Радикальная оптимизация сборки

Вынос сборки Android проектов с машины разработчика на сервер

Обо мне

Павел Васильев

- Специалист по информационной безопасности мобильных приложений в Positive Technologies
- Более 5 лет в разработке под android
- Работаю с необычными возможностями системы, закапываюсь в детали
- Иногда пишу gradle плагины

Сборка на удалённой машине

1. Зачем?
2. Как?
3. Личный опыт
4. Выводы

1

Зачем?

Чтобы собирать код быстрее

Проблемы

Сборка
на слабом железе
производится долго

У удалённщиков
не всегда есть мощное
железо (ноутбук)

Ноутбук
плохо рассчитан
на серьёзную нагрузку

Что раздувает сборку?

- Количество кода
- Количество gradle модулей
- Количество build type и flavor
- Процессинг аннотаций, кодогенерация (dagger, room, etc)
- Зависимости между модулями
- ...

¯_(ツ)_/¯

Важно

! Вынос сборки на сервер — не оптимизация

! В первую очередь оптимизируем сборку
традиционными способами
(JVM properties, gradle.properties, tools versions)

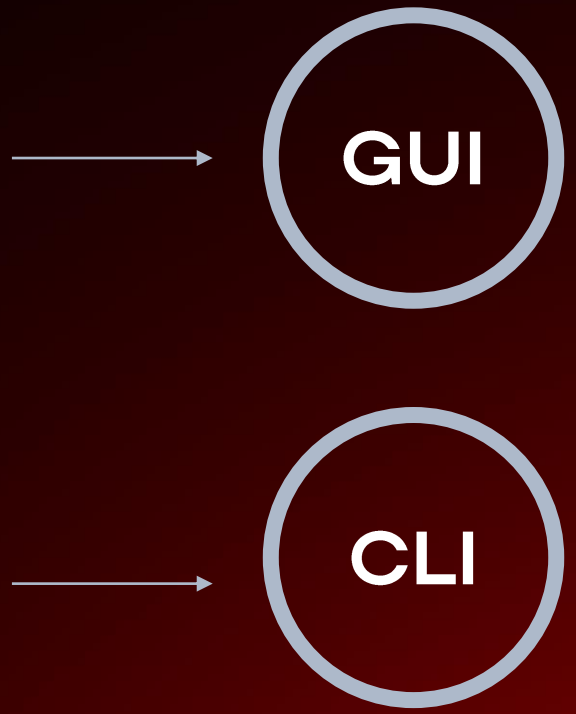
Обозначим требования

Безопасность	никаких облаков, только личный компьютер, только безопасный канал связи
Отказоустойчивость	в случае потери связи необходимо иметь возможность продолжить работу локально
Работа с реальным устройством	отладка по adb, эмуляторы — нежелательны
Интерфейс	работать с GUI, а не CLI

2

Как?

Пути для сборки на другом компьютере



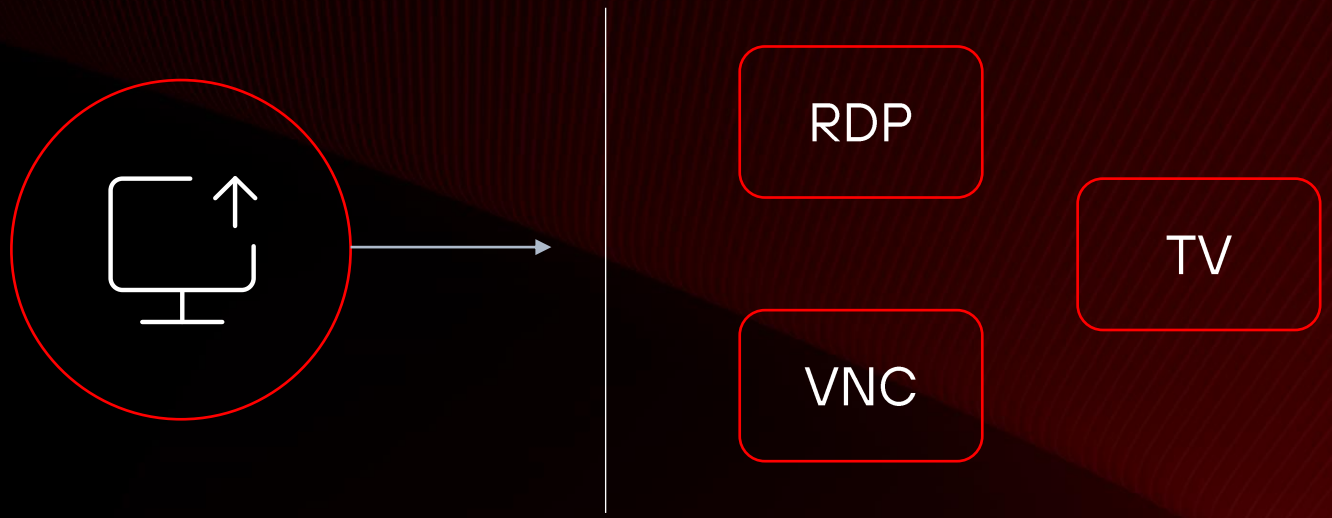
ПУТЬ

Работа с удалённым компьютером через GUI

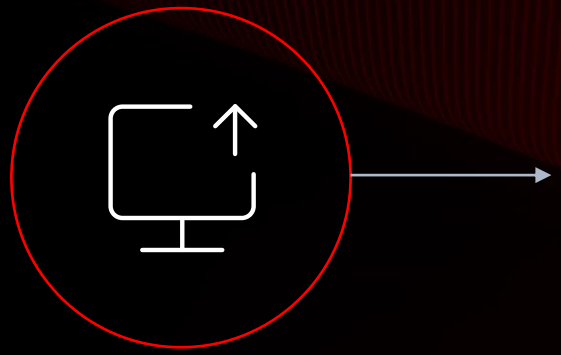
Передача всего экрана

Передача экрана одного окна

Передача всего экрана



Передача всего экрана



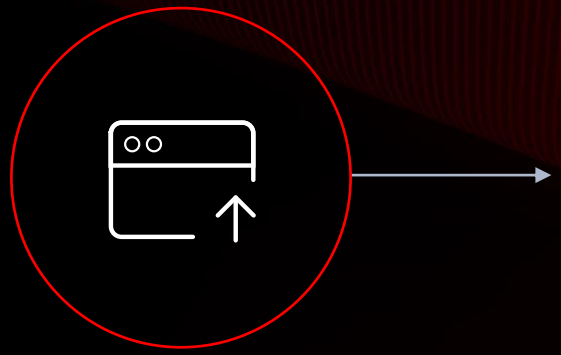
Просто

Ненативный UI

Не работают шорткаты

Оверхед

Передача экрана одного окна



X11forwarding

JetBrains remote development
VSCode remote development

X11Forwarding

- /etc/ssh/sshd_config: X11Forwarding Yes
- ssh -X ...

X11Forwarding

>

Окно в локальном графическом окружении

Работают шорткаты

>

Только для линуксоидов:

(но Xming, Xquartz)

Только для иксов:

(но -Y работает с Wayland)

Производительность

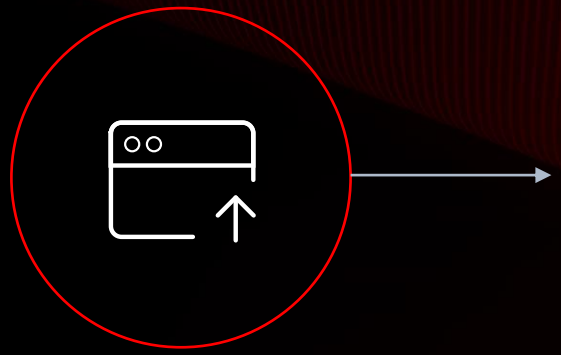
(но x2go)

X2GO

Значительно повышает производительность, уменьшает лаг!

При сильном лаге есть вероятность получать графические артефакты

Передача экрана одного окна



X11forwarding

Jetbrains remote development
VSCode remote development

Jetbrains remote development

- Server: projector
- Client: gateway
- UI through socket

Путь GUI, плюсы

- \+ Полная разгрузка локальной машины
- \+ Простота
- \+ Это будущее

Путь GUI, минусы

- Большая зависимость от качества сети
- Задержка в UI
- Необходимость работать в эмуляторе, ИЛИ держать устройство на конечной машине, ИЛИ прокидывать adb по tcp/ip

Путь GUI, советы

! Отдайте предпочтение профессиональным, хорошо поддерживаемым инструментам: Jetbrains Remote Development, VSCode Remote Development.

! Не надо велосипедов, выигрыша в производительности не будет, UI через иксы будет не очень

ПУТЬ

Сборка в gradle – это просто!

- /path/to/project/gradlew assembleDebug
- ssh user@host /path/to/project/gradlew assembleDebug

Передача файлов

Доставка кода

- git
- scp
- rsync

rsync

- Работает по безопасному соединению ssh
- позволяет эффективно доставить только последние изменения
- rsync не требует лишней синхронизации
- позволяет использовать исключения

Передача файлов

- Пример команды передачи кода:

Важно — только src/

```
rsync -avEWHhz --exclude '*.DS_Store' --exclude '*.patch' --exclude '*.hprof' --exclude '.idea'
--exclude '.gradle' --exclude '**/.git/' --exclude '**/.gitignore' --exclude '**/local.properties' --
exclude '**/build/' --exclude '**/.cxx/' --exclude '**/.externalNativeBuild/' --exclude '*.apk' --
exclude '*.iml' --exclude '*.keystore' -e "ssh" --info=progress2 /path/to/local/project/dir
host:~/path/to/remote/projects/dir
```

Передача файлов

- Пример команды передачи файлов

Важно — только build/output/ (размер директорий)

```
rsync -avEWHhz --exclude '*.DS_Store' --exclude '*.patch' --exclude '*.hprof' --exclude '.idea'
--exclude '.gradle' --exclude '**/.git/' --exclude '**/.gitignore' --exclude '**/local.properties' --
exclude '**/.cxx/' --exclude '**/.externalNativeBuild/' --exclude "**/src/" --exclude
"**/build/.transforms/**" --exclude "**/build/kotlin/**" --exclude "**/build/tmp/**" --exclude
"**/build/intermediates/**" --exclude '*.iml' --exclude '*.keystore' --include
'**/build/intermediates/apk/***' --include '**/build/intermediates/apk_ide_redirect_file/***' -e
"ssh" --info=progress2 --compress-level=9 host:~/path/to/remote/project/dir
/path/to/local/projects/dir
```

Подготовка окружения

Подготовка окружения

OS

- JDK
- переменные окружения
- cmdline-tools
- sdkmanager

Подготовка окружения

sdkmanager

- build-tools;33.0.2
- platform-tools
- platforms;android-33
- ndk;26.2.11394342
- cmake;3.22.1

Подготовка окружения

project

- local.properties (android sdk path)
- Готово!

Готовые решения

Mainframer

- ./gradlew build
- Mainframer ./gradlew build

Mirakle

Gradle init scripts

Применяются ко всем проектам во время старта демона

Не трогают исходный код

Могут применить gradle plugin выше AGP

Mirakle

./gradlew assembleDebug :

uploadToRemote

executeOnRemote

downloadFromRemote

Mirakle

./gradlew assembleDebug :

uploadToRemote	->	rsync local src to remote
executeOnRemote	->	ssh gradlew assembleDebug
downloadFromRemote	->	rsync remote output to local

Mirakle

Преимущества

- Вся коммуникация под капотом - исключительно по ssh
- Не требуется никак менять исходный код собираемого проекта
- Нативный локальный UI, никакой сетевой задержки

Улучшение опыта работы CLI пути

Улучшение опыта работы CLI пути

- Mirakle требует составления внешнего файла конфигурации под каждый проект
- Удалённую тачку приходится настраивать руками под каждый отдельный проект
- Mirakle применяется ко всем gradle проектам на локальной машине сразу. Необходимо постоянно включать и отключать его при работе на каждом нужном/ненужном проекте

Улучшение опыта работы CLI пути

- Все минусы решаемы, но требуют внешнего вмешательства
- Внешнее вмешательство
  - скрипты (под каждую ОС отдельно, забивание части данных в них руками под каждый проект, нужно лезть в терминал)
  - IDE plugin

3

Личный опыт

Плагин для Android Studio

Дисклеймер

- Написание плагинов к платформе Intellij Idea — отдельная история, которой касаться здесь и сейчас не будем
- Я буду описывать логику совершаемых действий, по сути — она просто реализуется в Java коде

Требования к функционалу

>

Хочу одной кнопкой включать и отключать mirakle

>

Хочу автоматизировать настройку удалённой тачки под проект и не лазить руками в ssh

Зависимости для работы плагина

- ssh
- rsync

но есть нюанс... (homebrew, cygwin)

Компоненты плагина

ПАНЕЛЬ С КНОПКАМИ

переключить режим удалённой сборки

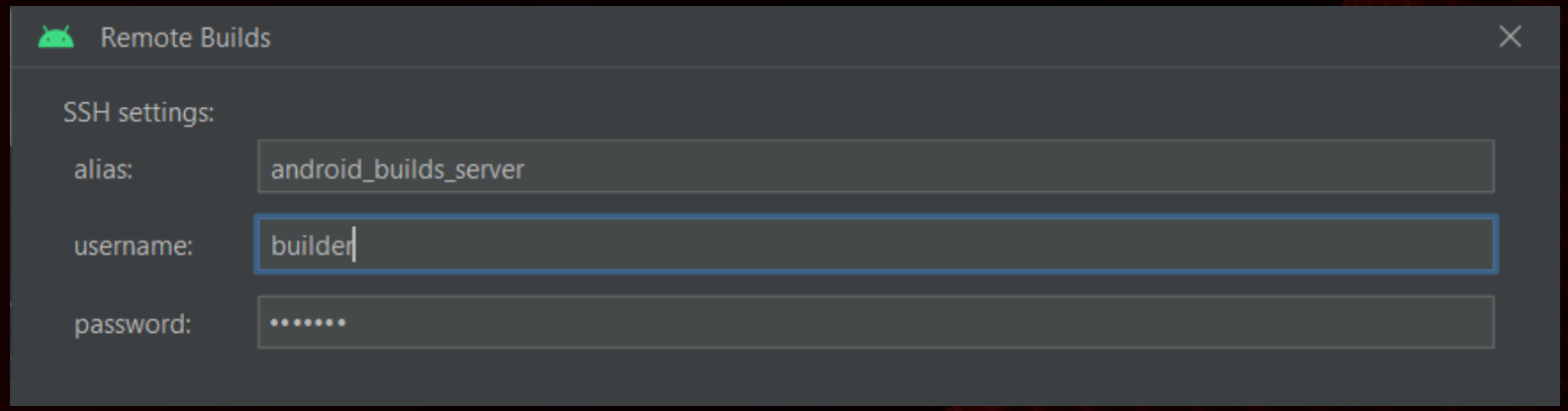
настройки плагина

приготовить удалённый хост к сборке проекта

Настройки плагина

Ssh alias
для подключения

Логин/пароль
от учётной записи



Подготовка окружения хоста к сборке, общая

- В коде плагина получаем доступ к состоянию корневого gradle проекта, проходимся по проектам, собираем версии:
  - jdk
  - android build-tools
  - android platform (target sdk version)
  - ndk
  - cmake
- Собираем другие общие данные: абсолютный путь к проекту, название и т.д.

Подготовка окружения к сборке, общая

- В коде плагина вызываем подключение по ssh к серверу сборки используя креды из настроек
- Смотрим, установлены ли общие пакеты необходимые для сборки (curl, wget, unzip, etc), если нет — устанавливаем
- Смотрим, установлена ли jdk нужной для проекта версии, если нет — устанавливаем
- Смотрим, установлены ли необходимые переменные окружения (JAVA_HOME, ANDROID_HOME, PATH), если нет — устанавливаем
- Обращаемся к sdkmanager-у, смотрим, приняты ли лицензии, если нет — принимаем
- Смотрим, установлены ли cmdline-tools, если нет — выдёргиваем курлом урл с сайта гугла, качаем вгетом, распаковываем, копируем по нужному пути)
- Обращаемся к sdkmanager-у, смотрим, установлены ли необходимые для нашего проекта зависимости (platform-tools, platform, build-tools, ndk, cmake, etc.). Если нет — устанавливаем
- Удалённая тачка готова к сборке (повторять при обновлении версий или на новых проектах)

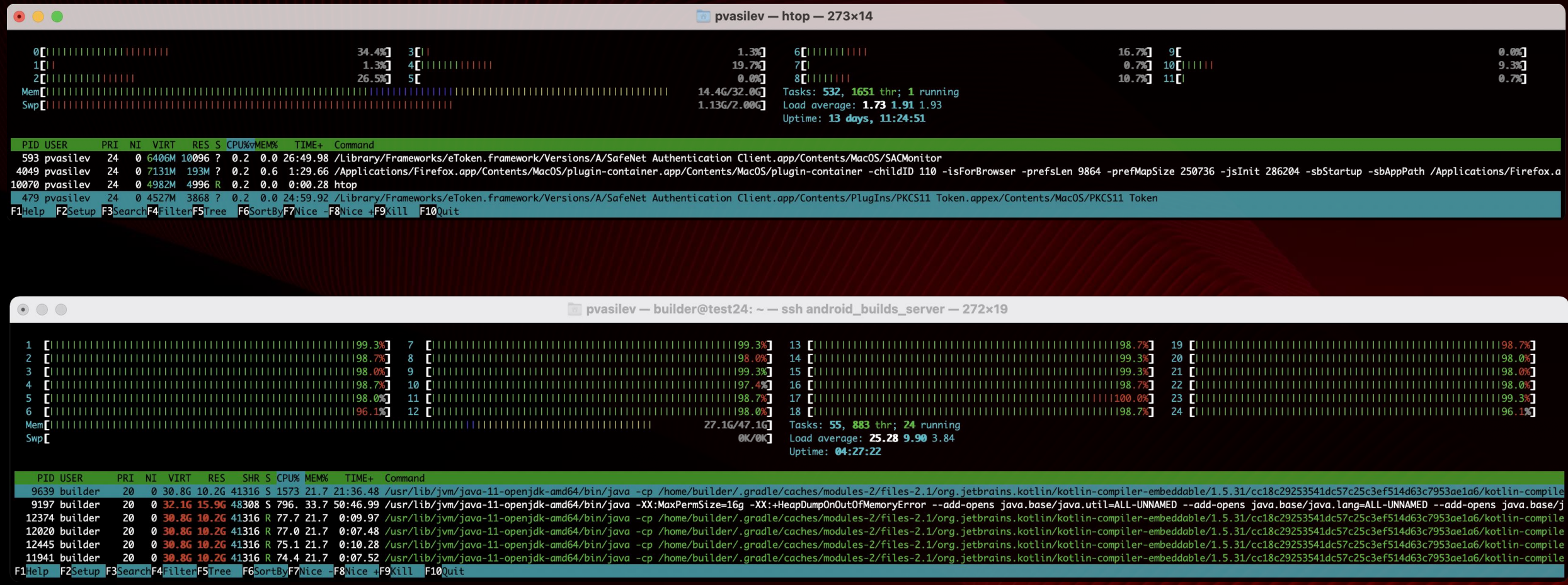
Включение режима сборки проекта на сервере

- В коде плагина получаем доступ к текущим настройкам mirakle
- Смотрим: если проект если не активирован для удалённой сборки — включить удалённую сборку для этого проекта
- Создаём текстовый файл gradle init script-а с путём до текущего проекта, наполняем его по шаблону настройками для нашего проекта, сохраняем
- Подключаемся по ssh к серверу сборки, создаём директорию под проект, создаём в ней файл local.properties с путём до android sdk и другими настройками
- Передаём по ssh на сервер сборки файл .android/debug.keystore и заменяем им существующий

Отключение режима сборки на сервере

- В коде плагина получаем доступ к текущим настройкам mirakle
- Смотрим: если проект активирован для сборки на сервере, значит необходимо режим выключить, файл с настройками удалить
- Удаляем gradle init script с настройками mirakle для проекта

Используем плагин


pvasilev — htop — 273×14
0[34.4%] 3[1.3%] 6[16.7%] 9[0.0%]
1[1.3%] 4[19.7%] 7[0.7%] 10[9.3%]
2[26.5%] 5[0.0%] 8[10.7%] 11[0.7%]
Mem[14.4G/32.0G]
Swp[1.13G/2.00G]
Tasks: 532, 1651 thr; 1 running
Load average: 1.73 1.91 1.93
Uptime: 13 days, 11:24:51
PID USER PRI NI VIRT RES S CPU% MEM% TIME+ Command
593 pvasilev 24 0 6406M 10096 ? 0.2 0.0 26:49.98 /Library/Frameworks/eToken.framework/Versions/A/SafeNet Authentication Client.app/Contents/MacOS/SACMonitor
4049 pvasilev 24 0 7131M 193M ? 0.2 0.6 1:29.66 /Applications/Firefox.app/Contents/MacOS/plugin-container.app/Contents/MacOS/plugin-container -childID 110 -isForBrowser -prefsLen 9864 -prefMapSize 250736 -jsInit 286204 -sbStartup -sbAppPath /Applications/Firefox.a
10070 pvasilev 24 0 4982M 4996 R 0.2 0.0 0:00.28 htop
479 pvasilev 24 0 4527M 3868 ? 0.2 0.0 24:59.92 /Library/Frameworks/eToken.framework/Versions/A/SafeNet Authentication Client.app/Contents/PlugIns/PKCS11 Token.appex/Contents/MacOS/PKCS11 Token
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

pvasilev — builder@test24: ~ — ssh android_builds_server — 272×19
1 [99.3%] 7 [99.3%] 13 [98.7%] 19 [98.7%]
2 [98.7%] 8 [98.0%] 14 [99.3%] 20 [98.0%]
3 [98.0%] 9 [99.3%] 15 [99.3%] 21 [98.0%]
4 [98.7%] 10 [97.4%] 16 [98.7%] 22 [98.0%]
5 [98.0%] 11 [98.7%] 17 [100.0%] 23 [99.3%]
6 [96.1%] 12 [98.0%] 18 [98.7%] 24 [96.1%]
Mem[27.1G/47.1G]
Swp[0K/0K]
Tasks: 55, 883 thr; 24 running
Load average: 25.28 9.90 3.84
Uptime: 04:27:22
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
9639 builder 20 0 30.8G 10.2G 41316 S 1573 21.7 21:36.48 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/builder/.gradle/caches/modules-2/files-2.1/org.jetbrains.kotlin/kotlin-compiler-embeddable/1.5.31/cc18c29253541dc57c25c3ef514d63c7953ae1a6/kotlin-compile
9197 builder 20 0 32.1G 15.9G 48308 S 796. 33.7 50:46.99 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -XX:MaxPermSize=16g -XX:+HeapDumpOnOutOfMemoryError --add-opens java.base/java.util=ALL-UNNAMED --add-opens java.base/java.lang=ALL-UNNAMED --add-opens java.base/j
12374 builder 20 0 30.8G 10.2G 41316 R 77.7 21.7 0:09.97 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/builder/.gradle/caches/modules-2/files-2.1/org.jetbrains.kotlin/kotlin-compiler-embeddable/1.5.31/cc18c29253541dc57c25c3ef514d63c7953ae1a6/kotlin-compile
12020 builder 20 0 30.8G 10.2G 41316 R 77.0 21.7 0:07.48 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/builder/.gradle/caches/modules-2/files-2.1/org.jetbrains.kotlin/kotlin-compiler-embeddable/1.5.31/cc18c29253541dc57c25c3ef514d63c7953ae1a6/kotlin-compile
12445 builder 20 0 30.8G 10.2G 41316 R 75.1 21.7 0:10.28 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/builder/.gradle/caches/modules-2/files-2.1/org.jetbrains.kotlin/kotlin-compiler-embeddable/1.5.31/cc18c29253541dc57c25c3ef514d63c7953ae1a6/kotlin-compile
11941 builder 20 0 30.8G 10.2G 41316 R 74.4 21.7 0:07.52 /usr/lib/jvm/java-11-openjdk-amd64/bin/java -cp /home/builder/.gradle/caches/modules-2/files-2.1/org.jetbrains.kotlin/kotlin-compiler-embeddable/1.5.31/cc18c29253541dc57c25c3ef514d63c7953ae1a6/kotlin-compile
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

Преимущества плагина IDE

Устранена ручная возня с настройкой mirakle

Переключение режимов в одно нажатие кнопки

Единообразная работа в любой ОС

Недоработки плагина IDE

- Минимальная версия платформы Intellij — 2021.3 (dolphin)
- В силу специфики, ожидается что все gradle projects будут в одной корневой директории, а не в нескольких
- На windows требуются дополнительные шаги настройки

Путь CLI, плюсы

- \+ Простое решение с минимумом потенциальных точек отказа
- \+ Низкая зависимость от качества и стабильности сети
- \+ Всё на открытых технологиях (безопасность, доверие, контроль)
- \+ Удобство работы с отладкой на локальном устройстве
- \+ Локальный UI без задержек

Путь CLI, минусы

- Часть нагрузки остаётся на локальной машине
- Требуются ОС-специфичные оптимизации на локальной машине
- Новое бутылочное горлышко — сеть, требуется экспериментировать со скоростью передачи по ssh

Путь CLI, советы

!

Попробуйте плагин :)

https://github.com/LuigiVampa92/SkyForge

Результаты сборки в моём проекте

Результаты сборки

Пример конфигурации проекта

- ~2M LOC на Java/Kotlin
- 500+ gradle модулей
- 3 build type, 3 flavor
- Много либ с кодогенерацией: dagger, room и т.д.

Результаты сборки

	БЫЛО:	СТАЛО:
Холодная	30-40 минут	9-10 минут
Переключение веток	10-15 минут	4-6 минут
Горячая	3-5 минут	< 1 минуты **(но есть нюанс)**
Железо	MBP 16' 2019, Intel core i9-9880H, 32GB RAM	AMD Ryzen 5950, 64GB RAM
ОС	MacOS 11	Ubuntu 22

4

Выводы

Выводы

Оба пути рабочие, лично я выбрал CLI

Идеального решения пока не существует

Выбирайте вариант под ваши нужды и требования

Могу ли я рекомендовать сборку на сервере?

- It depends
- Не рекомендую относиться к этому как к простому способу оптимизации
- Подумайте, есть ли у вас необходимые серверные мощности
- Помните о конфиденциальности кода
- Трезво оцените плюсы и минусы. Некоторые из них не такие уж и несущественные (сеть)
- Лучший кейс в моём представлении - только для ноутбуков

https://github.com/LuigiVampa92/SkyForge

Спасибо!