // 2 requirements

 struct XCBProtocol.XBSTrainInfo {

        // Properties
        let update : String // +0x0
        let name : String // +0x10
WARNING: couldn't find address 0x0 (0x0) in binary!
        let view : ?? // +0x20
        let xcconfigOverrideContents : String // +0x30
        let isSideTrain : Bool // +0x40
 }

 enum XCBProtocol.SourceTree {

        // Properties
        case buildSetting : String
        case absolute
        case groupRelative
 }
```

06_IDE融合

JoJo体系下的工作流程



06_IDE融合

- 索引缓存
- 二进制调试源码索引
- 错误提示优化与指引

结语

Q&A

飞书交流群

ByteTech

THANKS.

Jan,2022