快手Android编译优化

周全
Android工程师

我们为什么要做编译优化？？？

我们为什么要做编译优化？？？

全量编译20分钟

慢

增量编译5分钟

人生一半的时间都在编译！！！

我们为什么要做编译优化？？？

卡

进程名称	内存	VM 被压缩	线程	端口	PID
java	6.93 GB	6.92 GB	44	127	64777
Android Studio	4.81 GB	3.23 GB	133	821	34377
java	3.47 GB	1.10 GB	56	151	69597
java	1.66 GB	1.65 GB	41	118	61992
java	734.0 MB	730.8 MB	30	99	67318

进程名称	% CPU	CPU 时间	线程
java	403.2	47:15.28	164
java	72.0	6:45.54	36
Android Studio	50.9	3:08:13.95	165

为什么慢？？？

为什么慢？？？

千万行源码

快200个模块

3000+ task

不能拆分吗？

已经拆分了

数百个子库

几十个中台项目

十来个插件

release包里有9个dex
你想想我们的代码库有多么庞大！！

我们能做什么

常规手段

你用Google能搜到什么呢？

Parallel

Compiler daemon

ABI

组件化

Build Cache

incremental build

Compile avoidance

常规手段

我们把官方方案用到了极致

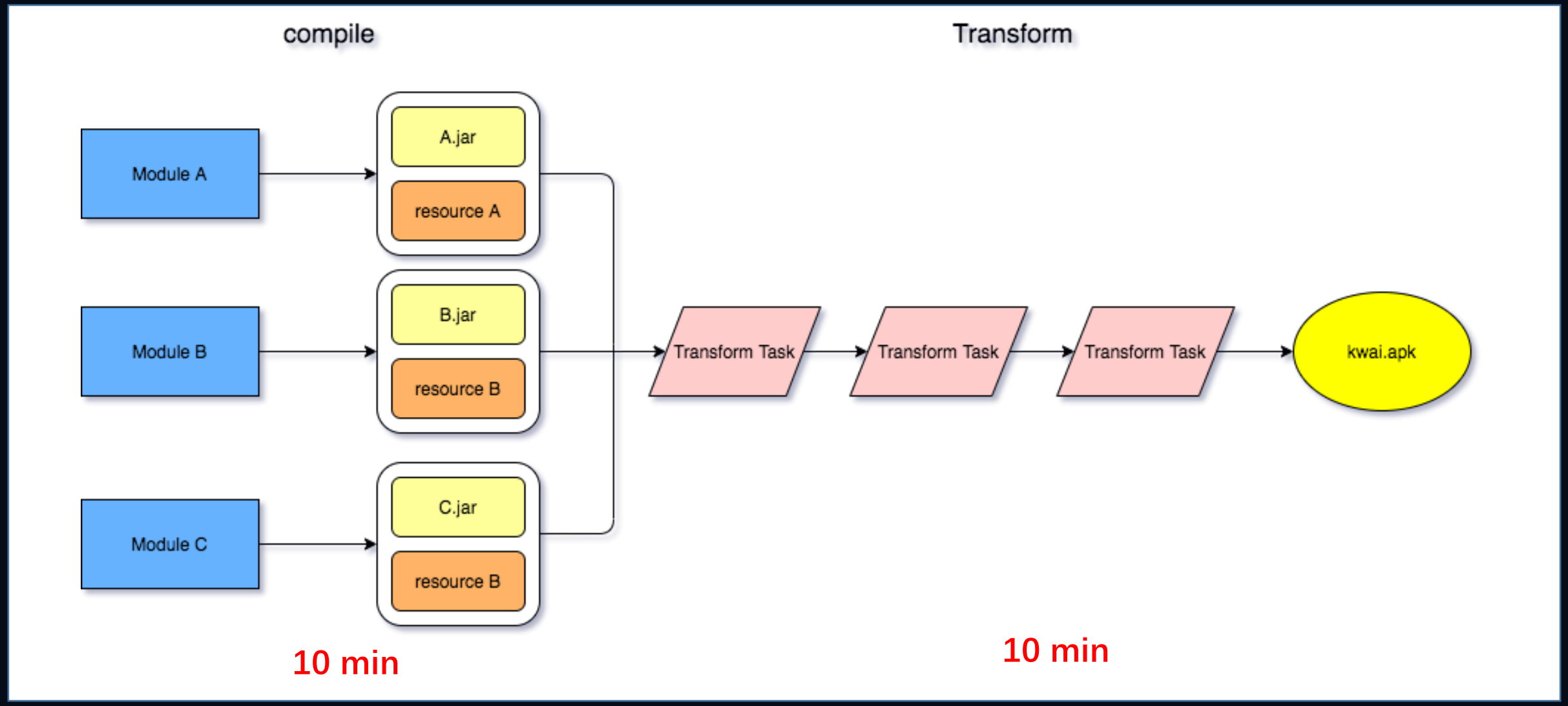
全量编译

增量编译

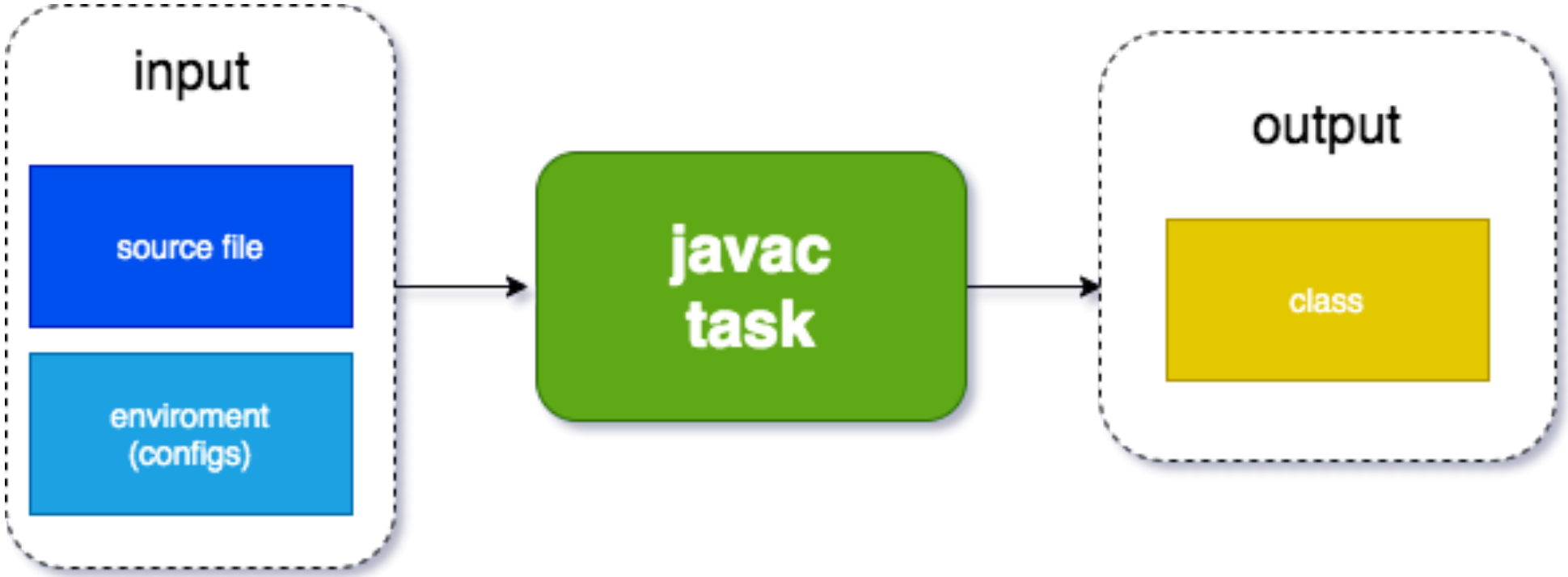
全量编译 25min -> 23 min

增量编译 10min -> 5 min

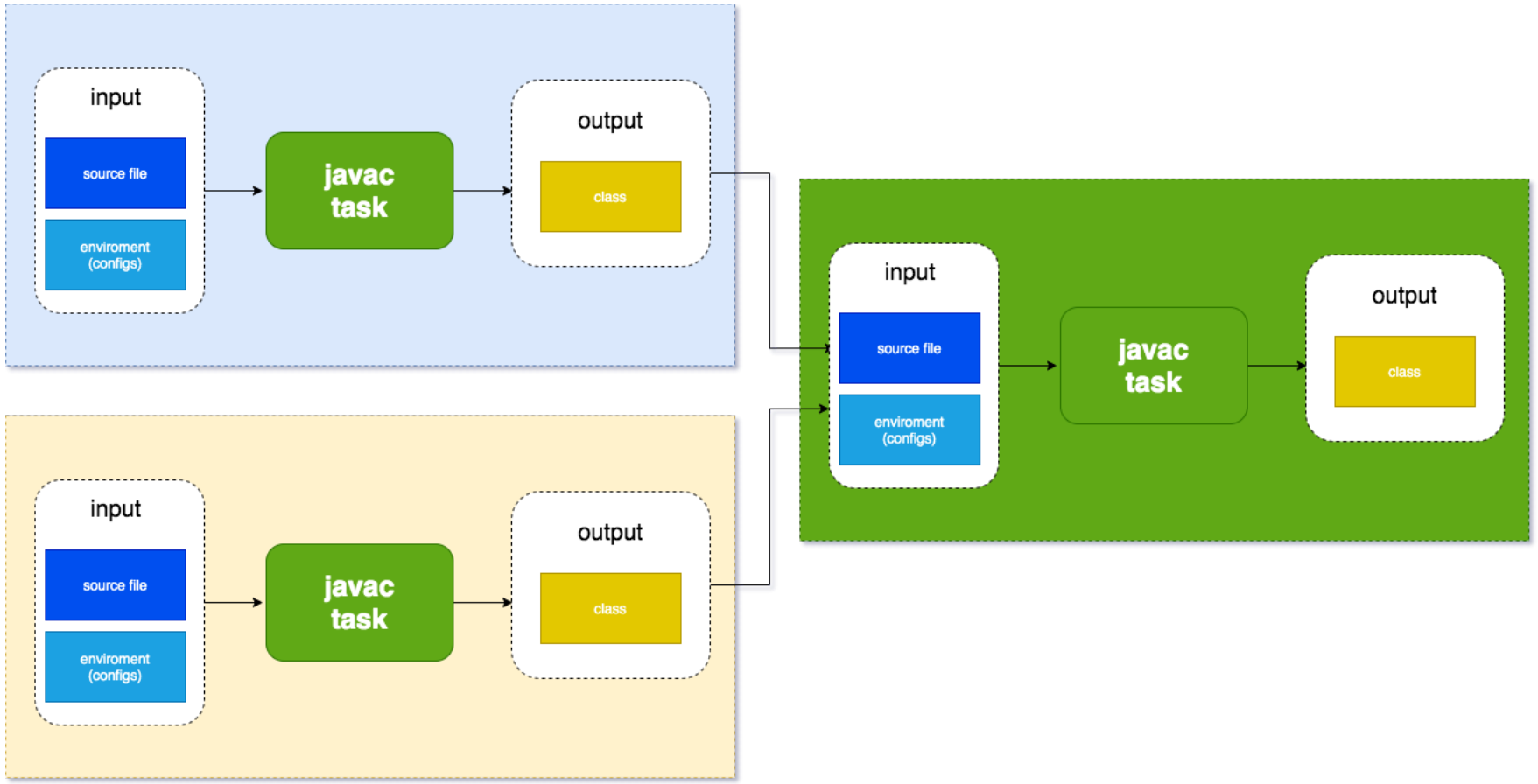
Android编译流程



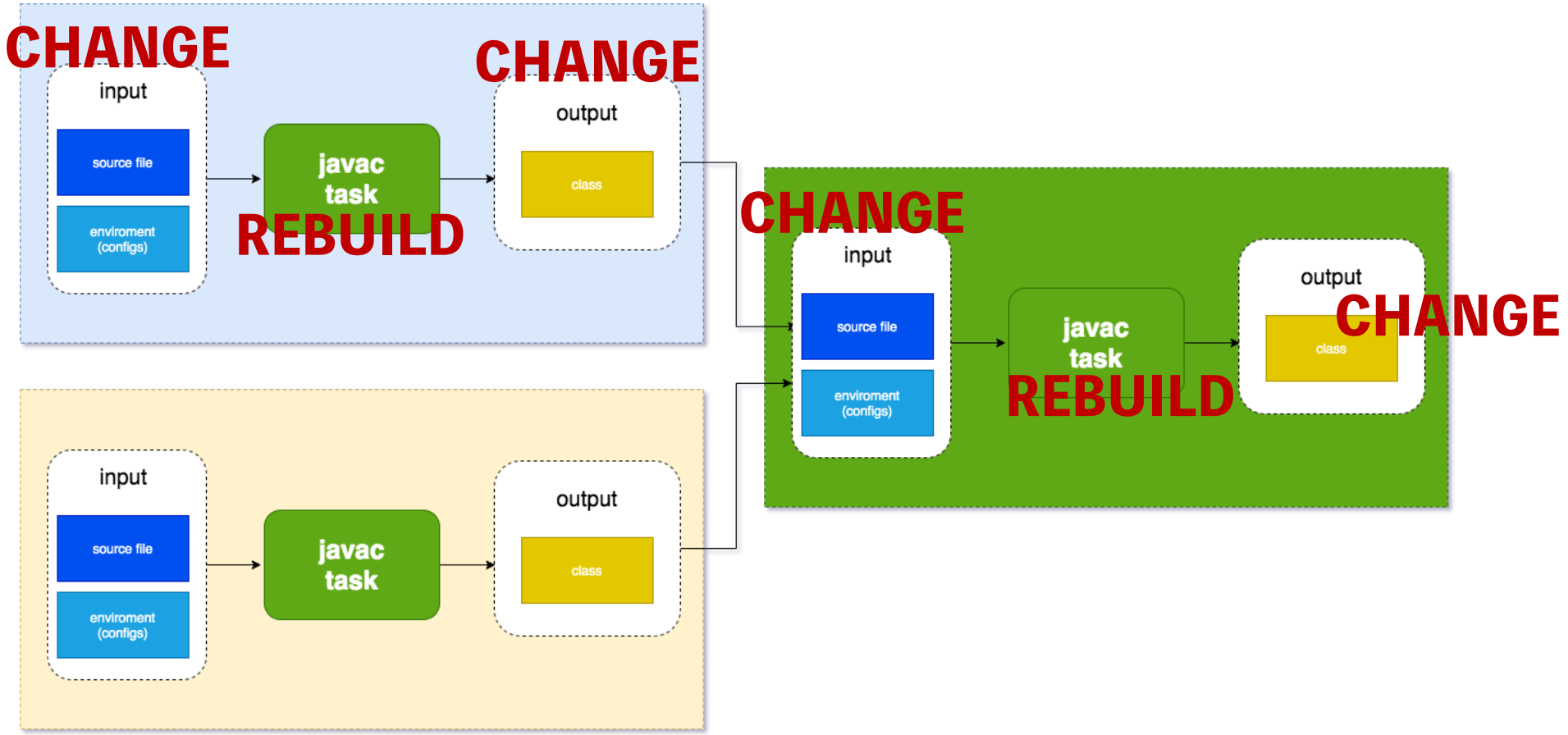
Compile流程


input
source file
enviroment
(configs)
javac
task
output
class

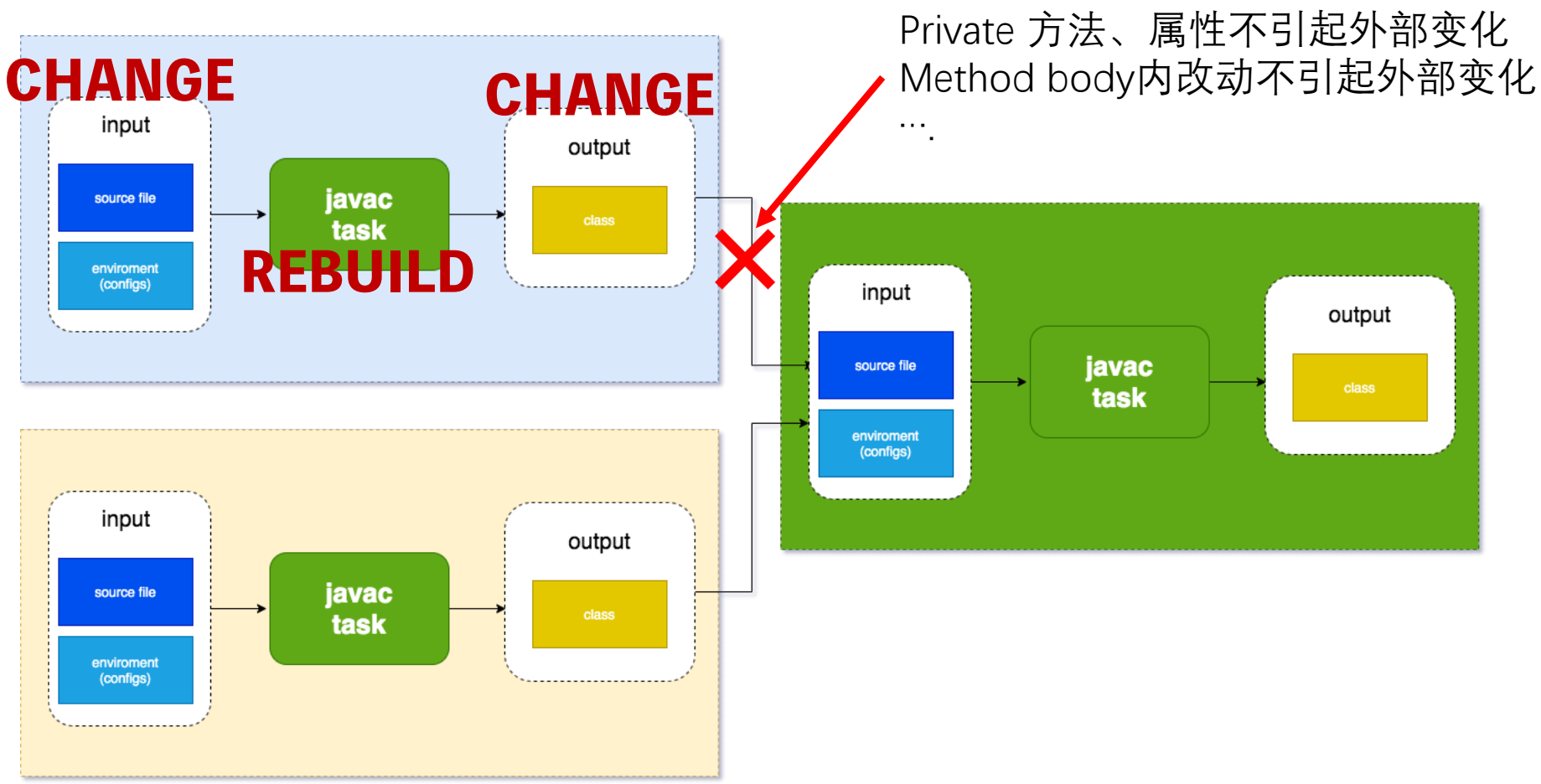
Compile ABI


input
source file
enviroment (configs)
javac task
output
class
input
source file
enviroment (configs)
javac task
output
class
input
source file
enviroment (configs)
javac task
output
class

Compile ABI


CHANGE
input
source file
enviroment (configs)
javac task
REBUILD
CHANGE
output
class
CHANGE
input
source file
enviroment (configs)
javac task
REBUILD
output
CHANGE
class
input
source file
enviroment (configs)
javac task
output
class

Compile ABI



Private 方法、属性不引起外部变化
Method body内改动不引起外部变化
….

Compile流程优化

设计更为激进的ABI模式

语法级别：
public method change
public filed change
public class added

文件级别：
Code Source file change
Resource Source file change

环境级别：
Dependency change
Gradle properties change

Compile Task

Kapt Task

Generate resource Task

```
task.doFirst {
            if (task.state.upToDate) {
              return@doFirst
            }
            val changed = runBlocking {
                diff change
            }

            if (!changed && !task.state.upToDate) {
              throw StopExecutionException()
            }

}
```

编译优化

compile

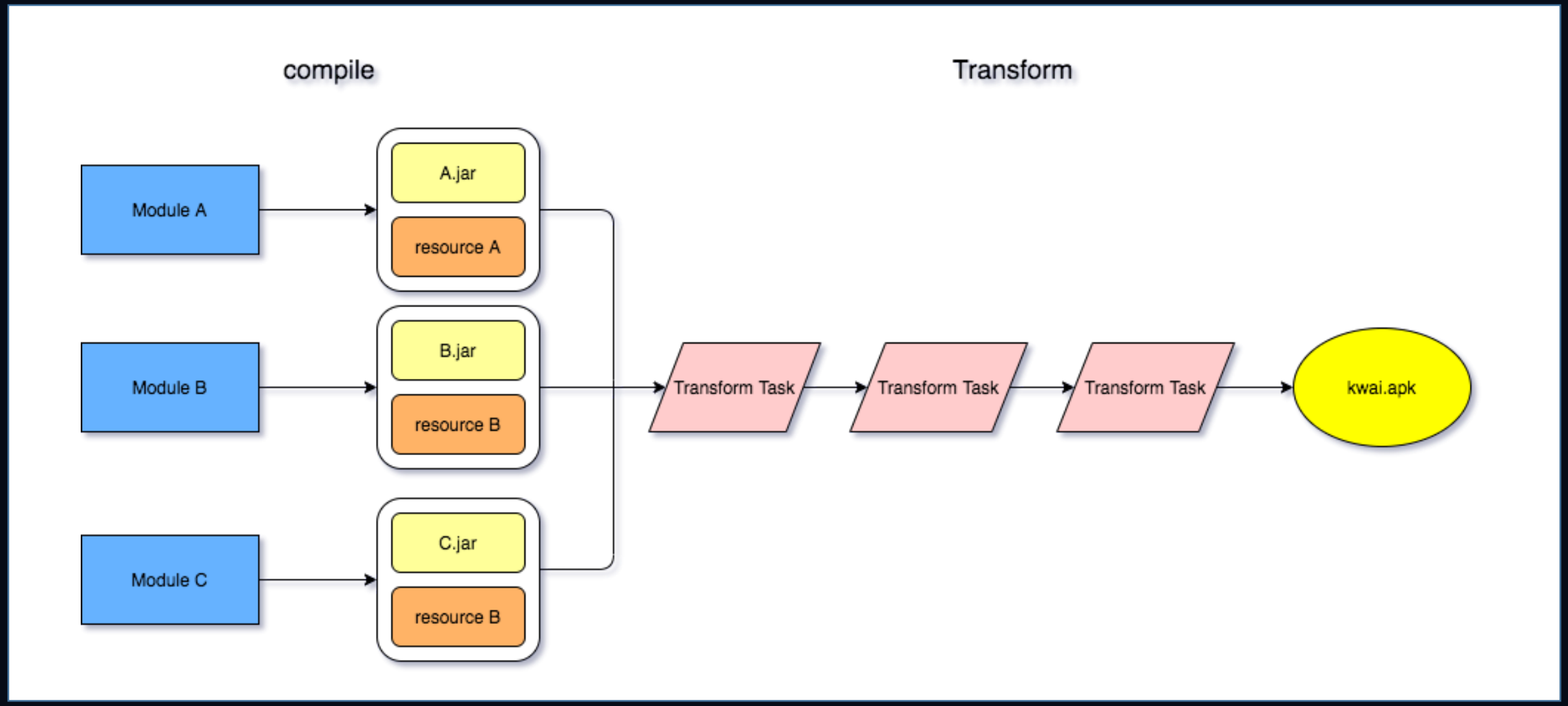
使用更为积极的ABI模式减少重编

全量编译 23 min
增量编译 5 min -> 3 min

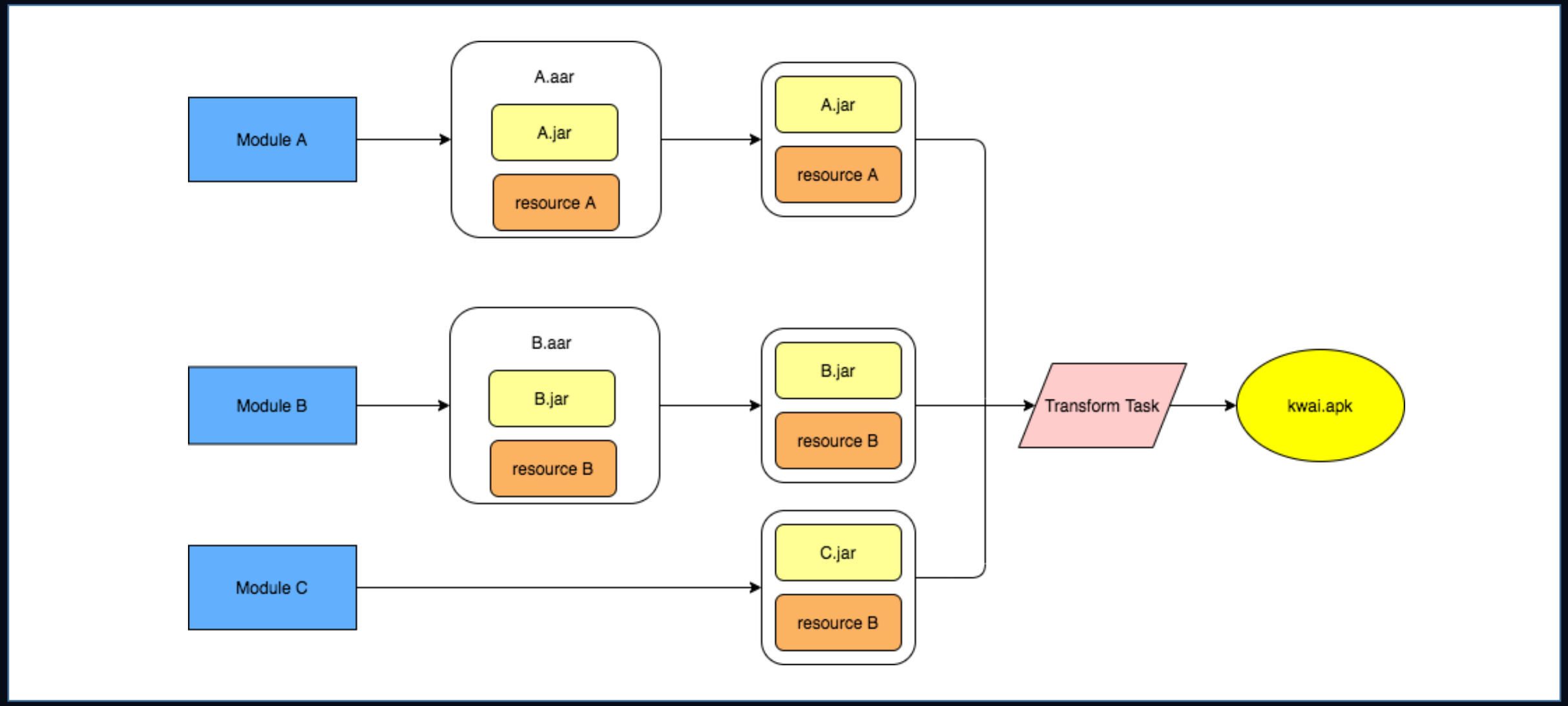
增量编译减少60%的模块重编

transform

动态aar依赖


compile
Transform
Module A
Module B
Module C
A.jar
resource A
B.jar
resource B
C.jar
resource B
Transform Task
Transform Task
Transform Task
kwai.apk

动态aar依赖


Module A
A.aar
A.jar
resource A
A.jar
resource A
Module B
B.aar
B.jar
resource B
B.jar
resource B
Module C
C.jar
resource B
Transform Task
kwai.apk

动态aar依赖

170 projects (1 included build)

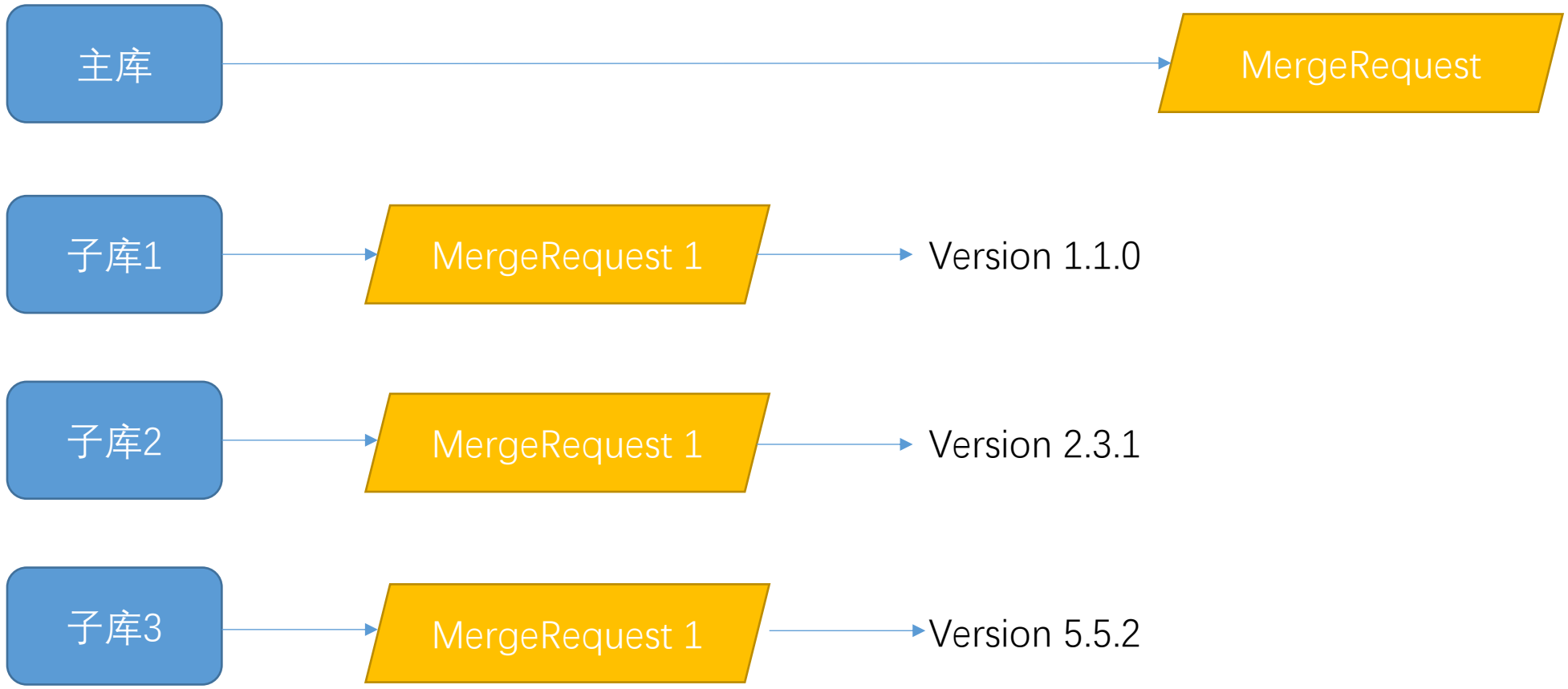
- gifshow-android-0 +8
  - buildSrc build logic
  - configPlugin included build
  - ks-applications +1
  - ks-components +26
  - ks-features +6
  - ks-kernels +20
  - ks-libraries +5
  - spring_2020 +1

170个模块需要拆分

1000万行源码的工作量

乐观估计超过一年的工期

动态aar依赖


主库
MergeRequest
子库1
MergeRequest 1
Version 1.1.0
子库2
MergeRequest 1
Version 2.3.1
子库3
MergeRequest 1
Version 5.5.2

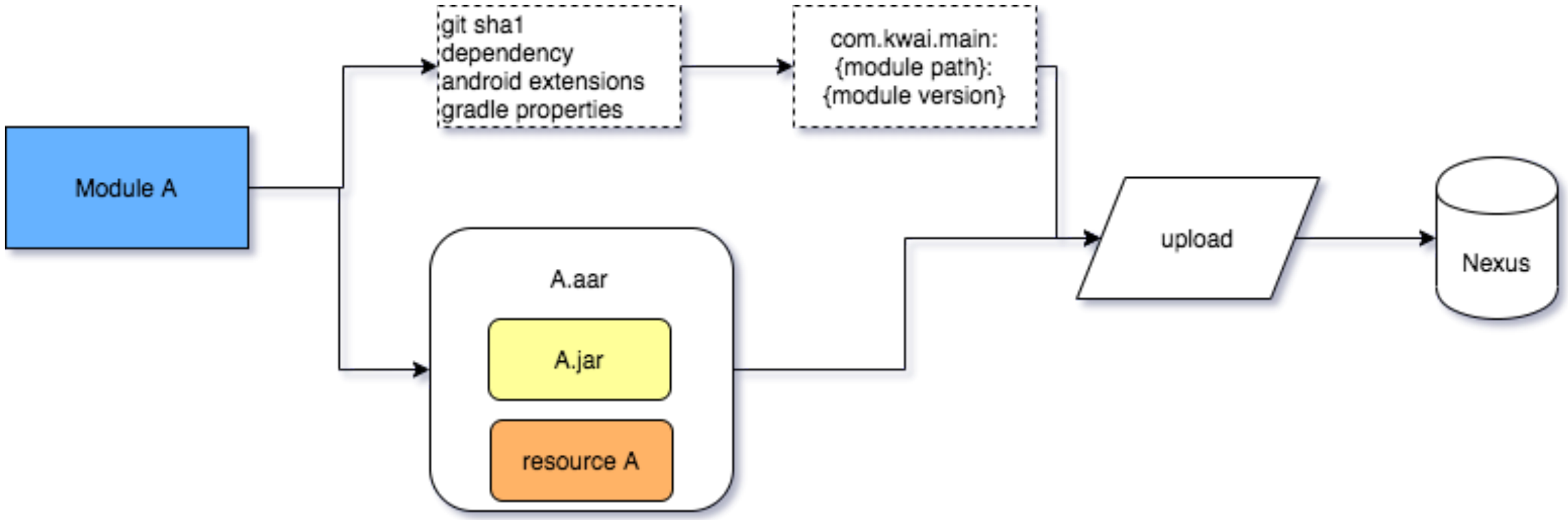
动态aar依赖

所有模块子库化

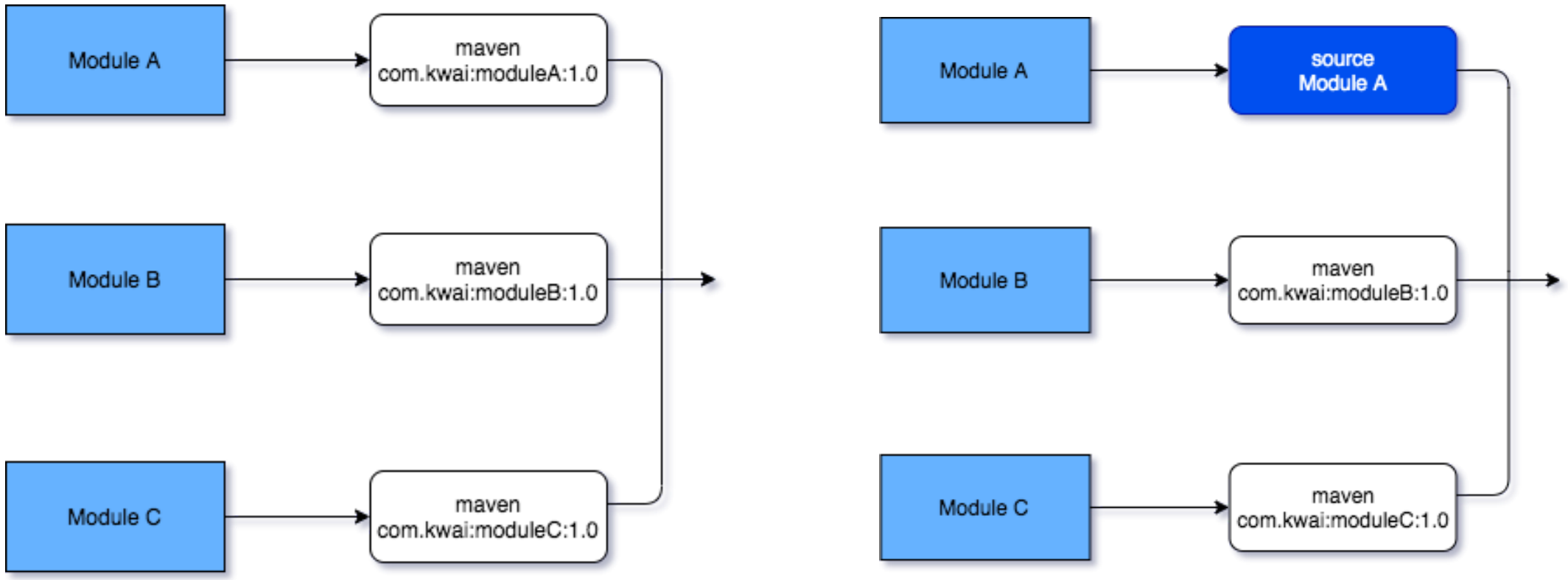
优点：1. 源码模块被提前编译为aar，降低编译时间
2. AndroidStudio无需解析并预编译源码模块，降低性能消耗

缺点：1. 需要消耗大量人力解耦，战线较长
2. 需要改变现有开发流程，N个子库需要提N+1个MR
3. 需要开发配套CI/CD工具链

动态aar依赖


Module A
git sha1
dependency
android extensions
gradle properties
com.kwai.main:
{module path}:
{module version}
A.aar
A.jar
resource A
upload
Nexus

动态aar依赖


Module A
maven
com.kwai:moduleA:1.0
Module B
maven
com.kwai:moduleB:1.0
Module C
maven
com.kwai:moduleC:1.0
Module A
source
Module A
Module B
maven
com.kwai:moduleB:1.0
Module C
maven
com.kwai:moduleC:1.0

编译优化

compile

使用更为积极的ABI模式减少重编

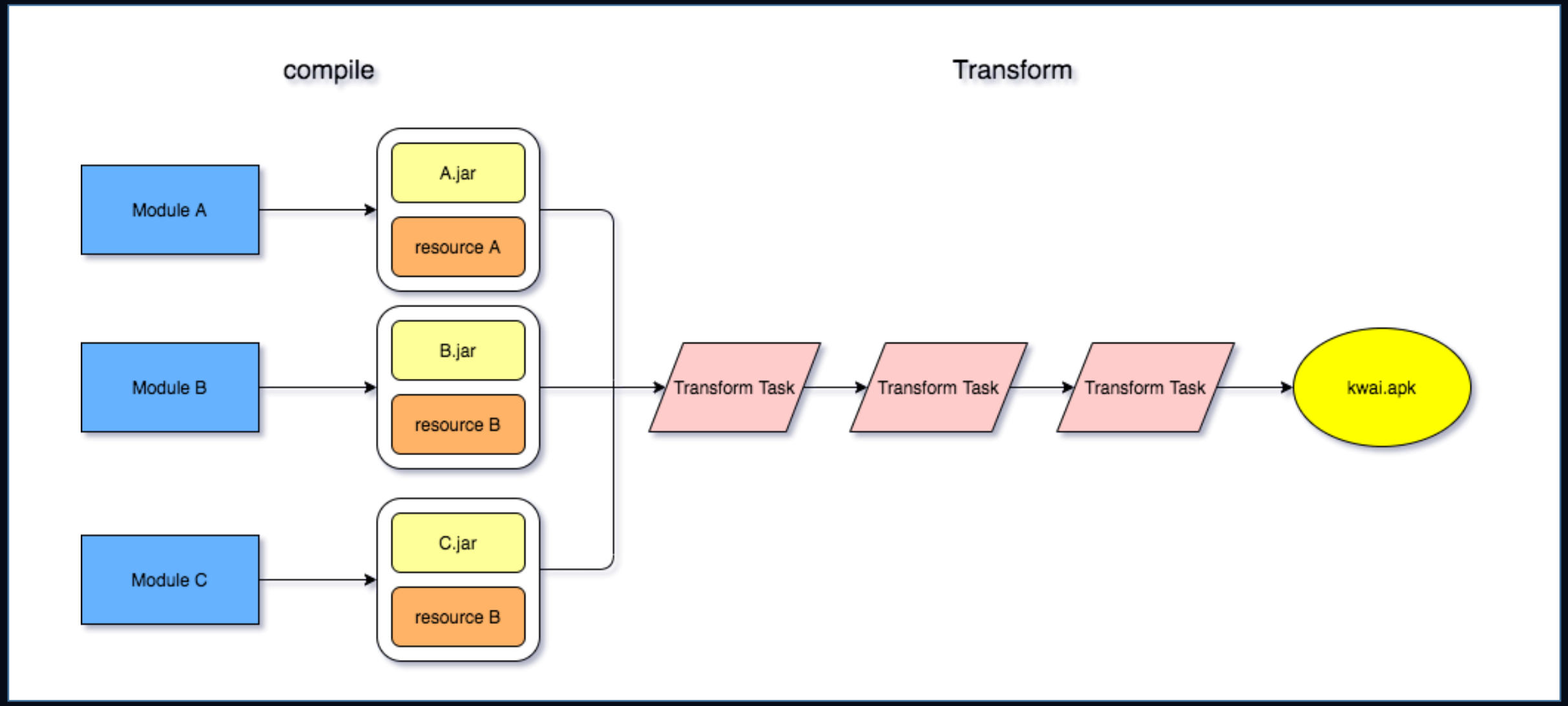
全量编译 23 min
增量编译 5 min -> 3 min

动态aar依赖

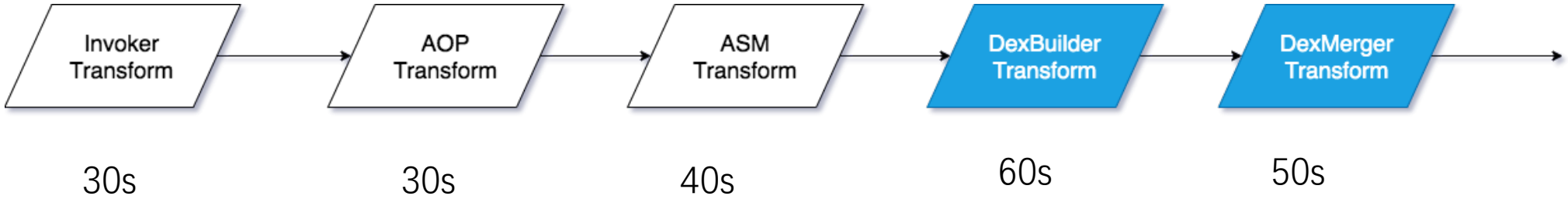
全量编译 23 min -> 15min
增量编译 3 min

transform

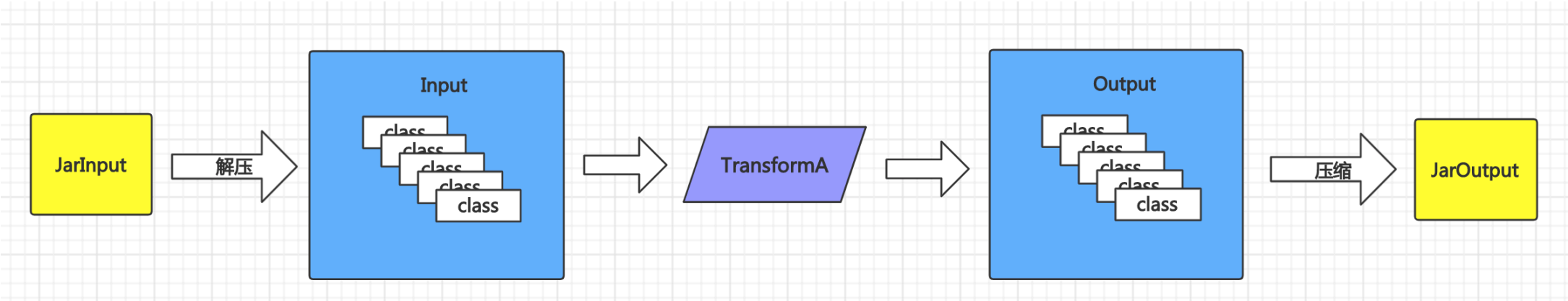
Android编译流程



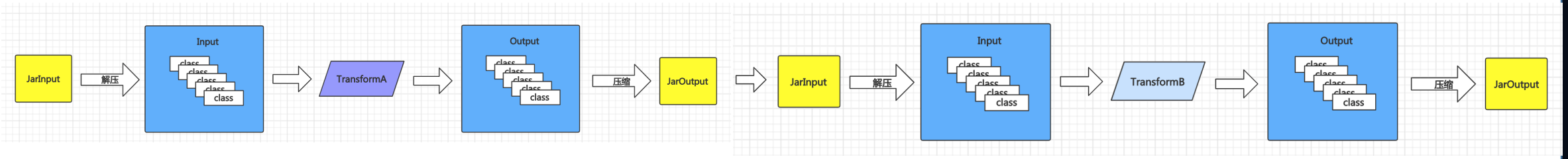
Transform流程



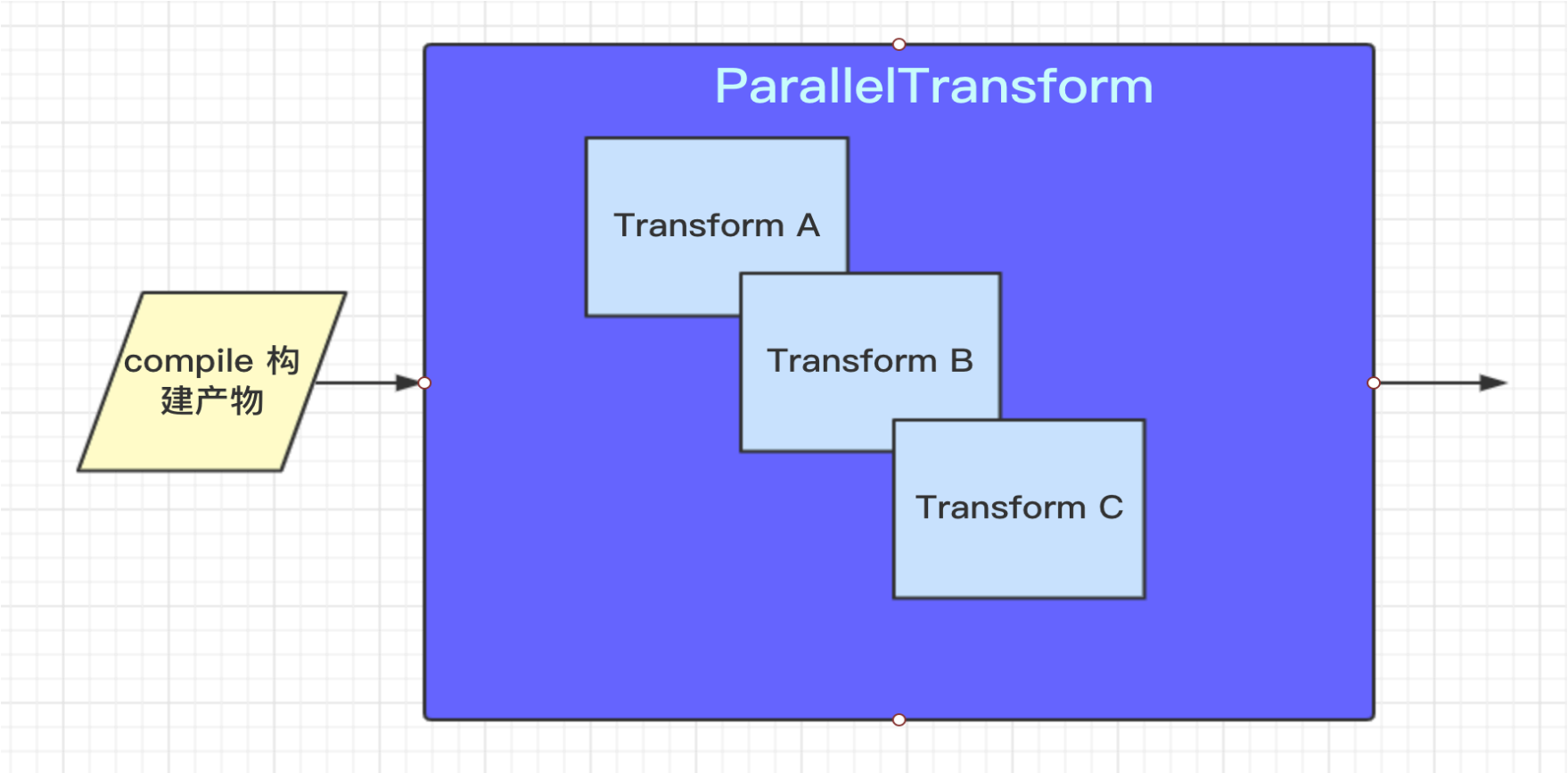
Transform优化



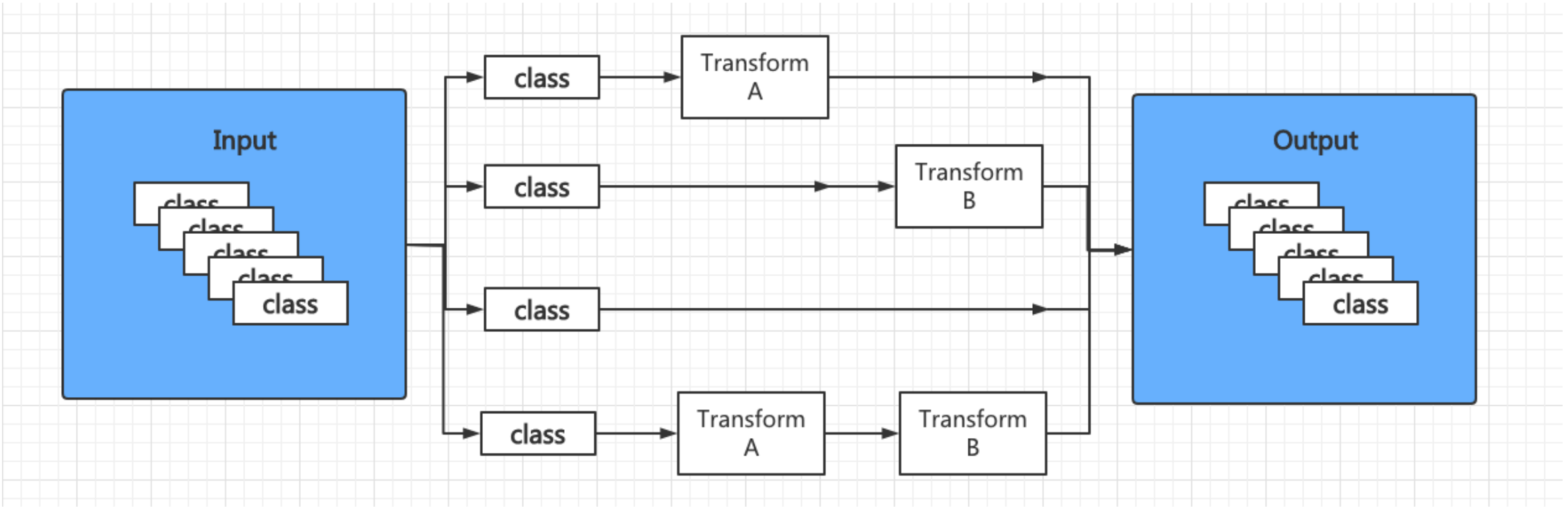
Transform优化



Transform优化

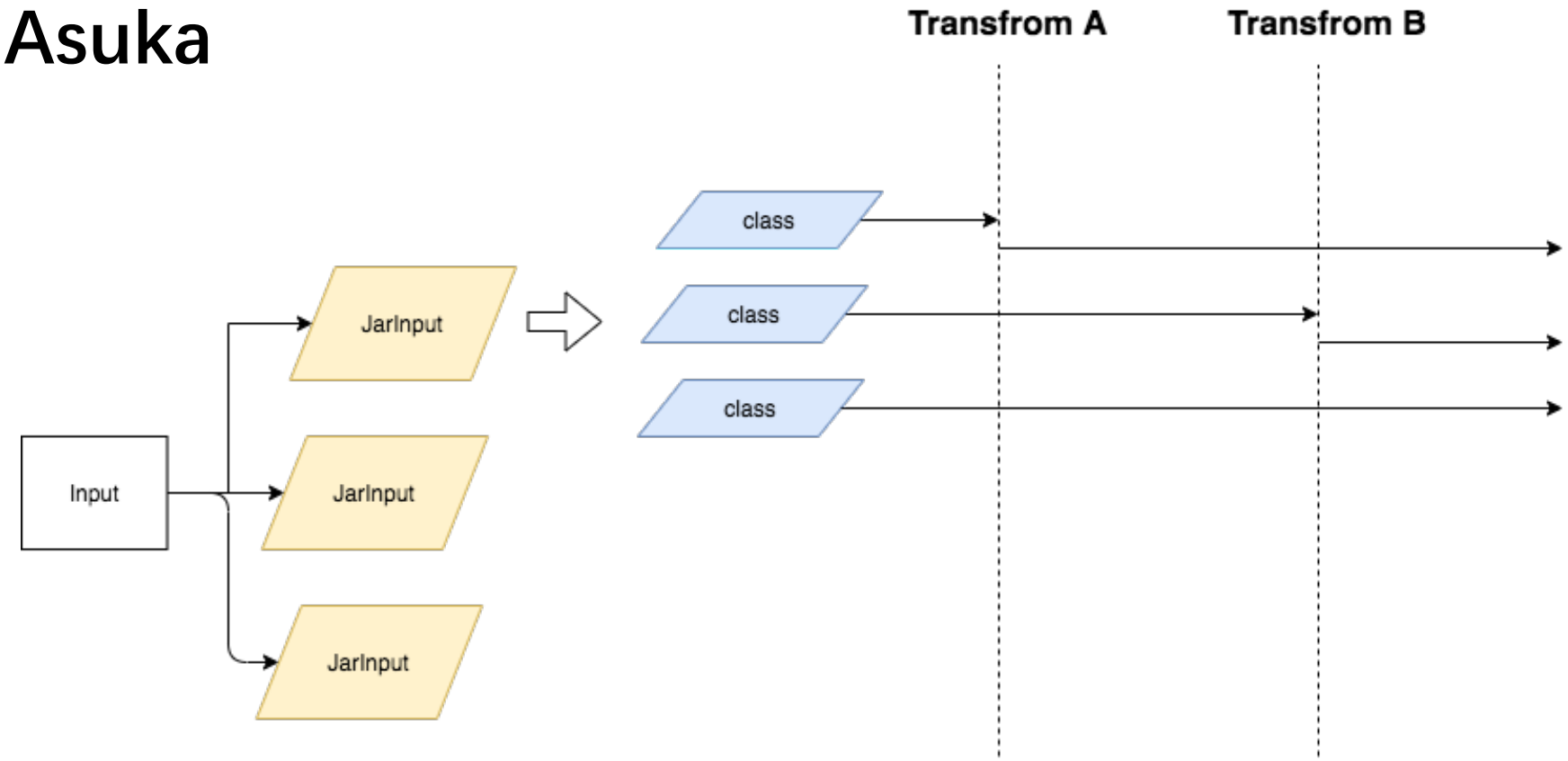


Transform优化

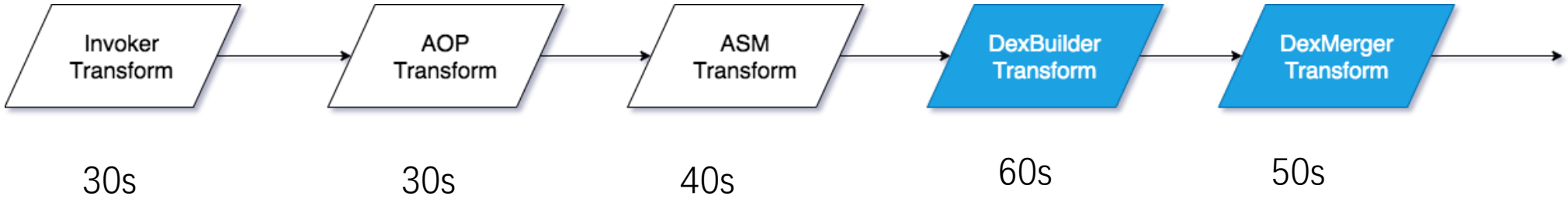


Transform优化

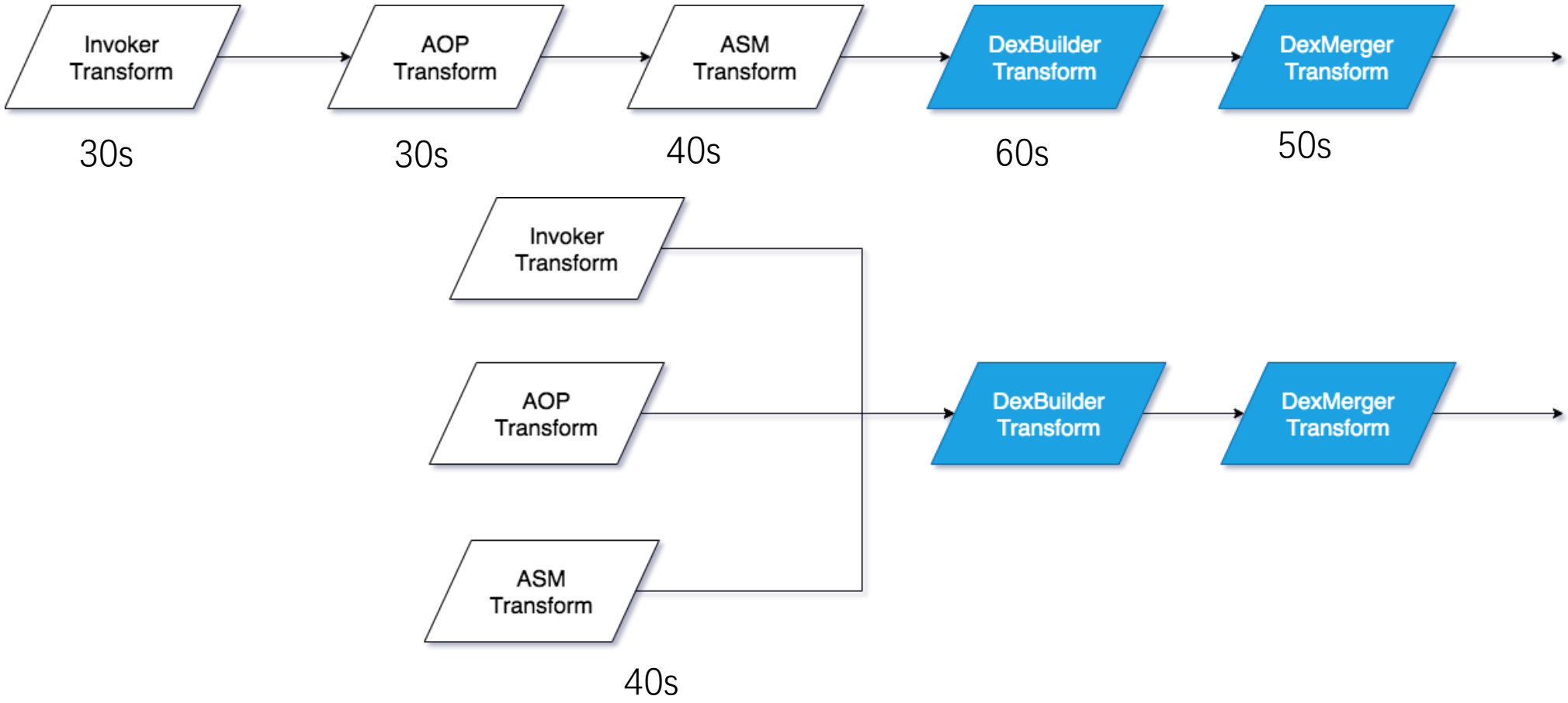
Asuka



Transform流程



Transform流程



编译优化

compile

使用更为积极的ABI模式减少重编

全量编译 23 min
增量编译 5 min -> 3 min

动态依赖aar

全量编译 23 min -> 15min
增量编译 3 min

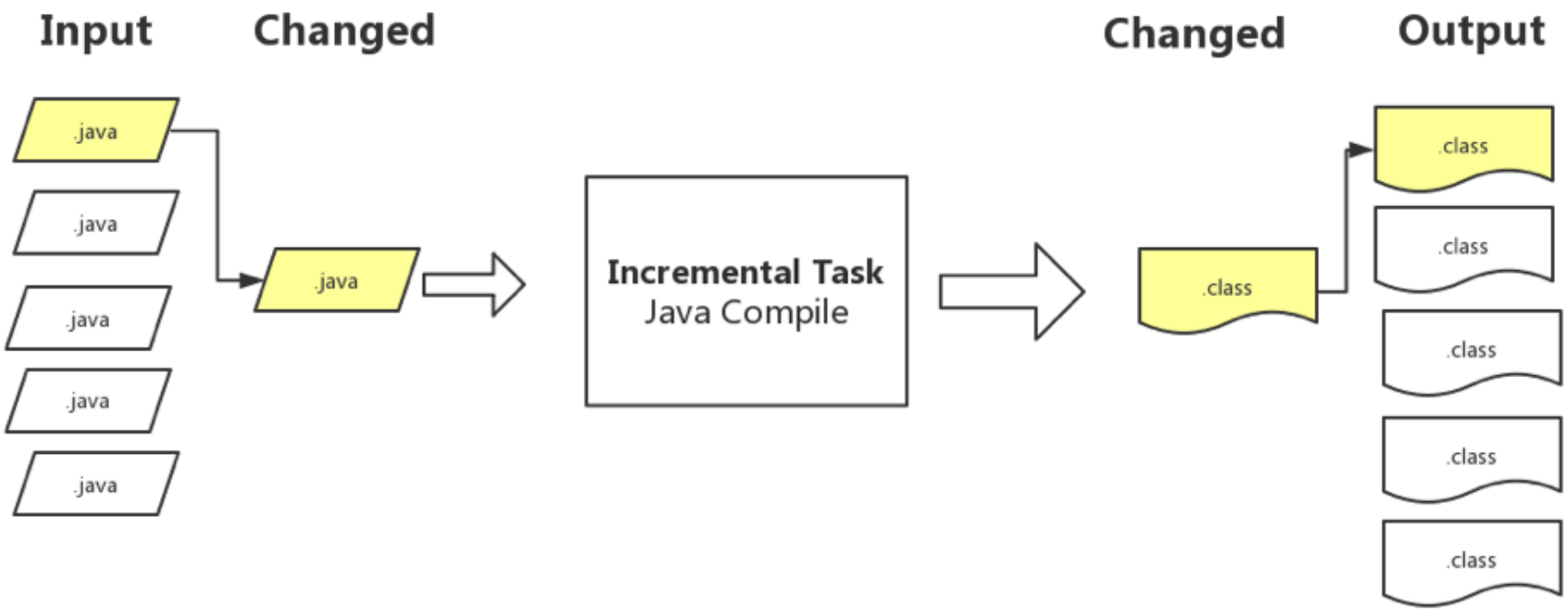
transform

Transform并行

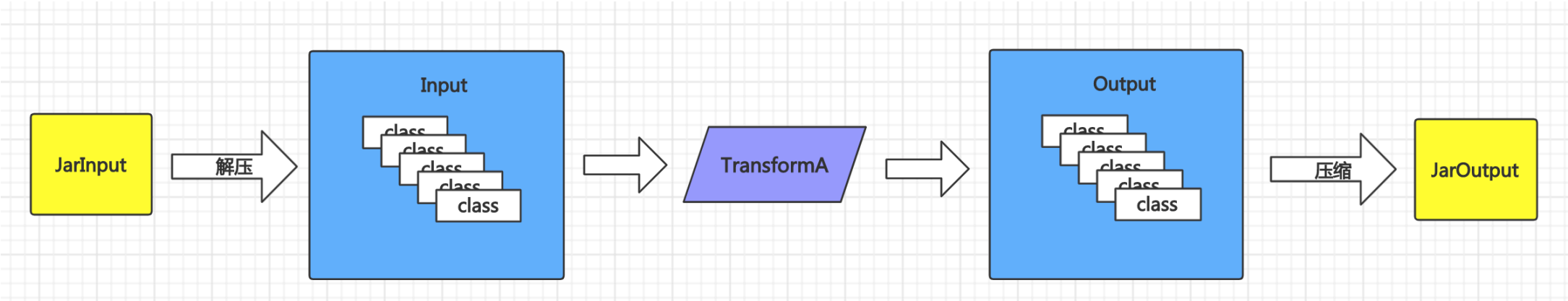
全量编译 15 min -> 10min
增量编译 3 min

Transform增量优化

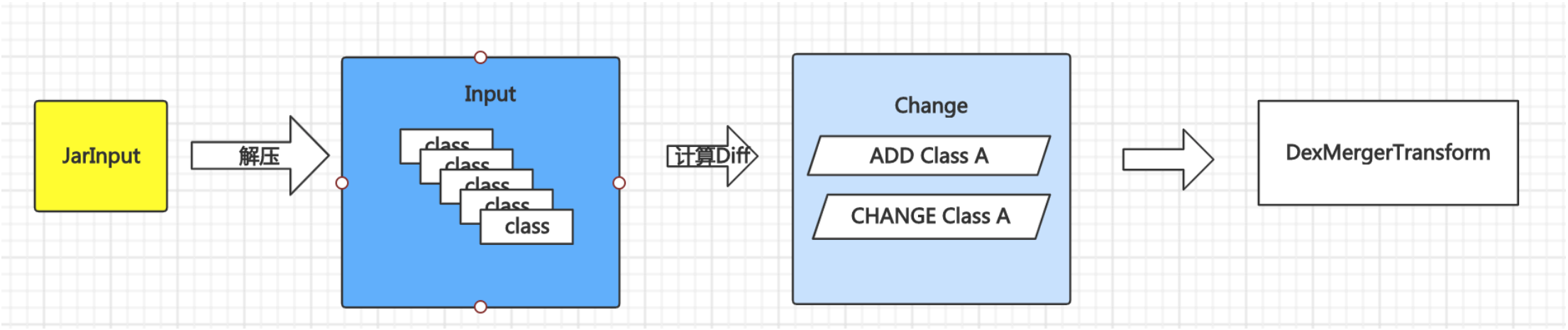
增量编译



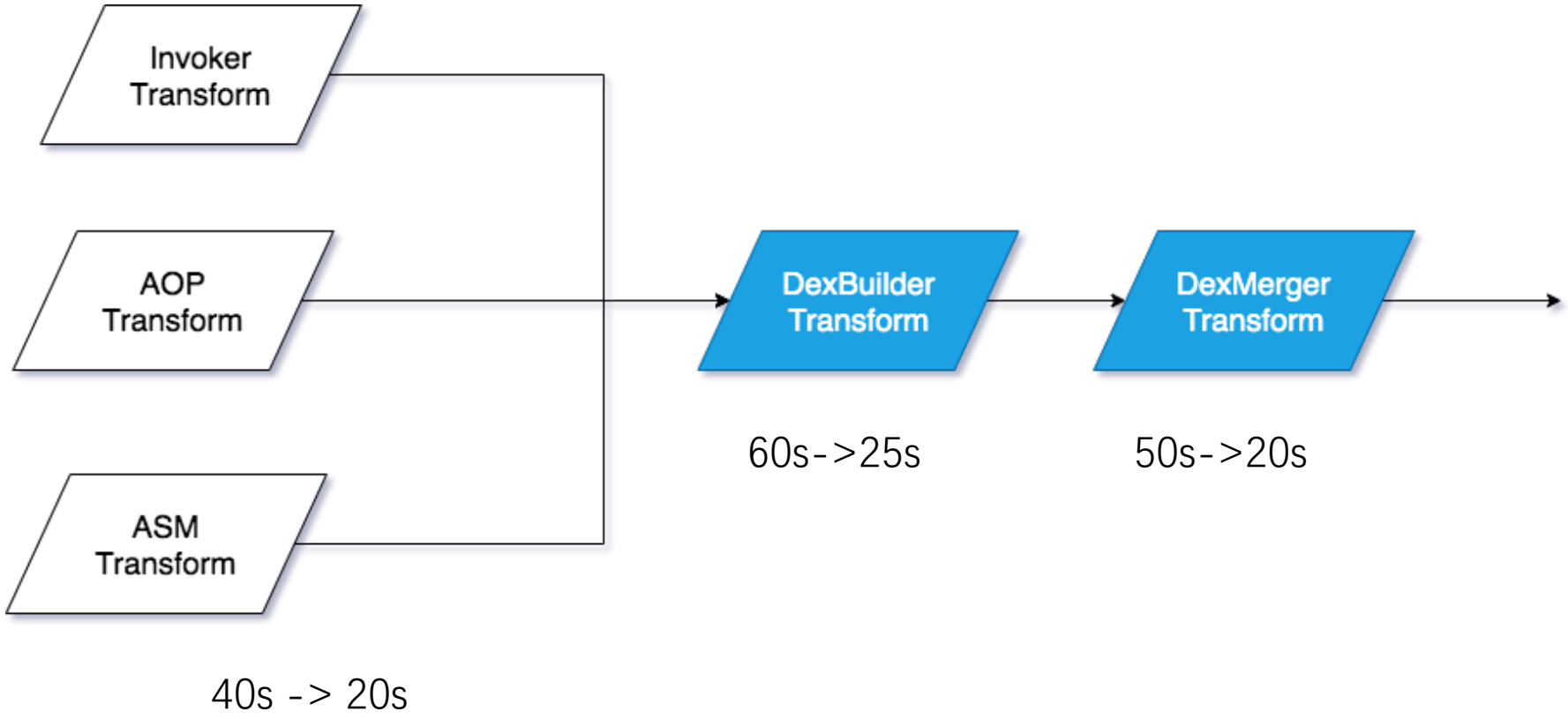
Transform增量优化



Transform优化



Transform增量优化



Transform IO优化

使用协程+NIO处理并行IO操作

降低内核态用户态切换带来的性能损失
IO操作让出CPU，增强CPU计算性能利用率
降低线程锁机制带来的性能损耗

```
private suspend fun unzip(file: File, toDir: File): Map<String, Long> {
  val zip = ZipFile(file)
  val list = LinkedList<Deferred<Pair<String, Long>>>()

  zip.entries().asSequence().forEach { zipEntry ->
    val r = GlobalScope.async(Dispatchers.IO) {
      extraZipEntry(zip, zipEntry, toDir)
    }
    list.add(r)
  }

  val map = list.map {
    it.await()
    it.getCompleted()
  }.toMap()

  try {
    zip.close()
  } catch (e: Exception) {
    e.printStackTrace()
  }

  return map
}
```

编译优化

compile

使用更为积极的ABI模式减少重编

全量编译 23 min
增量编译 5 min -> 3 min

动态依赖aar

全量编译 23 min -> 15min
增量编译 3 min

transform

Transform并行

全量编译 15 min -> 10min
增量编译 3 min

JarInput增量优化

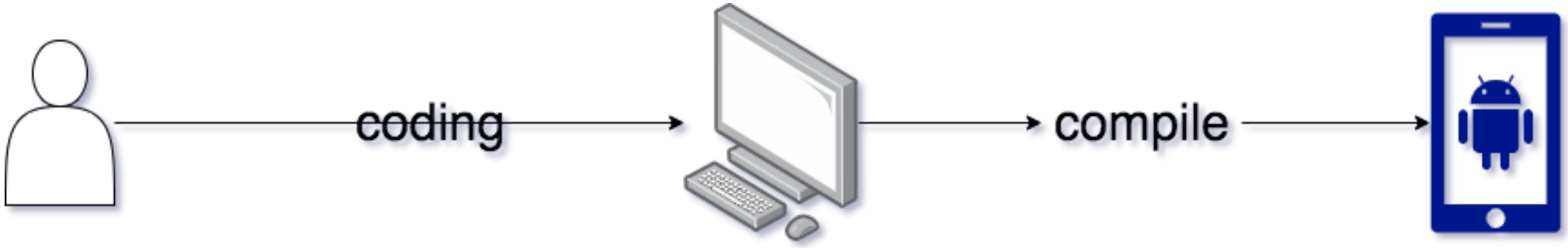
全量编译 10min
增量编译 3 min -> 2min

全量编译
还能再快些吗？

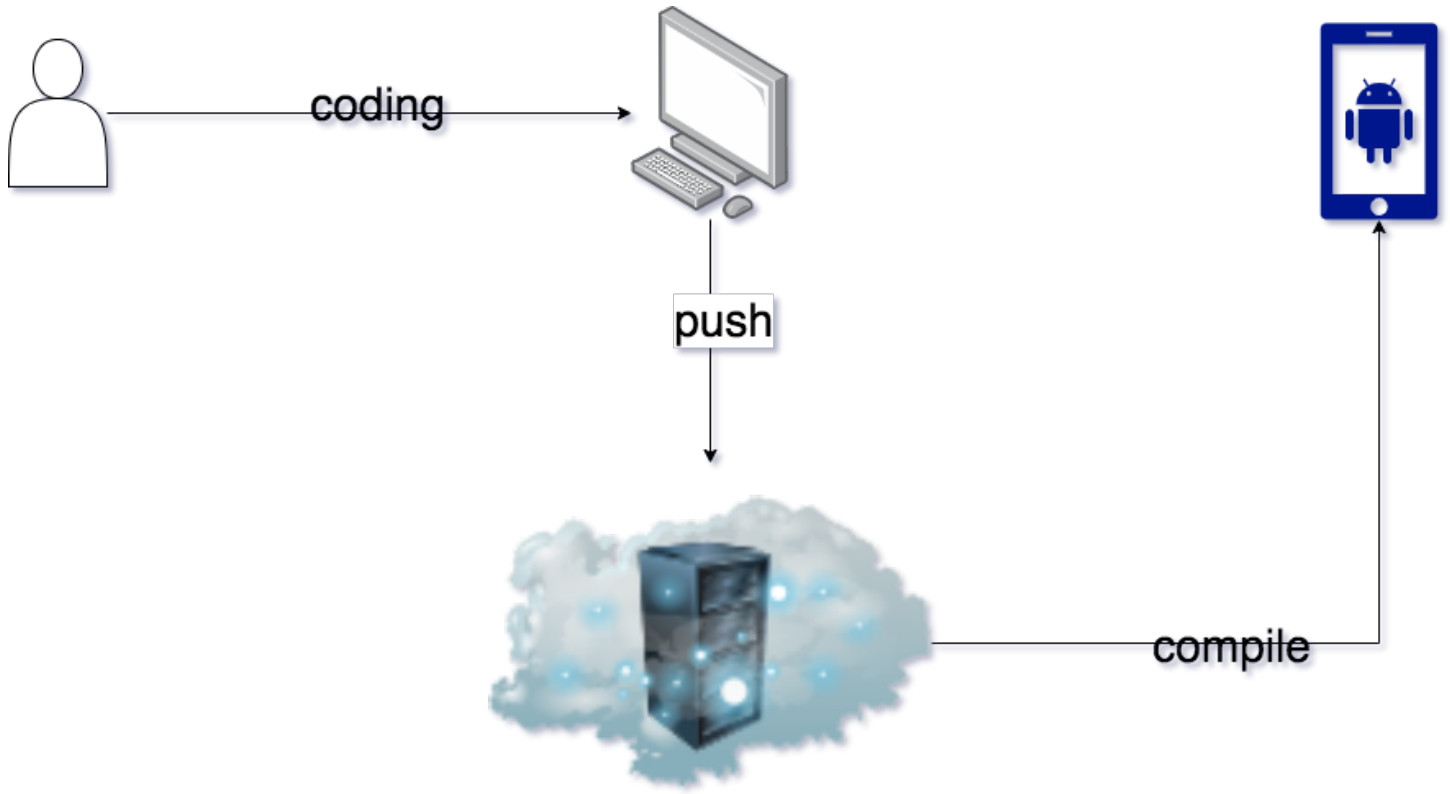
性能不够怎么办？

加机器

远程编译

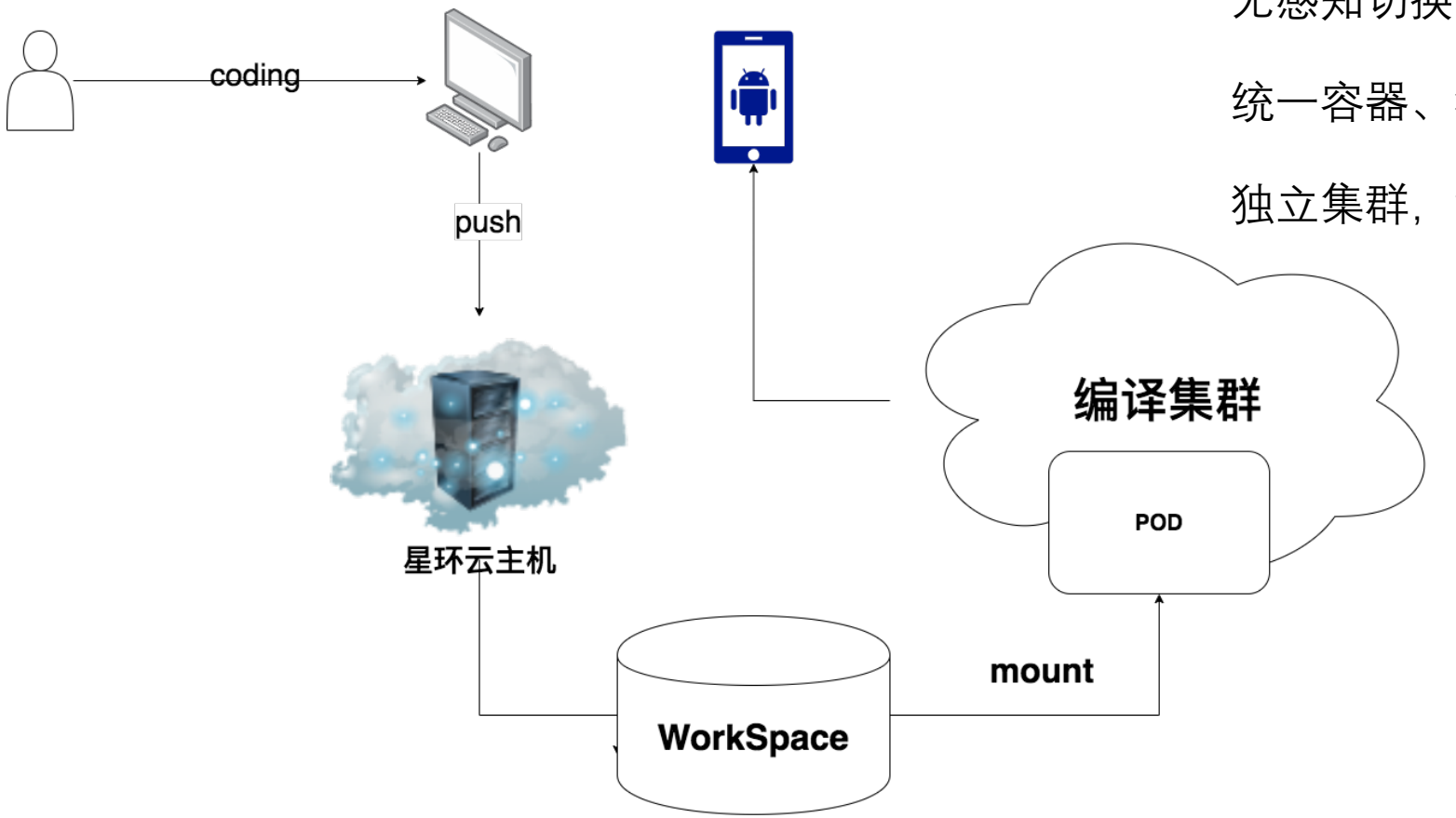


远程编译



星环云主机

远程编译



无感知切换，不改变开发习惯，无学习成本

统一容器、独立实例

独立集群，针对编译任务特别优化

编译优化

compile

使用更为积极的ABI模式减少重编

全量编译 23 min
增量编译 5 min -> 3 min

动态依赖aar

全量编译 23 min -> 15min
增量编译 3 min

远程编译

全量编译 23 min -> 7min
增量编译 5 min -> 1 min

transform

Transform并行

全量编译 15 min -> 10min
增量编译 3 min

JarInput增量优化

全量编译 10min
增量编译 3 min -> 2min

快手大前端
技术交流会 2020

THANKS