Как мы гоняем Android UI Автотесты на 20 игровых десктопах, и почему нам понравилось

kaspersky BRING ON THE FUTURE

Меня зовут

Сергей Павлов

- В автоматизации с 2016 года
- Android + iOS автоматизация на Appium + ruby/java
- XCUITest автоматизация на Swift

Какие приложения вообще тестируем?

- Kaspersky Internet Security для Android
- Kaspersky Endpoint Security для Android
- Kaspersky Safe Kids
- Kaspersky Password Manager
- Kaspersky Secure Connection
- Kaspersky Security Cloud
- Kaspersky Who Calls

О чём мы поговорим сегодня?

- О проблемах запуска эмуляторов на голом железе
- Как докер решил наши проблемы
- Почему мы выбрали игровые десктопы

Какие тесты есть?

Unit

Integration

UI

Многообразие используемых технологий в UI

Appium + ruby

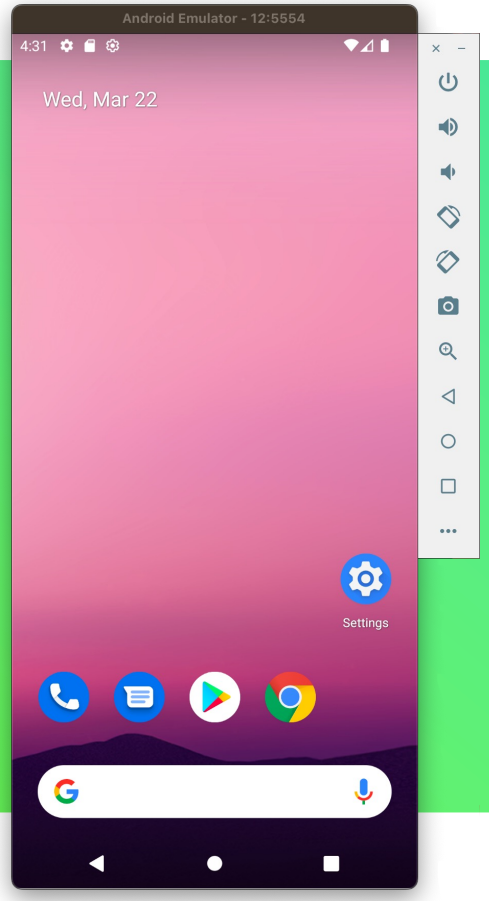
Appium + java

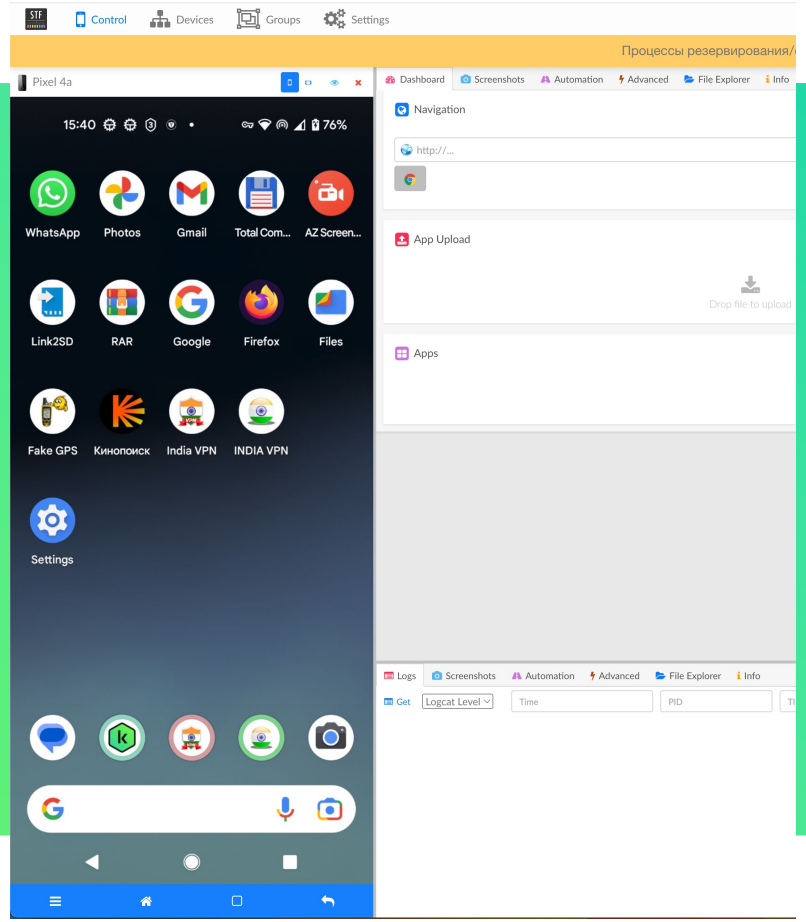
Appium + python

Kaspresso

kaspersky

Где можно запускать автотесты?



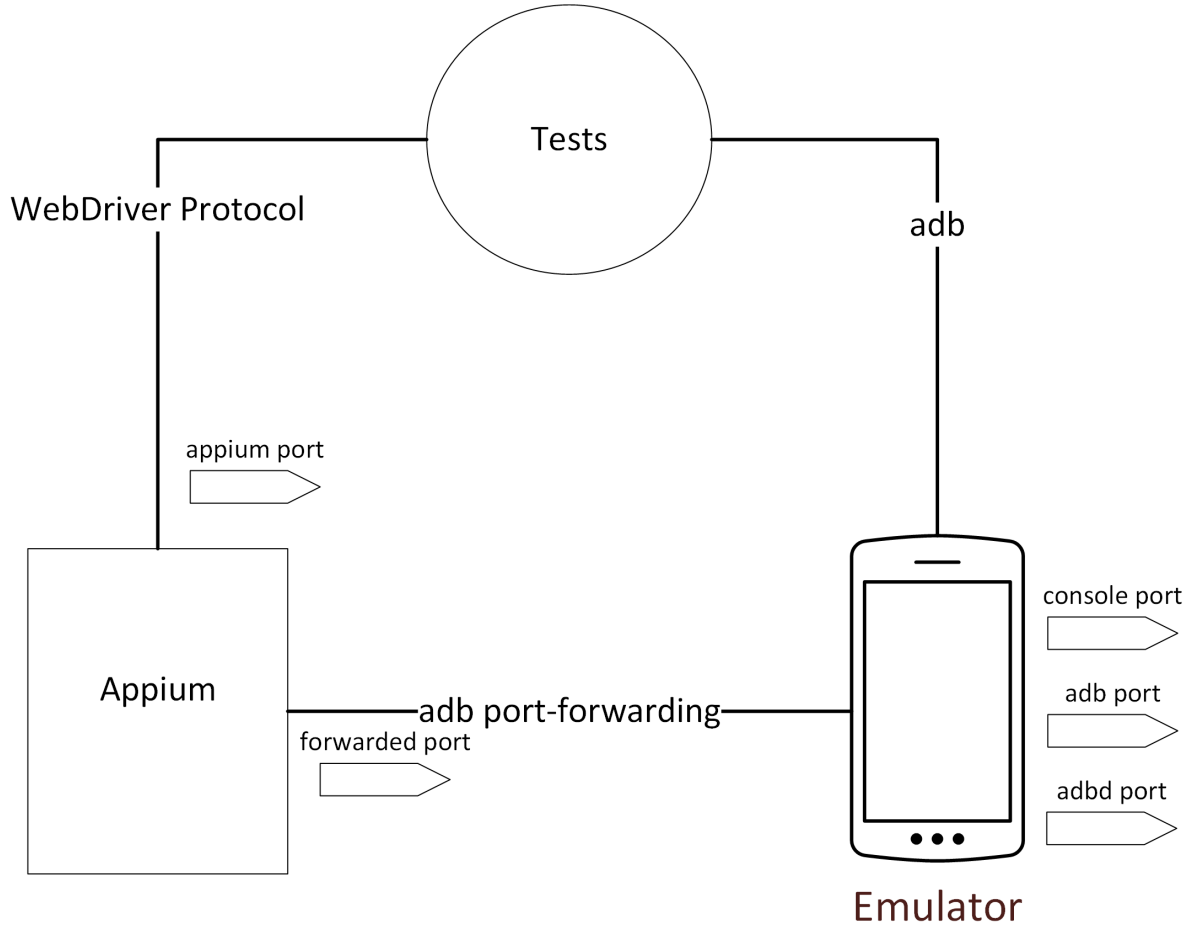


Проблемы запуска эмуляторов на голом железе

С чего всё начиналось?

Запуск тестов в несколько потоков без docker

- consoleport и adbport
- adbd
- Appium и uiautomator2



Предварительная настройка эмулятора

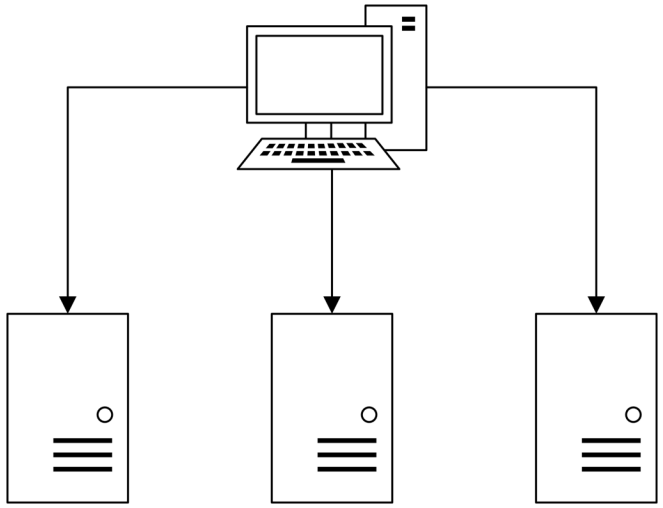
- Эмулятор должен иметь полностью сконфигурированный и готовый к работе Google Play
- Эмулятор не должен автообновлять приложения через Google Play
- Эмулятор должен иметь полностью сконфигурированный и готовый к работе Chrome
- Эмулятор не должен автоподставлять логины и пароли (Отключить Smart Lock в настройках аккаунта)
- Поставить хелпер, дать ему разрешения и включить фиктивные местоположения в настройках для разработчика

Предварительная настройка эмулятора

- Установлен последний доступный образ system-image. Для android версии больше 6 необходимо использовать образ с GPlay
- Все приложения должны быть обновлены
- Должен быть отключен Play Protect
- Эмулятор должен иметь полностью сконфигурированный и готовый к работе GMail (пройти визард + пройти диалог нового письма)
- Эмулятор должен иметь 24-часовой формат времени
- Отключить input assistance и spell correction в настройках клавиатуры (а лучше в принципе все помощники в клавиатуре)
- Возможно включить отображение касаний, для удобства
- Отключить анимации в настройках для разработчика

Проблемы распространения эмуляторов по разным машинам

- rsync
- smb share
- emulator + system-images



Запуск эмуляторов с другими тестами

- Пакеты в системе с Java/Ruby/Python и прочие системные либы
- Зависимости скриптовых языков
- Разные версии Appium
- Сервер с rest api для запуска и контроля эмулятора

Docker как спасение

Какие проблемы мы решили докером?

Контролируемое окружение

Хранение и дистрибьюция эмуляторов

Сетевая изоляция параллельный запуск тестов

Что получилось?

- Docker images для эмуляторов/тестов/Appium
- Docker-compose для запуска
- Хранение в docker-репозитории и автоматическое обновление при запуске

Основные слои образа с эмулятором

1. База

```
FROM ubuntu:18.04
ENV SHELL /bin/bash

RUN dpkg --add-architecture i386
RUN apt-get update && apt-get install -y --no-install-recommends \
    # Emulator & video bridge dependencies
    libc6 libdbus-1-3 libfontconfig1 libgcc1 \
    libpulse0 libtinfo5 libx11-6 libxcb1 libxdamage1 \
    libtcmalloc-minimal4 \
    libnss3 libxcomposite1 libxcursor1 libxi6 \
    libxext6 libxfixes3 zlib1g libgl1 pulseaudio socat \
    libglu1 libgl1-mesa-glx libgl1-mesa-dri lib32stdc++6 \
    # Enable turncfg through usage of curl
    curl net-tools ca-certificates && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

Основные слои образа с эмулятором

1. База
2. System-image

https://github.com/avito-tech/avito-android
https://github.com/google/android-emulator-container-scripts

```
COPY sys/ /android/sdk/system-images/android/
```

Основные слои образа с эмулятором

1. База
2. System-image
3. Emulator.exe

https://github.com/avito-tech/avito-android
https://github.com/google/android-emulator-container-scripts

```
COPY emu/ /android/sdk/
COPY platform-tools/adb /android/sdk/platform-tools/adb
```

Основные слои образа с эмулятором

1. База
2. System-image
3. Emulator.exe
4. Configs

avd/Pixel2.avd/config.ini:

```
PlayStore.enabled={{playstore}}
disk.dataPartition.size=4096
hw.cpu.ncore=2
hw.ramSize=2048

hw.lcd.density = 120
hw.lcd.width = 360
hw.lcd.height = 640

tag.id={{tag}}
abi.type={{abi}}
hw.cpu.arch={{cpu}}
image.sysdir.1=system-images/android/{{abi}}/
```

```
COPY avd/ /android-home
```

Основные слои образа с эмулятором

1. База
2. System-image
3. Emulator.exe
4. Configs

5. Настройки через adb

```
adb shell "settings put global window_animation_scale 0.0"
adb shell "settings put global transition_animation_scale 0.0"
adb shell "settings put global animator_duration_scale 0.0"
adb shell "settings put secure spell_checker_enabled 0"
adb shell "settings put secure show_ime_with_hard_keyboard 1"
```

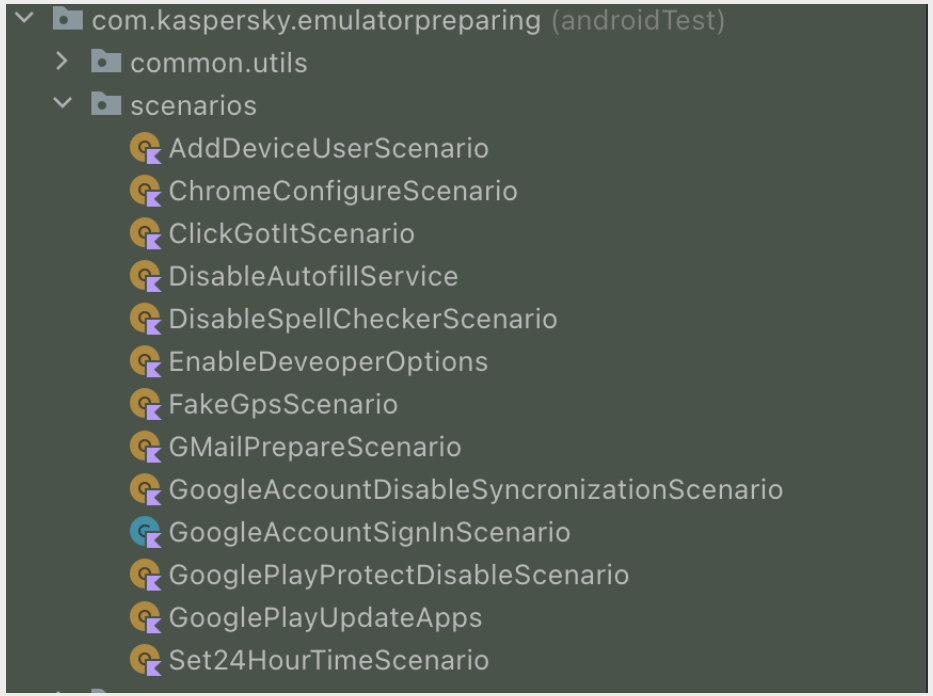
kaspersky

Основные слои образа с эмулятором

1. База
2. System-image
3. Emulator.exe
4. Configs

5. Настройки через adb

6. Настройка автотестом через UI



Основные слои образов с тестовым окружением

1. База

```
FROM ubuntu:18.04
# Install
RUN apt-get update && apt-get install -y --no-install-recommends \
   openjdk-8-jdk-headless \
   wget unzip curl \
   ca-certificates && \
   # Clean apt directories
   apt-get clean && \
   rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/* && \
   # Make java 8 default
   ln -s /usr/lib/jvm/java-8-openjdk-amd64 /usr/lib/jvm/default-java
```

Основные слои образов с тестовым окружением

1. База
2. Android SDK

```
ARG sdk_version=sdk-tools-linux-4333796.zip
ARG android_home=/opt/android/sdk
RUN mkdir -p ${android_home} && \
    wget -nc https://dl.google.com/android/repository/${sdk_version} \
    -O /tmp/${sdk_version} && \
    unzip -q /tmp/${sdk_version} -d ${android_home} && \
    rm /tmp/${sdk_version}

ENV ANDROID_HOME ${android_home}
ENV PATH=${ANDROID_HOME}/tools:${ANDROID_HOME}/tools/bin:${PATH}
RUN yes | sdkmanager --licenses && yes | sdkmanager --update

RUN sdkmanager \
    "tools" \
    "platform-tools" \
    "build-tools;28.0.3"

ENV PATH=${ANDROID_HOME}/emulator:${ANDROID_HOME}/build-tools/28.0.3:
:${ANDROID_HOME}/platform-tools:${PATH}
```

Основные слои образов с тестовым окружением

1. База
2. Android SDK
3. Java/Marathon/ Python+pytest/ Ruby+RubyGems

```
FROM android-minimal:latest

# Install android platfom 28
RUN sdkmanager "platforms;android-28"

# Install Gradle 6.6.1
RUN wget https://services.gradle.org/.../gradle-6.6.1-bin.zip -P /tmp
RUN unzip -d /opt/gradle /tmp/gradle-*.zip
ENV GRADLE_HOME=/opt/gradle/gradle-6.6.1
ENV PATH=${PATH}:${GRADLE_HOME}/bin
```

Проверка что эмулятор запущен

Основные проверки

- `adb devices -l | grep 'device'`
- `adb shell getprop sys.boot_completed`

Дополнительные

- `adb shell dumpsys window | grep mFocusedWindow | grep launcher`
- `adb shell uiautomator dump && cat /sdcard/window_dump.xml | grep some_button`

Пример docker-compose.yaml

```
emulator:
    image: emulator-base-v${SDK_VERSION}:latest
    container_name: ${prefix}_emulator
    networks:
      - "emu_net"
    devices: [/dev/kvm, /dev/dri]
    environment:
      - GPU_ENABLED=true
```

Пример docker-compose.yaml

```
appium:
    image: appium:1.18
    container_name:
      ${prefix}_appium
    depends_on:
      - "emulator"
    networks:
      - "emu_net"
```

Пример docker-compose.yaml

```
tests:
  image: java-tests-env:latest
  container_name: ${prefix}_tests
  command: [
      "./wait_for_emulator.sh",
      "java -jar core.jar $TEST_FILTER"
  ]
  networks:
    - "emu_net"
  depends_on:
    - "emulator"
    - "appium"
```

Docker – обязательно для автотестов Android

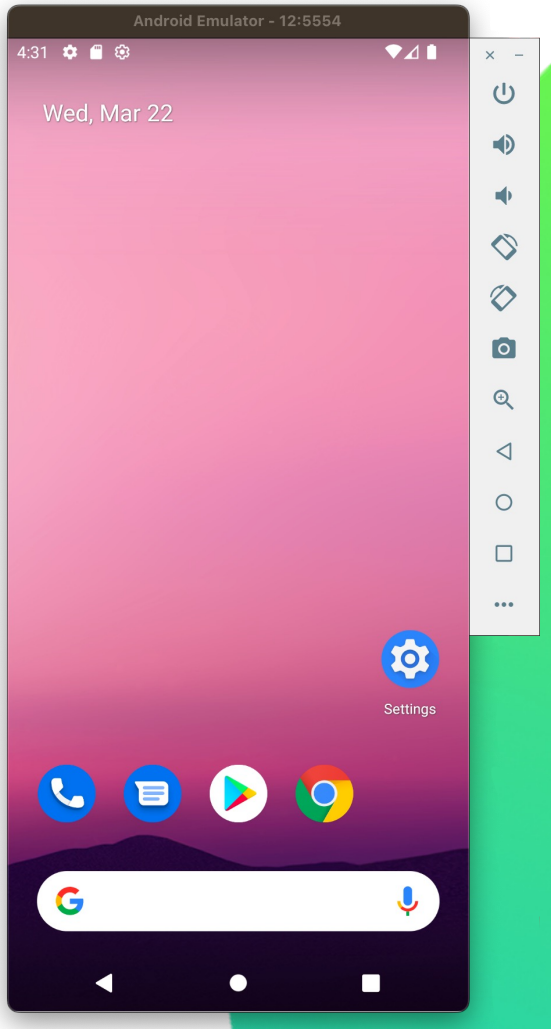
- Использование Docker избавило от головных и не только болей
- Тесты можно перенести на любую машину
- Описание сервисов и контейнеров в коде
- Можно запустить эмулятор для локальной отладки

Железо

Куда все движутся

Куда движемся мы

Что такое эмулятор?



Что же нагружает эмулятор?

CPU

полная эмуляция системы

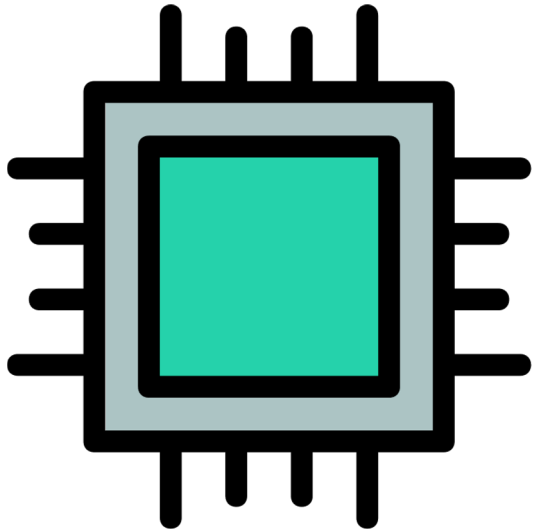
RAM

система и прикладное ПО

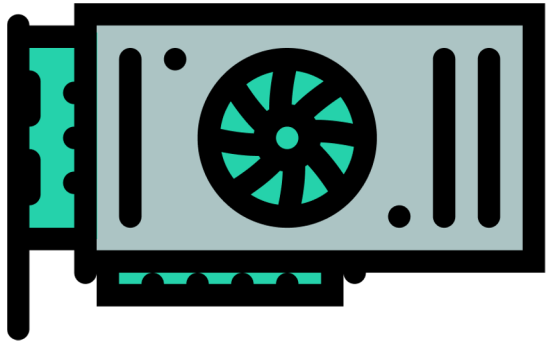
GPU

отрисовка всех компонентов системы и приложений

Опции по GPU



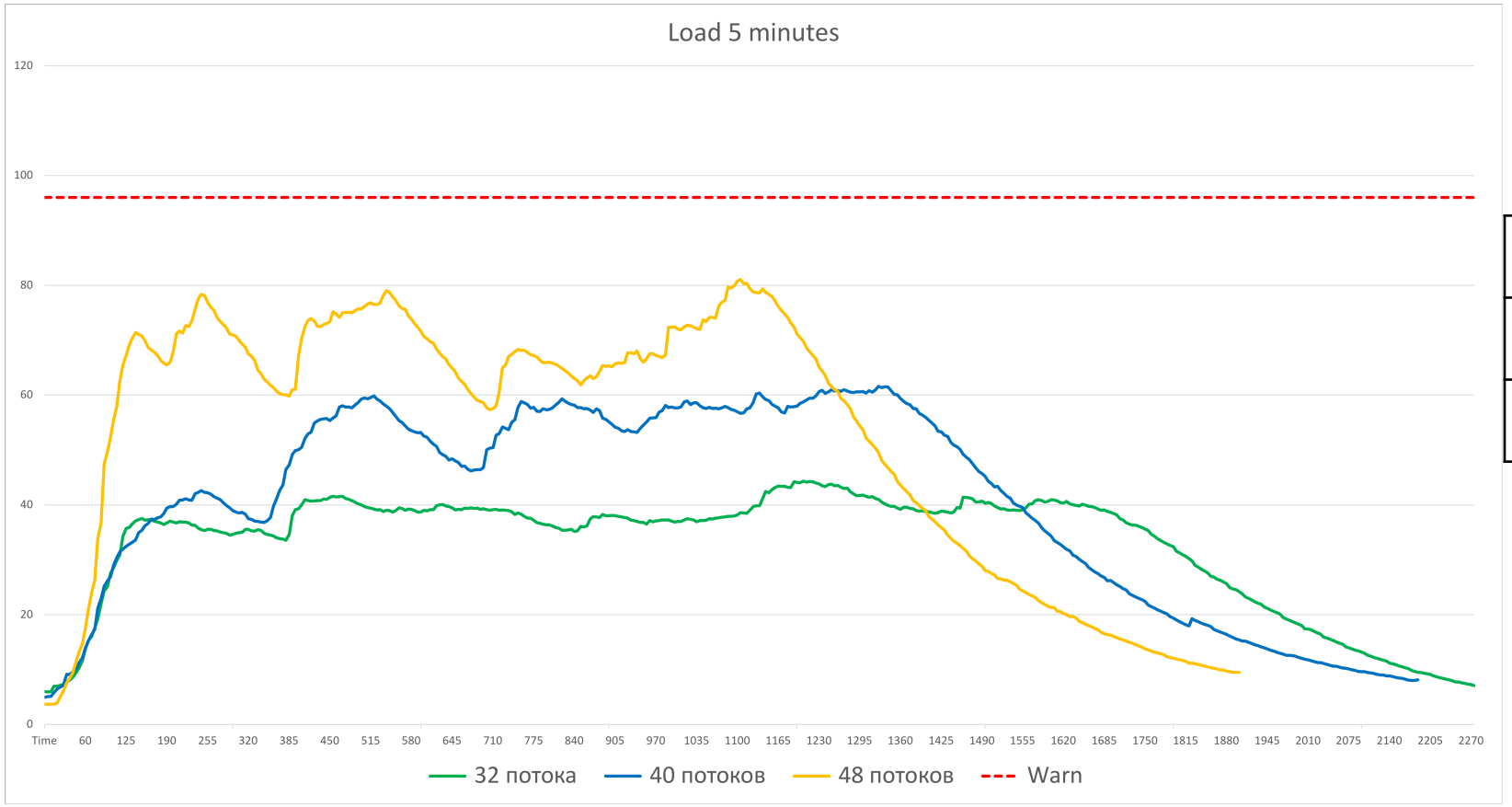
Отрисовка на CPU



Отрисовка на GPU

kaspersky

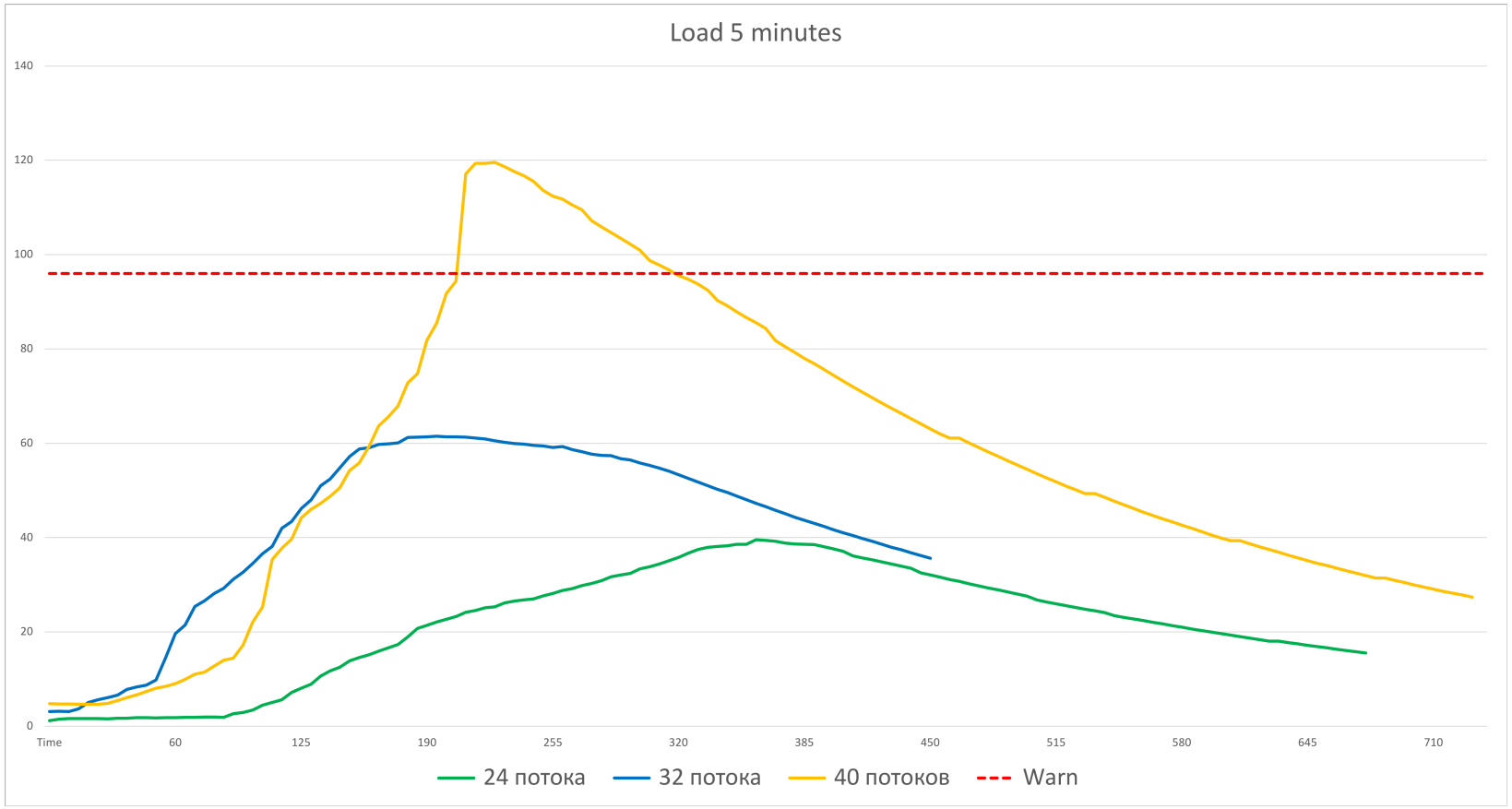
Общая нагрузка системы на сервере



32 потока	38 минут
40 потоков	36 минут
48 потоков	30 минут

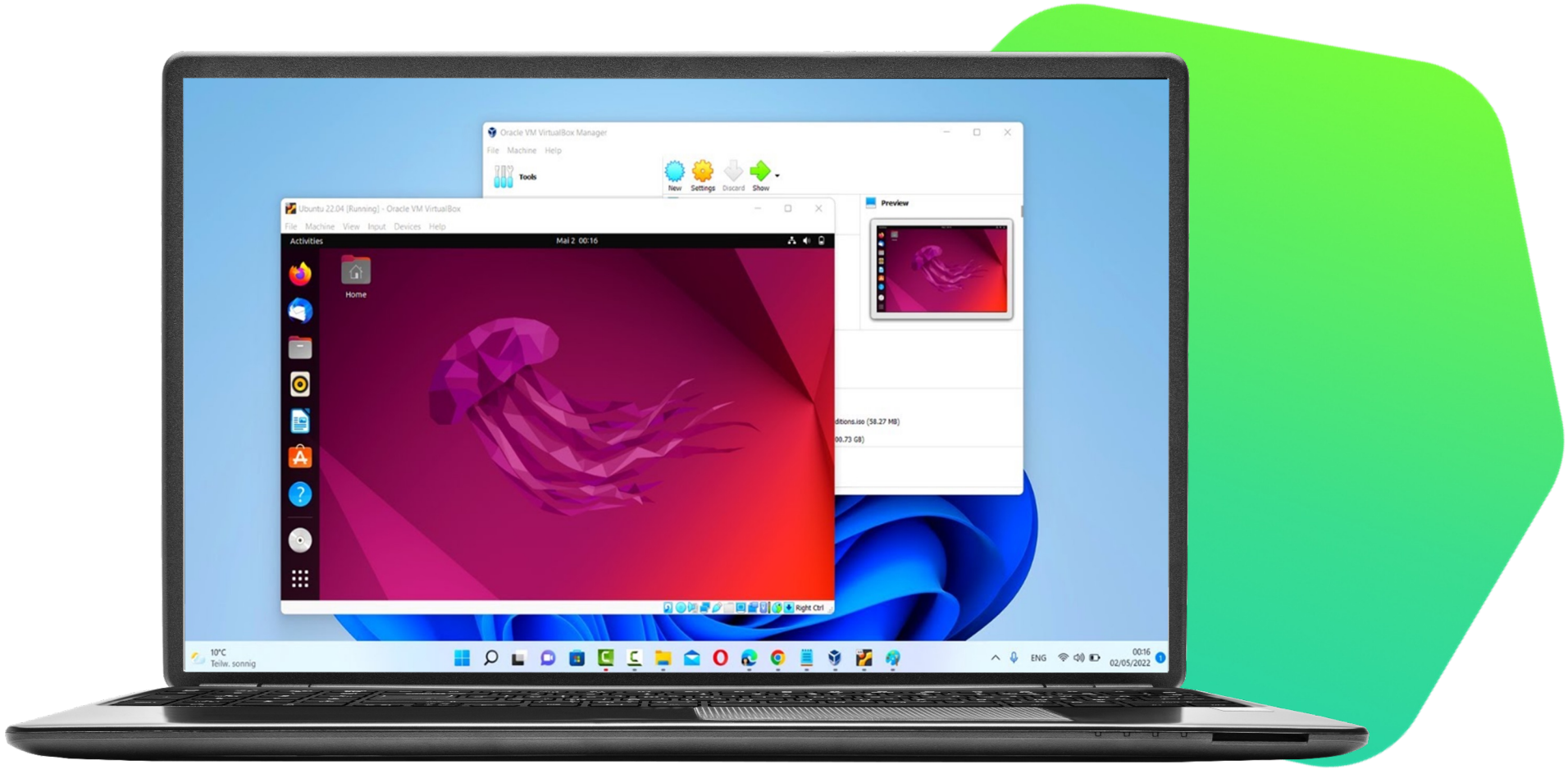
kaspersky

Общая нагрузка системы на сервере



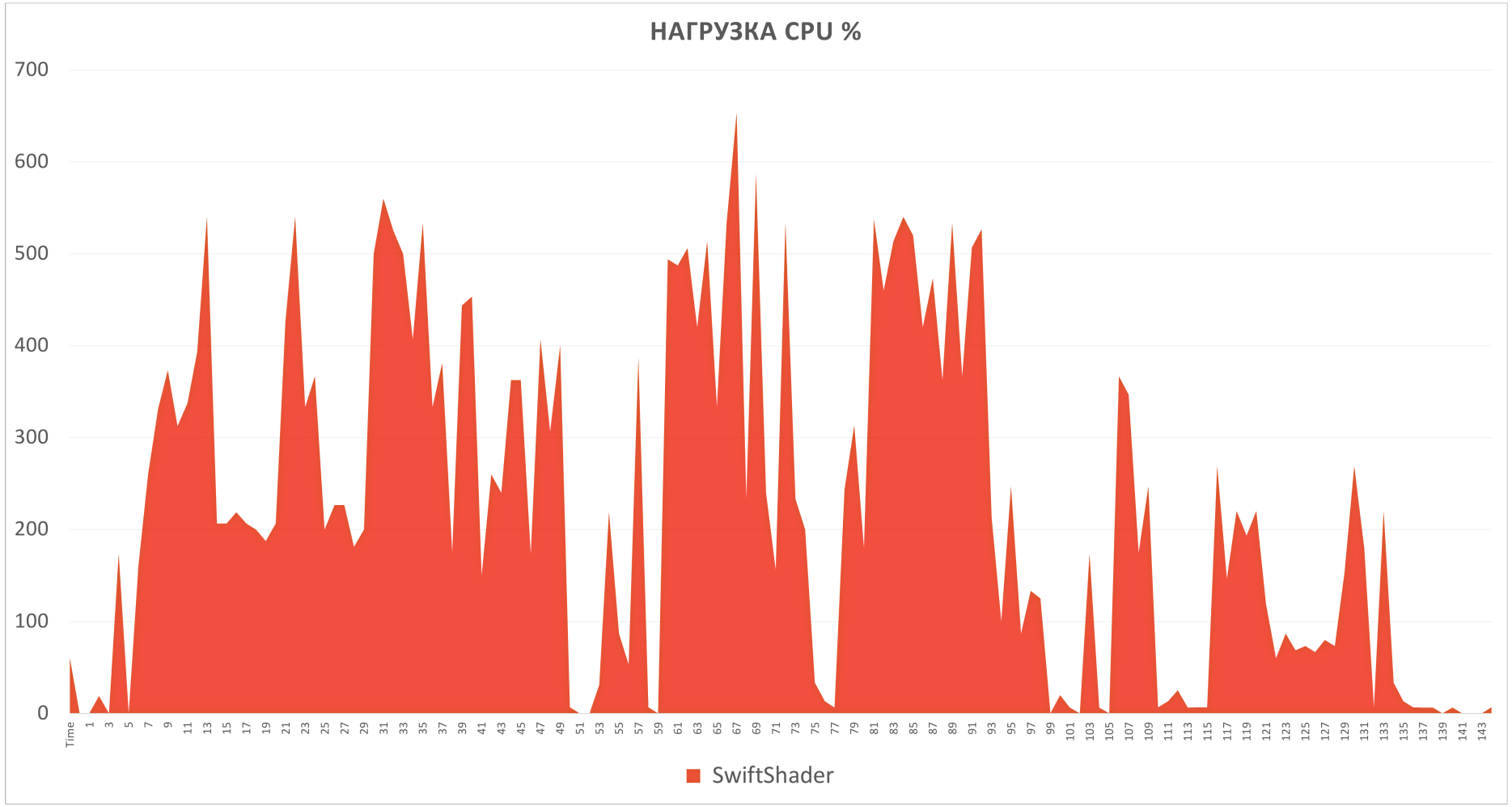
kaspersky

Дополнительная нагрузка от вложенной виртуализации



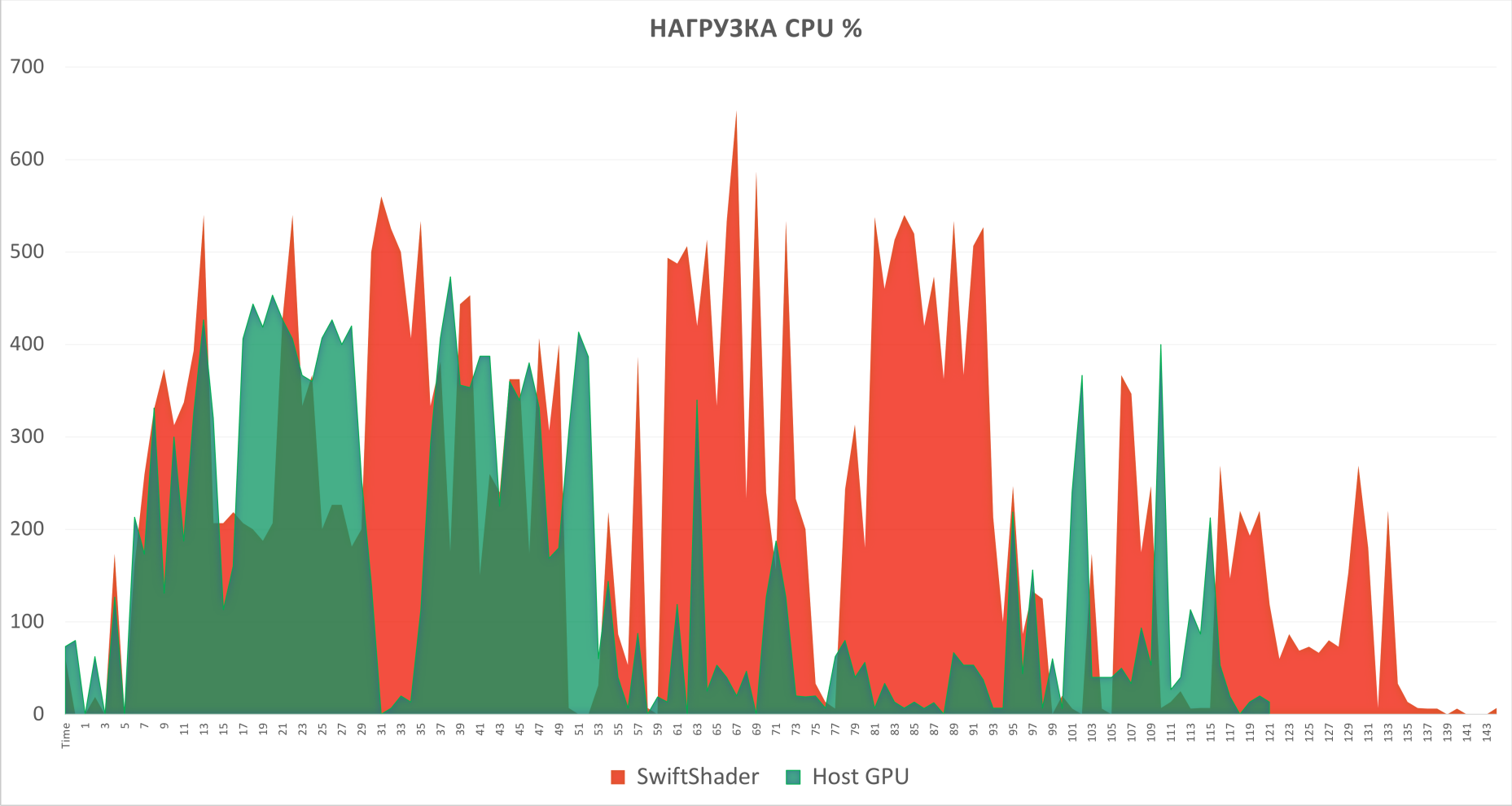
kaspersky

Разница в нагрузке ЦПУ



kaspersky

Разница в нагрузке ЦПУ

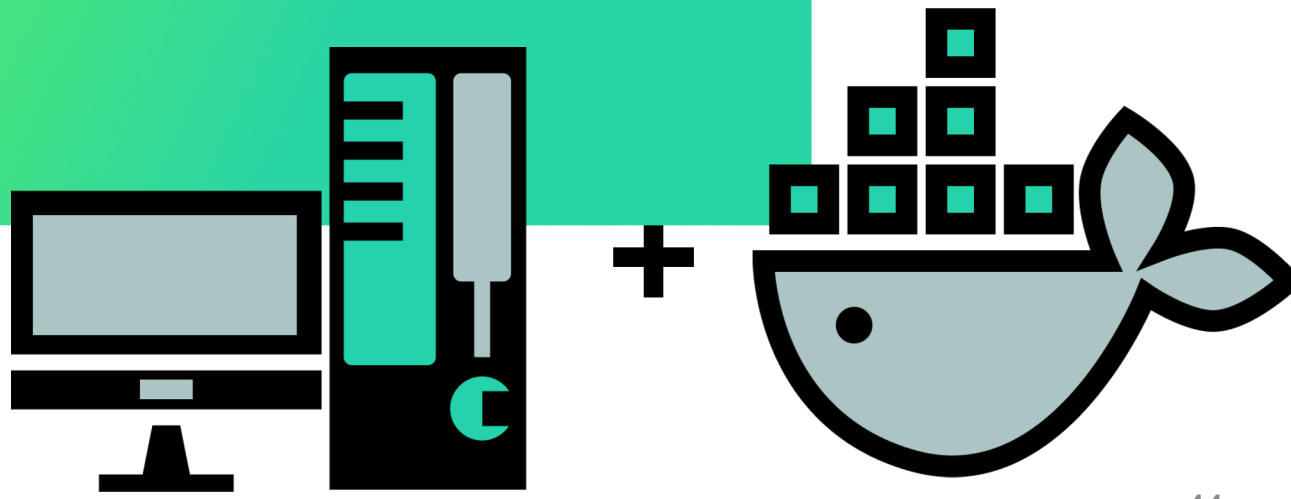


Почему не подходят сервера?

- Нужен GPU или дорогая по нагрузке на ЦПУ эмуляция
- Общая нагрузка на систему
- Вложенная виртуализация – медленно

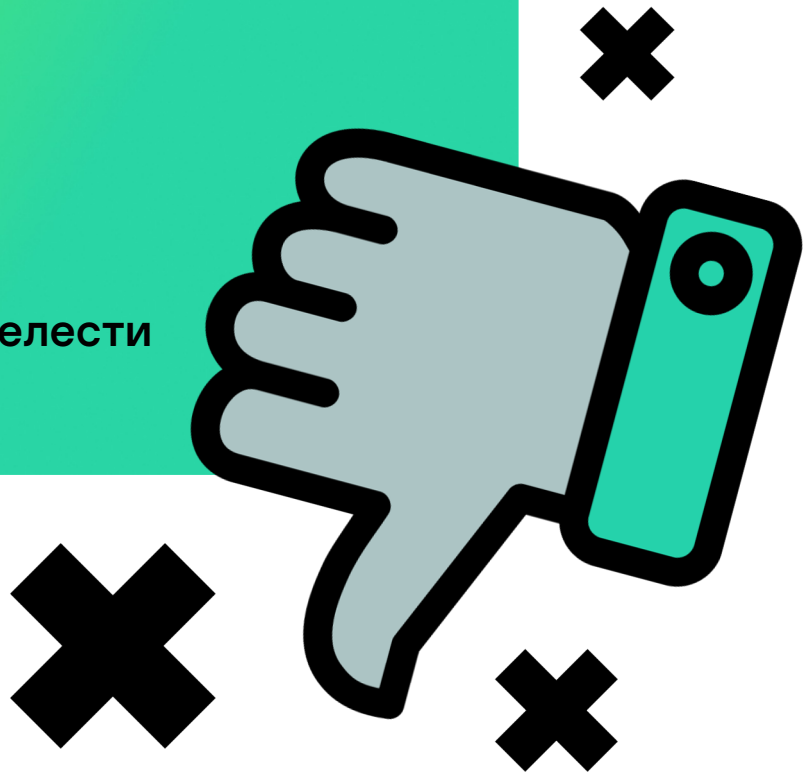
Решено – десктопы с GPU

- AMD Ryzen 7 2700X 8c/16t
- 32 GB
- RX570 8GB
- NVMe SSD 512 GB
- HDD 1TB
- 8 потоков тестов



Проблемы десктопов

- ssd – умирают
- Порты под cpu_fan – умирают
- Hdd – катастрофически не справляется
- Странные зависания, кернел паники и прочие прелести



Как решаем?

1

Создали зип,
на случай выхода из
строя компонентов

2

Оптимизируем
нагрузку
на дисковую
подсистему

3

Планируем закупки
нового
оборудования
по мере выхода из
строя старого

Почему такая конфигурация?

Большое количество потоков

Быстрый SSD

Nvidia не позволяет запустить больше 6-7 эмуляторов на 1 хосте

У Radeon свободные драйвера

Используем дополнительно новую конфигурацию

- AMD Ryzen 9 5900X 12c/24t
- 64 Gb RAM
- 2 TB NVMe SSD с большим TBW
- Radeon PRO WX 3200 4Gb
- 14 потоков тестов

kaspersky

Ресурсы на поток тестов

Название конфигурации	Количество потоков	Количество ядер на поток	Размер ОЗУ на поток	Примерная стоимость потока тестов
Сервер **2 x Intel Xeon Gold 6240R** **CPU = 48c/96t** **188GB RAM**	32-40	2.4 - 3	4	700-600 $ (2019 г.)
Десктоп игровой без использования GPU **AMD Ryzen 7 2700X** **8c/16t** **32 GB**	4	3.5-4	4	170 $ (2018 г.)
Десктоп игровой **AMD Ryzen 7 2700X** **8c/16t** **32 GB** **RX570 8GB**	8	2	4	100 $ (2018 г.)
Десктоп игровой NG **AMD Ryzen 9 5900X** **12c/24t** **64 Gb RAM** **WX 3200**	14	1.7	4.5	150 $ (2019 г.)

Выводы

- Запускать тесты на голом железе – не разумно
- Docker необходим для эмуляторов и окружения
- Десктопы дешевле для организации фермы эмуляторов

Спасибо
за внимание!

Вопросы?

kaspersky BRING ON THE FUTURE