Опыт изучения передовых китайских СПО-решений для построения сложных IT-инфраструктур на базе единой программной платформы OpenScaler/openEuler

Павлов Владимир / Варенов Дмитрий

Сообщество OpenScaler

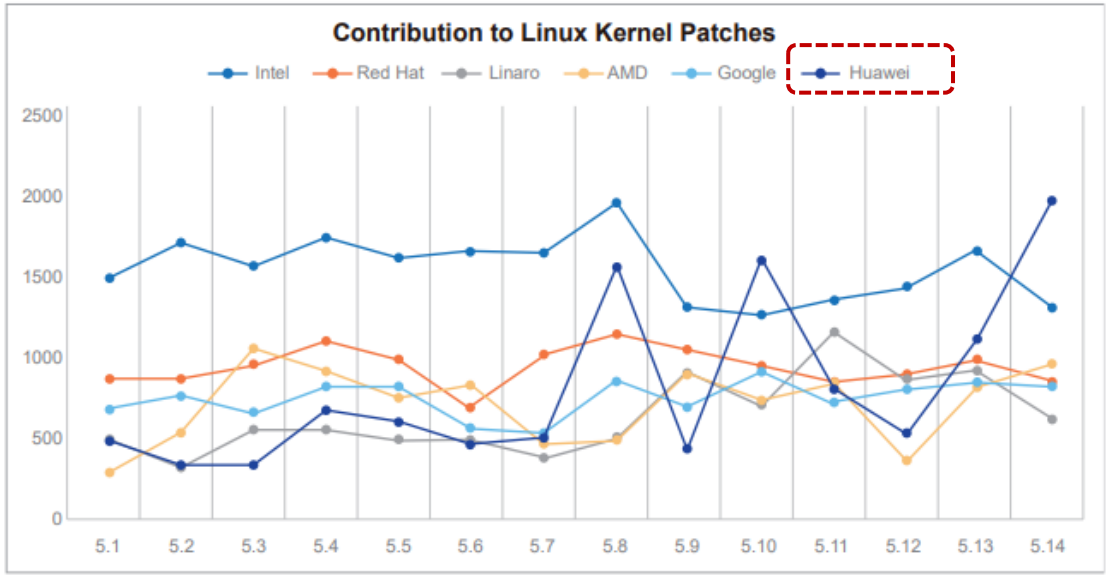
HL 2023 HighLoad++

История развития openEuler

Корпоративная ОС для внутреннего использования

2010	2012	2016	2017	2018	2019
Старт проекта	Внутреннее использование	Коммерческий дистрибутив		Развитие	

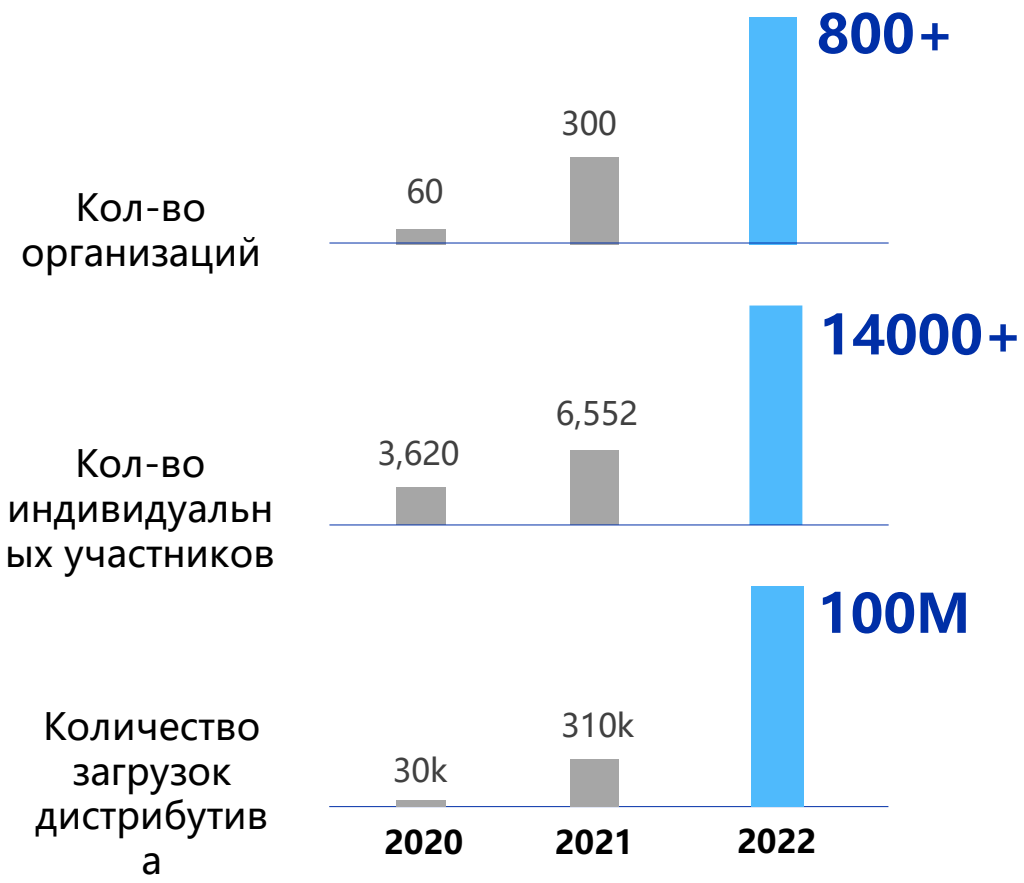
OpenEuler



Переход в Open Source в июле 2019

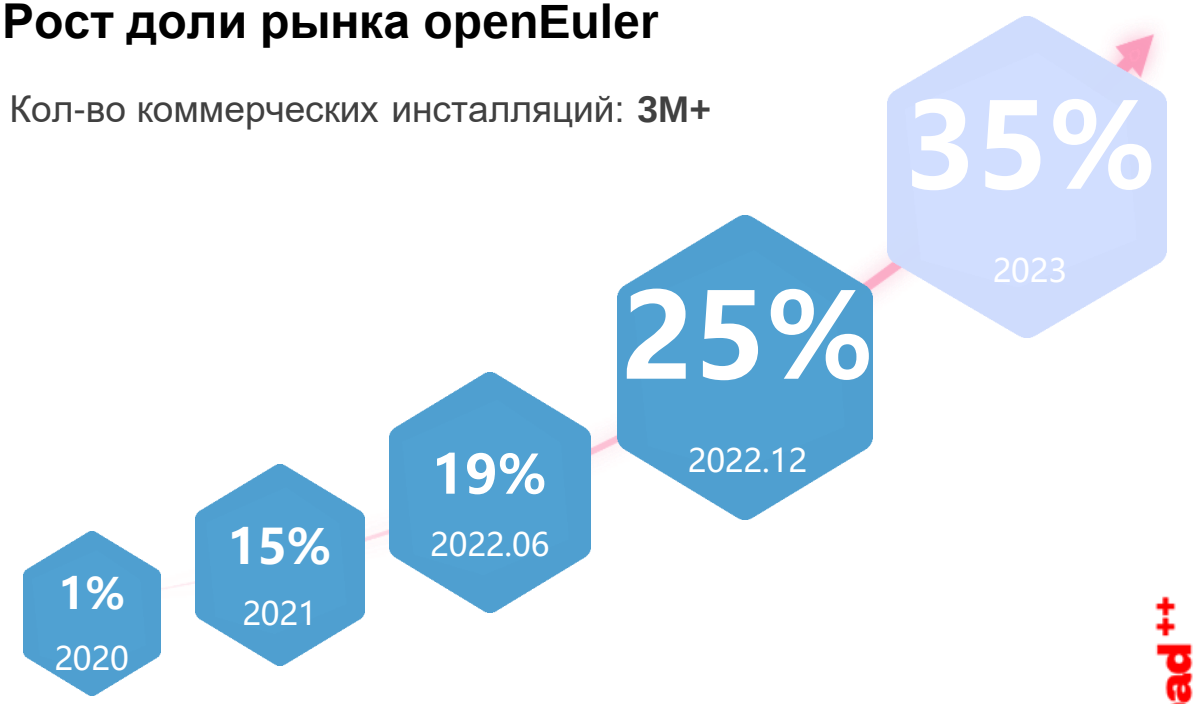
В 2023 openEuler – операционная система №1 в Китае и крупнейшее Open Source сообщество

Рост сообщества openEuler



Рост доли рынка openEuler

Кол-во коммерческих инсталляций: **3M+**



50%+ гос-во

40%+ телеком

30%+ финансы

25%+ энергетика

HighLoad++

20+ коммерческих дистрибутивов

百度智能云	中国电信 CHINA TELECOM	中国移动 China Mobile	China unicom 中国联通
Server/Cloud	Server/Cloud	Server/Cloud	Server/Cloud

KYLINSOFT 麒麟软件	统信软件技术有限公司	麒麟信安 KYLINSEC	SUSE	xFUSION	CETC	ISCAS	HOPERUN 润和软件
Server/Cloud	Server/Cloud	Server/Cloud/Embedded	Server/Cloud	Server/Cloud	Server/Cloud	Server/Cloud	Server/Cloud

H3C	turbolinux Powerful Thinking.	ThunderSoft 中科创达	翰思	红方 Linux	中科方德	KYLAND 科东（广州）软件科技有限公司	浪潮
Server/Cloud	Server/Cloud	Edge	Server/Cloud/Embedded	Server/Cloud	Server/Cloud	Embedded	Server/Cloud

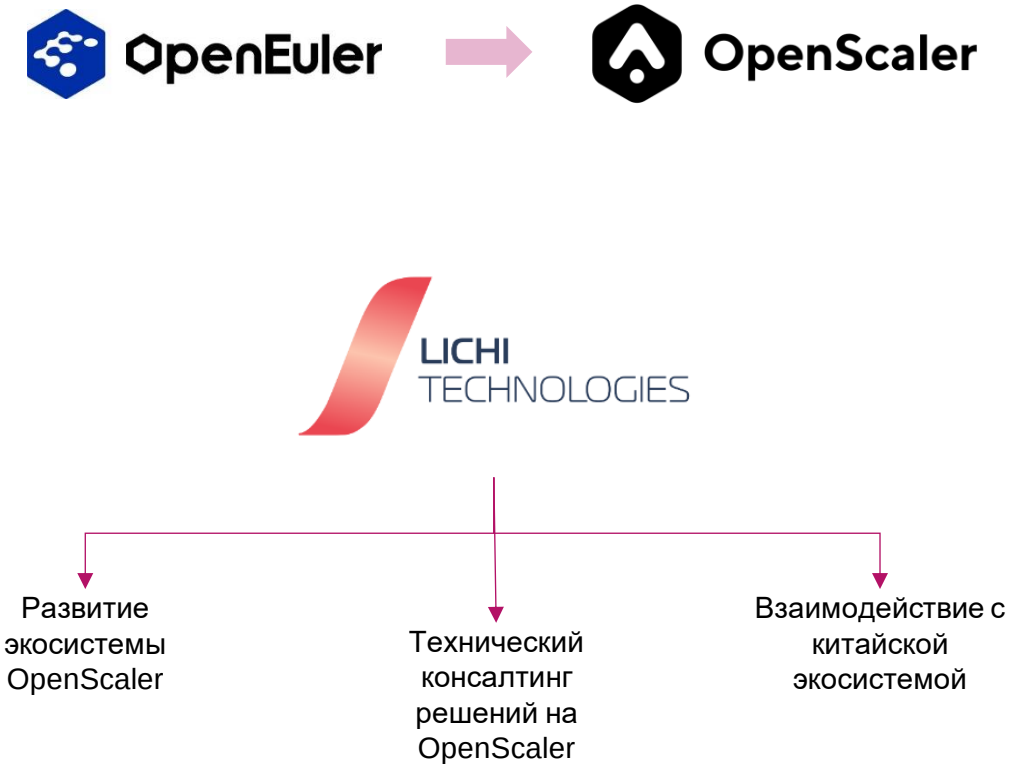
2022 - 2023: Российское сообщество OpenScaler

2022

- Создано локальное сообщество OpenScaler
- Официально выпущены 6 комьюнити-версий ОС для x86 и для ARM
- Начало разработки комм. версии ОС на базе OpenScaler
- Достигнуты договоренности о запуске пилотных проектов с рядом крупнейших клиентов

2023

- Весной 2023 года завершен образовательный курс с общим охватом 350+ студентов на базе НГУ и НГТУ
- На базе дистрибутива OpenScaler прошел всероссийский конкурс среди разработчиков системного ПО OpenOS Challenge, который привлек 1200+ участников
- Достигнуты соглашения о совместной работе с крупнейшими вендорами ОС и СИ Китая
- Запланировано 50+ активностей по развитию экосистемы



OpenScaler – в учебном процессе ВУЗов

- С весеннего семестра OpenScaler применяется в учебном процессе в:
  - Институте вычислительной математики и математической геофизики (ИВМиМГ СО РАН)
  - Новосибирском государственном университете
  - Новосибирском государственном техническом университете

- OpenScaler демонстрируется студентам в рамках курсов по операционым системам, разъясняются особенности решения.

- OpenScaler предлагается к установке для выполнения заданий по разработке программного обеспечения в рамках лабораторных работ, семестровых проектов в университетах в ходе курсов по организации высокопроизводительных вычислений, по операционным системам, а в ИВМиМГ СО РАН в рамках практики студентов.

ИВМиМГ

Новосибирский государственный технический университет
НЭТИ

Новосибирский государственный университет
*НАСТОЯЩАЯ НАУКА

OpenScaler – в Yandex Cloud

Yandex Cloud

Поиск

Сервисы Решения Почему Yandex Cloud Ресурсы Тарифы Документация Блог

← Marketplace

ОпенСкейлер

Обновлено 23 ноября 2023 г.

Дистрибутив OpenScaler включает следующие передовые программные наработки:

- A-Tune — систему для автоматической оптимизации настроек с помощью механизма машинного обучения. С помощью технологий искусственного интеллекта подбираются оптимальные параметры конфигурации операционной системы для повышения общей эффективности работы системы в соответствии с рабочей нагрузкой;
- собственный стек легковесной виртуализации (StratoVirt) и контейнеризации (iSulad);
- инструмент secPaver для настройки политик информационной безопасности SELinux;
- Kernel Live update — обновление ядра ОС без перезапуска виртуальной машины или физического сервера
- NFS MutiPath — Предлагается установить несколько каналов связи между клиентом и сервером в рамках одной точки монтирования для поддержки передачи данных ввода-вывода по нескольким каналам, повышая производительность одной точки монтирования. Кроме того, периодически проверяется состояние соединения, чтобы обеспечить быструю отработку отказа ввода-вывода при сбое соединения.
- SysCare — технология SysCare позволяет произвести установку патча для процесса пользователя без его перезапуска.

Особенности дистрибутива

... и некоторые инновации

Кроссплатформенность: любые приложения на любой платформе

Информационные технологии + Коммуникационные технологии + Промышленные технологии

Основные приложения: облачные технологии, большие данные и CDN, MEC, промышленный контроль...

100% охват основных сценариев приложений

Поддержка любых приложений

OpenScaler

Поддержка различных устройств

100% охват основных вычислительных архитектур

ARM, x86, RISC-V, SW-64, LoongArch

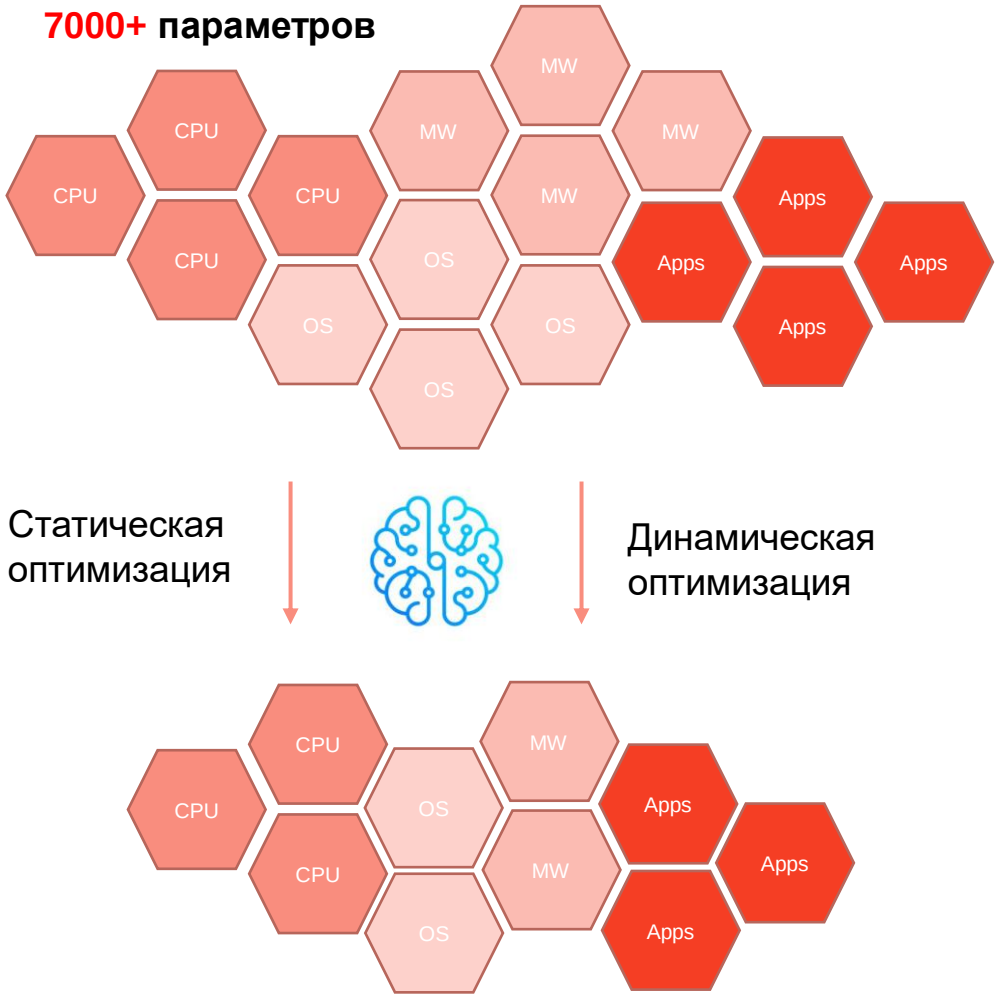
NPU, GPU, DPU, 100 + устройств, 300 + плат




Сервера Облака Краевые Встроенные

Единая платформа распространения ОС

A-Tune: оптимизация производительности с помощью ИИ



Определение оптимальных значений и комбинации параметров на разных уровнях стека (оборудование, ОС, промежуточный уровень, приложение)

- **Статическая оптимизация**: анализ логов и подбор профиля настройки из предустановленных
- **Динамическая оптимизация:** выбор значимых параметров, подбор оптимальной конфигурации

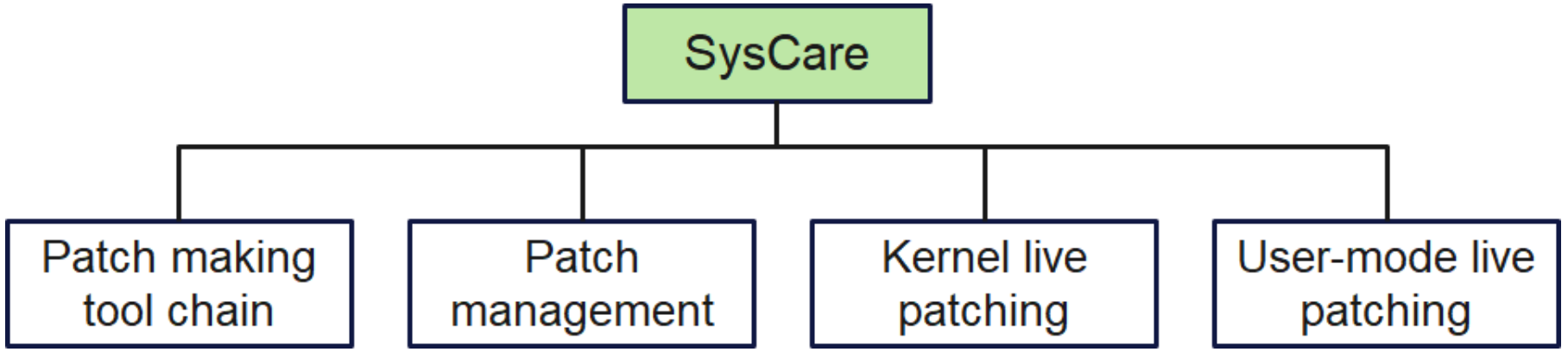
20+%	10+%	10+%
Postgres	Terasort, HDFS	HBase, HIVE, Spark

HighLoad++

Kernel Live-Update: Смена ядра ОС без перезагрузки физического сервера или виртуальной машины

- Смена ядра ОС без физической перезагрузки сервера
- Возможность «заморозить» состояние отдельных процессов (при указание юнита system, PID)
- Поддержка архитектур ARM / X86

SysCare – Применение устранений уязвимостей



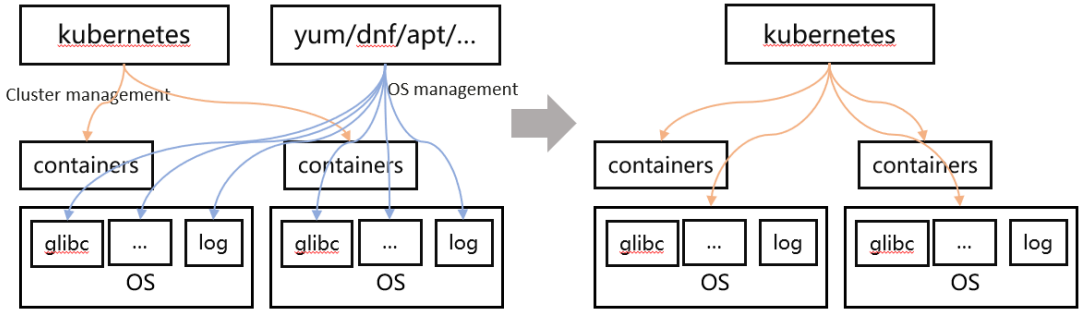
Возможности утилиты:

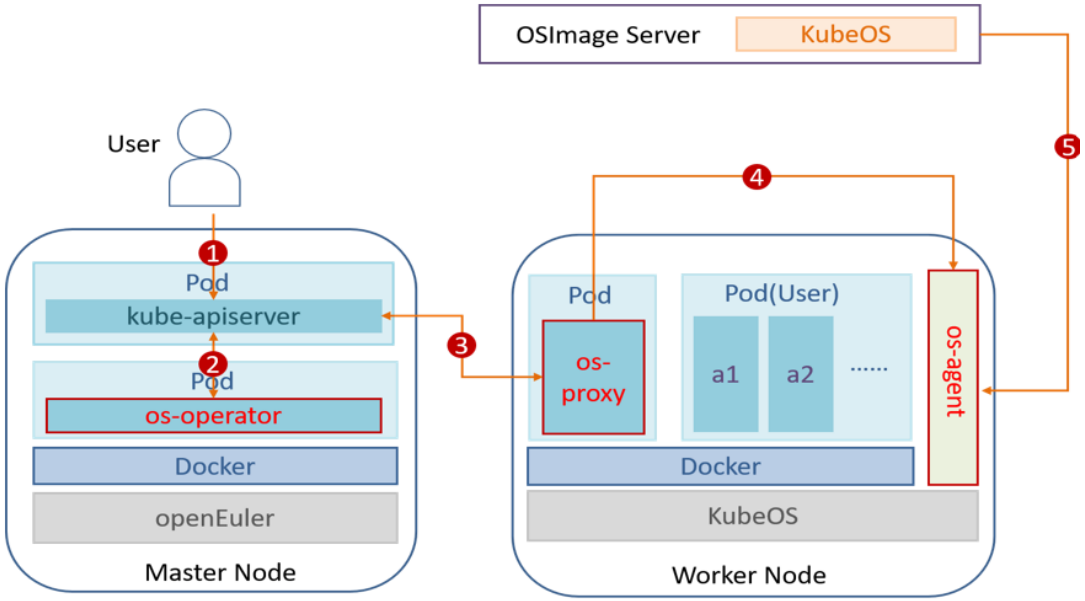
1. Простое создание патчей, как для ядра, так и для уровня пользователя.
2. Интерфейс для управления патчами, включая их установку, активацию, деактивация и удаление.

Используются следующие возможности:

1. Унифицированные патчи: SysCare скрывает разницу в деталях создания патча, предоставляя унифицированный инструмент для улучшения эффективности эксплуатации и обслуживания.
2. Применение патчей на лету для приложения уровня пользователя: SysCare поддерживает наложение патчей на многопоточные и много-процессные приложения в пространстве пользователя, что позволяет достичь эффекта без перезапуска процесса или потока.
3. Ленивый механизм: SysCare отлавливает все системные вызовы через ptrace и ждёт их завершения, что позволяет повысить эффективность успешного применения.

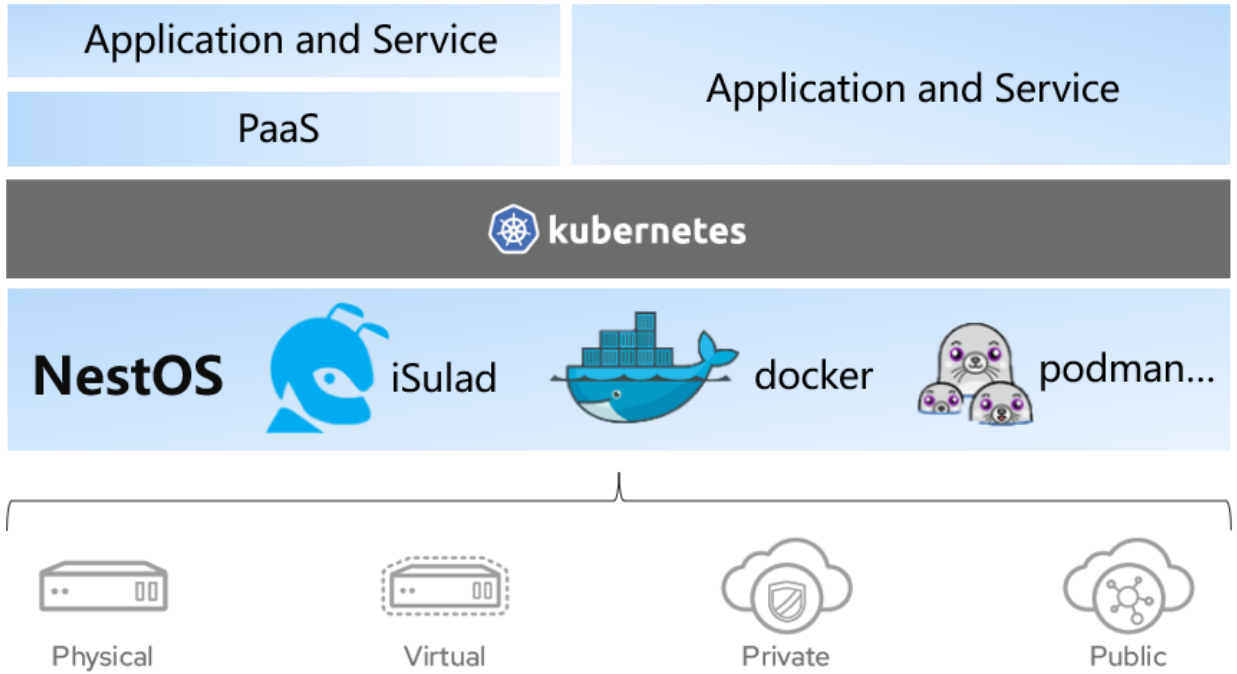
KubeOS: упрощение управления обновлениями ОС





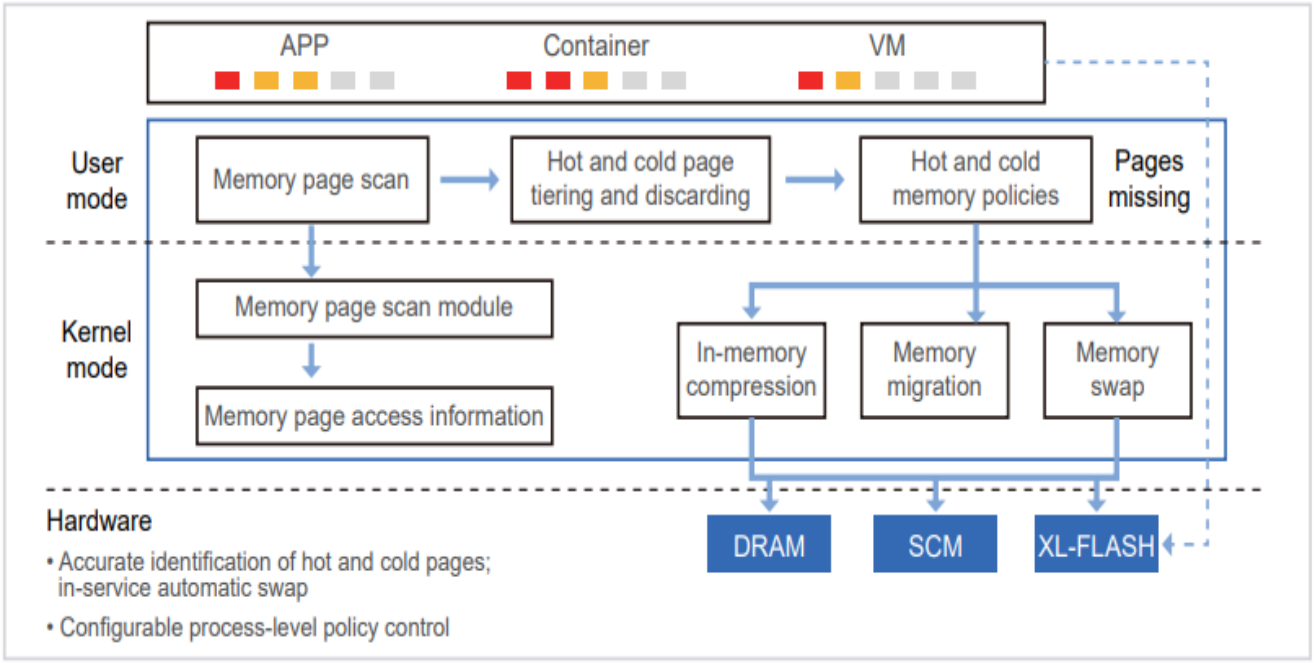
- Используйте API Kubernetes для управления ОС узлов кластера.
- Подключение ОС к Kubernetes через CRD + Operator
- Управление ОС через утилиты управления Kubernetes
- Упрощение обновления и отката ОС

NestOS: Минималистичная ОС для узлов контейнерной виртуализации



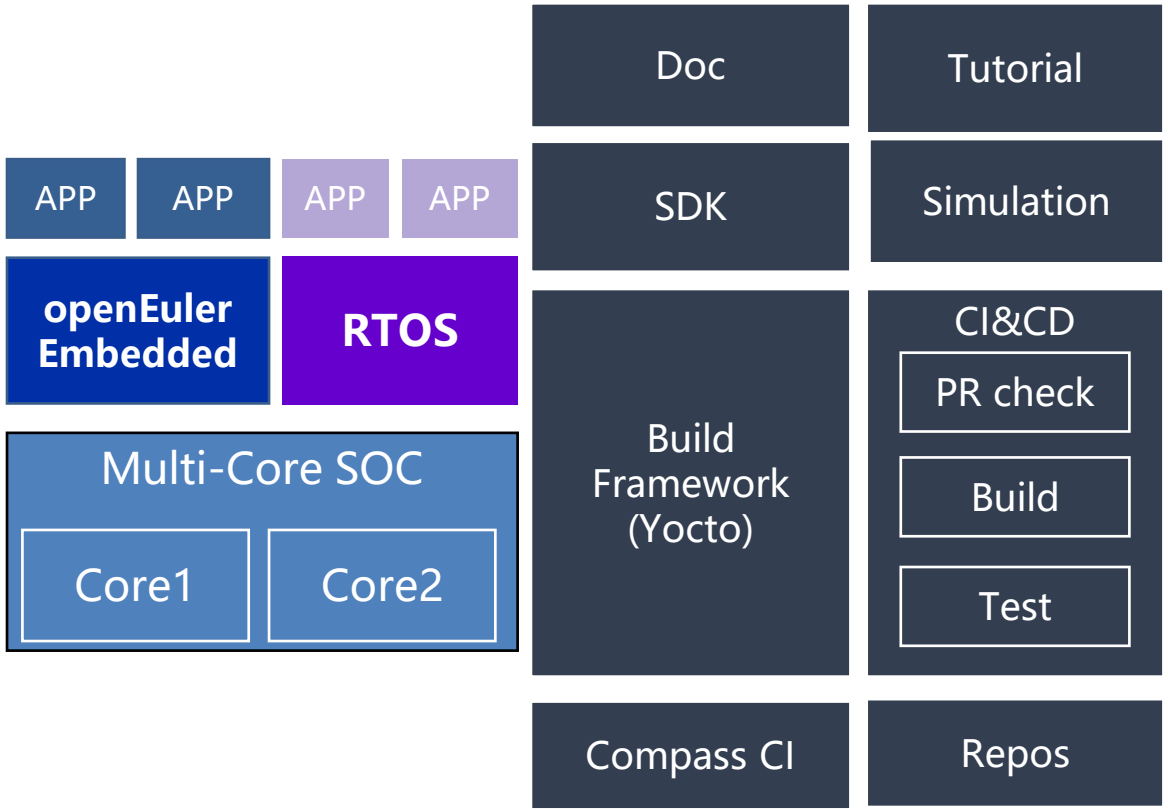
- Содержит минимально необходимый набор компонентов
- Базовая ОС для запуска кластера k8s
- Для записи доступны только разделы /etc, /var. Остальные разделы доступны только на чтение.
- Удобная система обновления (используются образы системы, обновляемые при помощи rpm-ostree).
- Наличие инструментов для создания своих сборок NestOS, содержащих сразу необходимый набор софта или конфигурационных файлов
- Поддержка "из коробки" среды выполнения контейнеров iSulad

etMem: новый взгляд на раздел подкачки



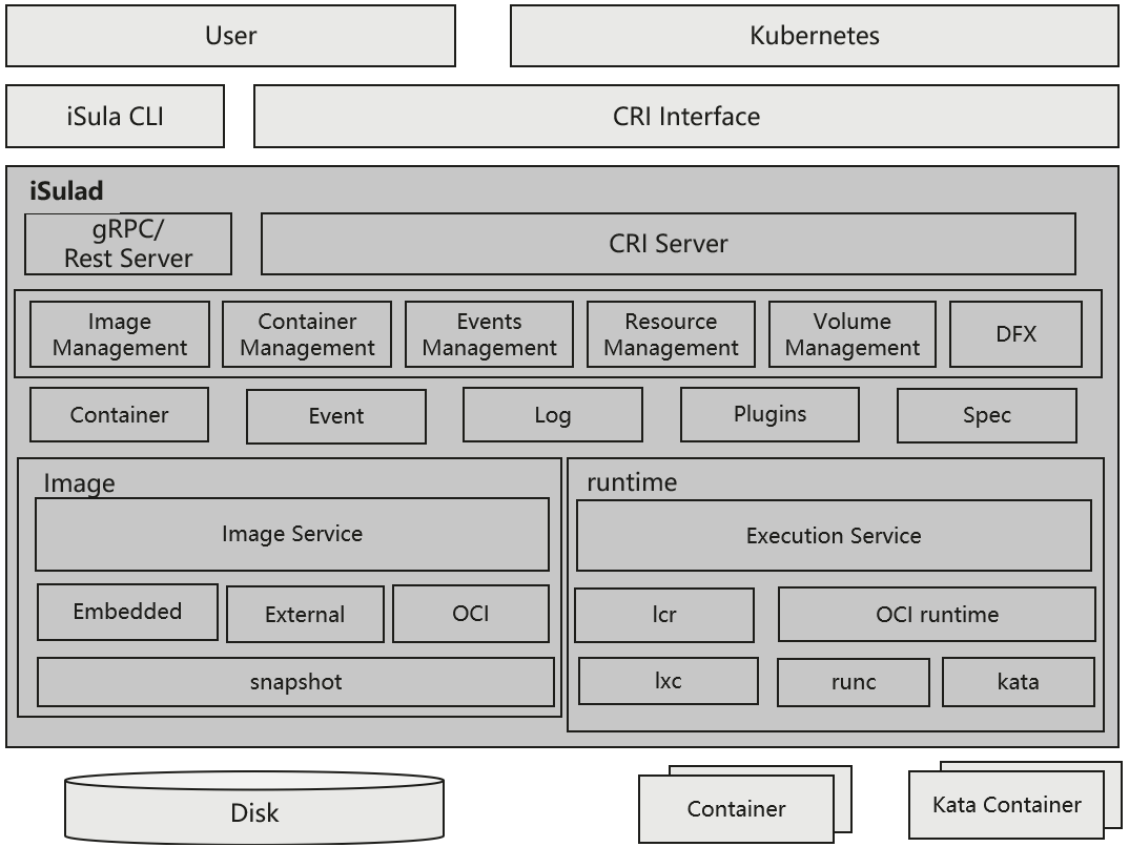
- DRAM составляет более 40% стоимости серверов центров обработки данных
- Разрыв в производительности ЦП и DRAM увеличивается: за 10 лет количество ядер ЦП увеличилось в 7 раз, а объем памяти увеличился только в 4 раза
- etMem хорошо подходит для приложений, использующих большой объем ОЗУ, но при ограниченных обращениях к ней к примеру, таких как MySQL, Redis, или Nginx
- Особенности:
  - Механизм контроля: за счет наличия возможности настройки по сравнению со стандартным kswap
  - Механизм сканирования: сканирует процессы в пользовательском и привилигированном режиме.
  - Вытеснение “горячих”/”холодных” данных.
  - Политика вытеснения: гибкие политики вытеснения данных.

ОС для Embedded-устройств



- Возможность создания легковесных дистрибутивов для встраиваемых устройств
  - Поддержка фреймворка Yocto, создание дистрибутива ОС для любого типа устройств размером от 5 MB
  - Наличие механизмов оптимизации дистрибутива и времени запуска
- Поддержка широкого спектра оборудования, включая SoC-решения на ARM, такие как Rasberry Pi и аналоги
- Наличие возможности использования ядра «реального времени» и RTOS (Real time Operating system)
- Наличие Distributed soft bus (DSoftBus) для взаимодействия встраиваемых и мобильных устройств между собой
- Наличие 80+ апробированных и готовых к использованию пакетов в специализированных дистрибутивах

iSula: легковесные контейнеры



iSula разработан на C/C++ и использует меньше вычислительных ресурсов для оптимизации производительности

- Обеспечивает совместимость со стандартами kubernetes CRI и сетевыми плагинами CNI
- Поддерживает изоляцию namespace и Cgroup v1/v2
- Поддерживает application container, system container и secure container
- Поддержка Seccomp, технологии SELinux Security-Enhanced
- Соответствует спецификации OCI (на базе среды выполнения lxc)
- Поддержка встроенных и внешних форматов образов, а также спецификаций образов OCI и образов докеров.

Тестирование системы контейнерной виртуализации iSula

Технические условия тестирования

Для проведения тестов использовалась последняя на текущий момент LTS-версия дистрибутива OpenScaler - 22.03 LTS SP-2.

Тестирование решений контейнерной виртуализации проводилось на физических серверах следующей конфигурации:

X86_64
- Сервер Huawei 1288H V5
- CPU: Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz – 2шт., 20 ядер
- RAM: 128 Гб
- SSD: INTEL SSDPE2KE016T8 - 2 шт. (NVMe)
- OS: openScaler 22.03 LTS SP1
- kernel: 5.10.0-153.12.0.92.os2203sp2.x86_64

ARM64
- Сервер Huawei TaiShan 200 (Model 2280)
- CPU: Kunpeng 920-4826 – 2шт., 96 ядер
- RAM: 512 Гб
- SSD: HUAWEI HWE52P433T2M005N – 4 шт. (NVMe)
- OS: openScaler 22.03 LTS SP1
- kernel: 5.10.0-153.12.0.92.os2203sp2.aarch64

OpenScaler

В качестве объектов тестирования были отобраны следующие системы контейнеризации:
- Docker (версия 18.09.0) из репозитория OpenScaler 22.03 SP2
- Docker (версия 24) с сайта docker.com
- iSulad (версия 2.1.2) из репозитория OpenScaler 22.03 SP2
- podman (3.4.4) из репозитория OpenScaler 22.03 SP2

Тестирование системы контейнерной виртуализации iSula

Используемый инструментарий тестирования

- Для проведения тестирования будет использоваться утилита тестирования решений контейнерной виртуализации – ptcr (https://gitee.com/openeuler/ptcr)

- ptcr - это инструмент тестирования скорости выполнения операций над контейнерами как в последовательном, так и в параллельном режиме исполнения команд.

- Утилита предназначена для тестирования скорости выполнения типовых команд над контейнером, в том числе включающих:
    - create
    - start
    - stop
    - rm
    - run

При тестировании будут проводится измерения времени выполнения этих команд в двух режимах:

- последовательный запуск (по 10 раз) команд create, start, stop, rm, run
- параллельный запуск (по 100 одновременно) команд create, start, stop, rm, run

Для теста, в качестве контейнера выбран образ busybox:1.36.0, в котором будет запускаться команда sleep в цикле.

Сам процесс тестирования запускается представленной ниже командой: **ptcr -c конфиг_теста.yml > результат_теста.log**

Для каждой системы контейнеризации запускаем тесты двух видов:

- операции выполняются последовательно
- операции выполняются параллельно

После проведения тестирования каждой системы сервер в обязательном порядке перезагружается

Тестирование системы контейнерной виртуализации iSula

x86	average spent			
Последовательно (по 10 операций). Принимаем isula за 100%				
	docker 18 (x86)	docker24 (x86)	podman (x86)	iSulad (x86)
Create	33	16	129	27
Start	293	282	237	66
Stop	27	11	132	19
Remove	29	12	129	20
Run	255	184	230	81
Параллельно (100 операций). Принимаем isula за 100%				
	docker 18 (x86)	docker24 (x86)	podman (x86)	iSulad (x86)
Create	298	78	726	159
Start	9299	9226	5434	561
Stop	173	86	1376	133
Remove	145	77	928	84
Run	5091	5104	4628	739

	vs Docker 18 (%)	vs Docker 24 (%)	vs Podman (%)
Create	22,22	-40,74	377,78
Start	343,94	327,27	259,09
Stop	42,11	-42,11	594,74
Remove	45,00	-40,00	545,00
Run	214,81	127,16	183,95

	vs Docker 18 (%)	vs Docker 24 (%)	vs Podman (%)
Create	87,42	-50,94	356,60
Start	1557,58	1544,56	868,63
Stop	30,08	-35,34	934,59
Remove	72,62	-8,33	1004,76
Run	588,90	590,66	526,25

Итоги проведенных сравнительных тестирований X86_64

Таблица сравнения: насколько процентов iSulad быстрее (зелёное поле) или медленнее (красное поле).

- Из результатов таблицы (в таблице приведены средние значения времени в миллисекундах, затраченных на операции при тестировании) видно, что iSula быстрее docker (версии 18.09) как в последовательном, так и в параллельном режиме выполнения команд.

- В сравнении с docker (версии 24.0.6) iSulad показал себя быстрее на операциях start, run.

- В противостоянии же с podman видно подавляющее преимущество iSula по всем командам сравнения.

Тестирование системы контейнерной виртуализации iSula

aarch64	average spent			
Последовательно (по 10 операций). Принимаем isula за 100%				
	Docker 18 (aarch64)	docker24 (aarch64)	podman (aarch64)	iSulad (aarch64)
Create	47	21	140	31
Start	522	452	287	75
Stop	39	14	141	26
Remove	41	16	140	26
Run	411	296	244	83
Параллельно (100 операций). Принимаем isula за 100%				
	Docker 18 (aarch64)	docker24 (aarch64)	podman (aarch64)	iSulad (aarch64)
Create	276	92	1042	183
Start	18922	19143	4591	789
Stop	1303	105	1526	121
Remove	160	106	993	114
Run	9791	8122	2171	874

	vs Docker 18 (%)	vs Docker 24 (%)	vs Podman (%)
Create	51,61	-32,26	351,61
Start	596,00	502,67	282,67
Stop	50,00	-46,15	442,31
Remove	57,69	-38,46	438,46
Run	395,18	256,63	193,98

	vs Docker 18 (%)	vs Docker 24 (%)	vs Podman (%)
Create	50,82	-49,73	469,40
Start	2298,23	2326,24	481,88
Stop	976,86	-13,22	1161,16
Remove	40,35	-7,02	771,05
Run	1020,25	829,29	148,40

Итоги проведенных сравнительных тестирований ARM64

Таблица сравнения: на сколько процентов iSulad быстрее (зелёное поле) или медленнее (красное поле).

Мы видим приблизительно такую же картину, что и на платформе x86_64, но отставание iSulad от docker (версии 24) в выполнении команд create, stop, remove незначительно сократилось, а преимущество в выполнении команд start и run значительно выше.

Некоторые наши опыты, тестирования и сборки под Edge-устройства

Железки

Технические характеристики nanoPi M4:

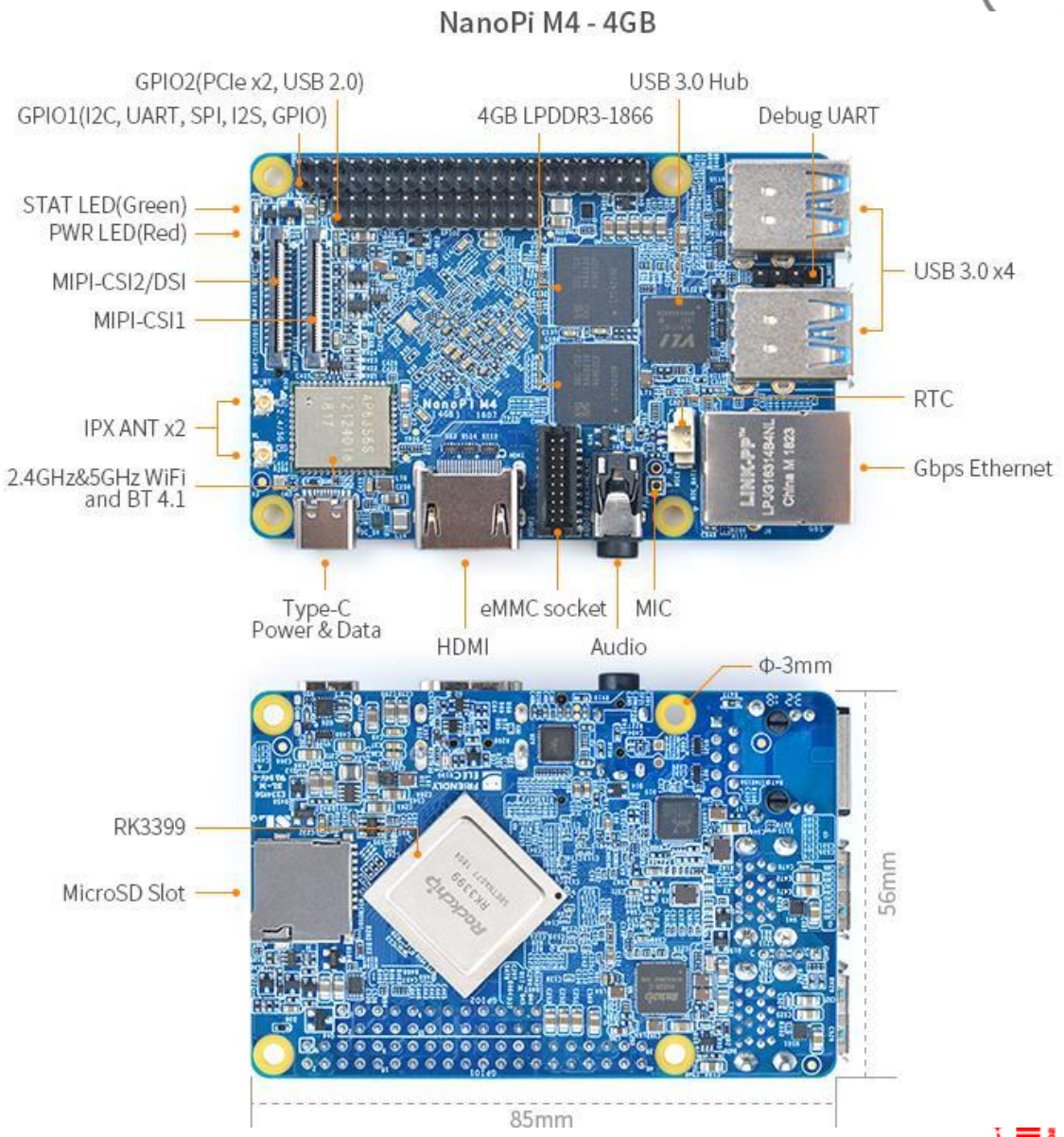
SoC: Rockchip RK3399
- CPU: big.LITTLE, Dual-Core Cortex-A72(up to 2.0GHz) + Quad-Core Cortex-A53(up to 1.5GHz)
- GPU: Mali-T864 GPU, supports OpenGL ES1.1/2.0/3.0/3.1, OpenCL, DX11, and AFBC
- VPU: 4K VP9 and 4K 10bits H265/H264 60fps decoding, Dual VOP, etc

RAM: Dual-Channel 4GB LPDDR3-1866, or Dual-Channel 2GB DDR3-1866
Flash: no Onboard eMMC, but has a eMMC socket
Ethernet: Native Gigabit Ethernet
Wi-Fi/BT: 802.11a/b/g/n/ac, Bluetooth 4.1, Wi-Fi and Bluetooth combo module, 2x2 MIMO, dual antenna interface
Video Input: one or two 4-Lane MIPI-CSI, dual ISP, up to 13MPix/s, supports simultaneous input of dual camera data
Video output
- HDMI: HDMI 2.0a, supports 4K@60Hz, HDCP 1.4/2.2
- one 4-Lane MIPI-DSI

Audio Out: 3.5mm Dual channel headphone jack, or HDMI
Audio In: one microphone input interface
USB 3.0: four USB 3.0 Type-A ports
USB Type-C: Supports USB2.0 OTG and Power input
microSD Slot x 1
40Pin GPIO Extension ports:
- 3 X 3V/1.8V I2C, up to 1 x 3V UART, 1 X 3V SPI, 1 x SPDIF_TX, up to 8 x 3V GPIOs
- 1 x 1.8V 8 channels I2S

24Pin Extension ports:
- 2 independent native USB 2.0 Host
- PCIe x2
- PWM x1, PowerKey

Debug: one Debug UART, 4 Pin 2.54mm header, 3V level, 1500000bps
PCB: 8 Layer, 85 mm x 56 mm



Прошивка, чёрный ящик?

4.2.2 Flashing the OS to the microSD card

Follow the steps below:

- Get an 8G microSD card;
- Visit download link to download image files (in the "01_Official images/01_SD card images" directory);
- Download the win32diskimager tool (in the "05_Tools" directory), or use your preferred tool;
- Extract the .gz format compressed file to get the .img format image file;
- Run the win32diskimager utility under Windows as administrator. On the utility's main window select your SD card's drive, the wanted image file and click on "write" to start flashing the SD card.
- Take out the SD and insert it to NanoPi-M4's microSD card slot;
- Power on NanoPi-M4 and it will be booted from your TF card, some models may require pressing the Power button to start;

Icon	Image Filename	Version	Description	Kernel Version
	rk3399-XYZ-android10-YYYYMMDD.img.zip	10	Android 10	4.4.y
	rk3399-XYZ-android8-YYYYMMDD.img.zip	8.1	Android 8.1	4.4.y
	rk3399-XYZ-android7-YYYYMMDD.img.zip	7.1.2	Android 7.1.2	4.4.y
	rk3399-XYZ-debian-bullseye-minimal-4.19-arm64-YYYYMMDD.img.gz	bullseye	Debian 11 Desktop, LXDE desktop, no pre-installed recommended software, supports hardware acceleration	4.19.y
	rk3399-XYZ-debian-bullseye-desktop-4.19-arm64-YYYYMMDD.img.gz	bullseye	Debian 11 Desktop, LXDE desktop, pre-installed mpv, smplayer and chromium brower, supports hardware acceleration	4.19.y
	rk3399-XYZ-debian-bullseye-core-6.1-arm64-YYYYMMDD.img.gz	bullseye	**Debian 11(Bullseye) Core**, No desktop environment, command line only	6.1.y
	rk3399-XYZ-ubuntu-focal-desktop-4.19-arm64-YYYYMMDD.img.gz	focal	Ubuntu 20.04 Desktop, LXQT desktop, pre-installed mpv, smplayer and chromium brower, supports hardware acceleration	4.19.y
	rk3399-XYZ-friendlydesktop-bionic-4.4-arm64-YYYYMMDD.img.zip	bionic	64-bit FriendlyDesktop image file based on Ubuntu desktop 18.04 64bit	4.4.y
	rk3399-XYZ-friendlycore-focal-4.19-arm64-YYYYMMDD.img.gz	focal	64-bit FriendlyCore image file(Qt 5.10.0) based on Ubuntu core 20.04 64bit	4.19.y

Скачать бинарник из интернета и установить:
dd if=sd.image of=/dev/sdb

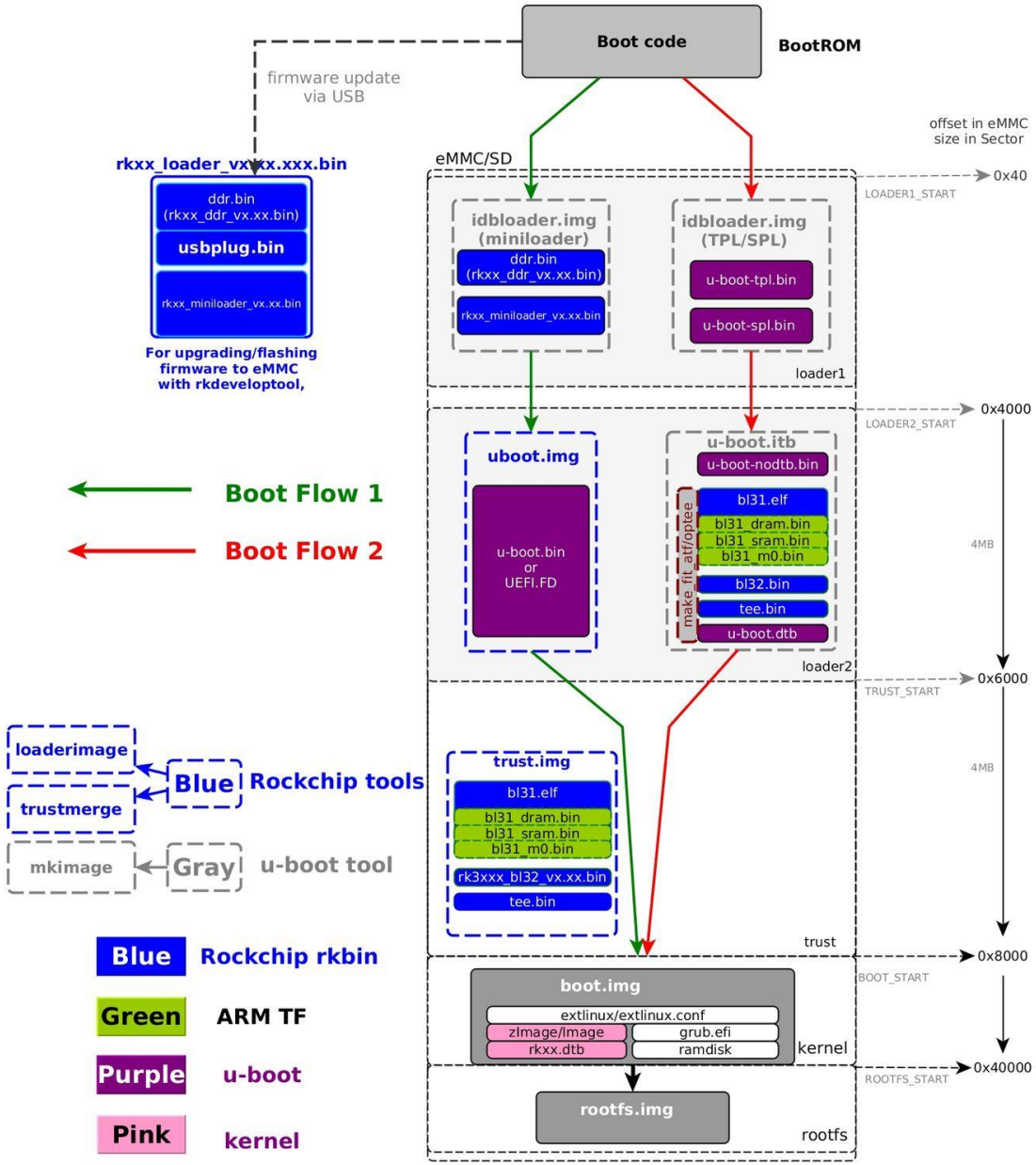
Или скачать и запустить какой-то скрипт, также из интернета:
wget https://где-то-в-интернете/install.sh
bash install.sh

Процесс загрузки

Как загружается PC:
- BIOS/UEFI
- GRUB
- Kernel
- OS

Как загружается одноплатник:
- ROM
- Idbloader.img
- Uboot.img
- Trust.img
- Boot.img
- Rootfs.img

Целевая схема:
Boot code
uBoot
Kernel
OS



Build Toolchain


```
mkdir -p tpl/dts/
  FDTGREP tpl/dts/dt-tpl.dtb
  COPY    tpl/u-boot-tpl.dtb
  CAT     tpl/u-boot-tpl-dtb.bin
  COPY    tpl/u-boot-tpl.bin
  SYM     tpl/u-boot-tpl.sym
  COPY    u-boot.dtb
  MKIMAGE u-boot-dtb.img
  BINMAN  .binman_stamp
Image 'simple-bin' is missing external blobs and is non-functional: atf-bl31

/binman/simple-bin/fit/images/@atf-SEQ/atf-bl31 (atf-bl31):
   See the documentation for your board. You may need to build ARM Trusted
   Firmware and build with BL31=/path/to/bl31.bin

Image 'simple-bin' is missing optional external blobs but is still functional: tee-os

/binman/simple-bin/fit/images/@tee-SEQ/tee-os (tee-os):
   See the documentation for your board. You may need to build Open Portable
   Trusted Execution Environment (OP-TEE) and build with TEE=/path/to/tee.bin

Some images are invalid
make: *** [Makefile:1124: .binman_stamp] Ошибка 103
[root@soc-devel u-boot]#
```

Расширяем пакетную базу:

- arm-none-eabi-binutils-cs
- arm-none-eabi-newlib
- arm-none-eabi-gcc-cs
- plcloog
- gprolog
- ppl

2.7.2.4. EL3 Runtime Firmware: `AP_BL31`

Also known as “SoC AP firmware” or “EL3 monitor firmware”. Its primary purpose is to handle transitions between the normal and secure world.

Build U-boot


```
[root@soc u-boot]# git branch -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/WIP/19Aug2019
  remotes/origin/WIP/2021-07-01-update-CI-containers
  remotes/origin/WIP/25Feb2021
  remotes/origin/dependabot/pip/test/py/setuptools-65.5.1
  remotes/origin/master
  remotes/origin/next
  remotes/origin/origin
  remotes/origin/u-boot-2009.11.y
  remotes/origin/u-boot-2013.01.y
  remotes/origin/u-boot-2016.09.y
  remotes/origin/u-boot-2023.07.y
[root@soc u-boot]# git checkout u-boot-2023.07.y
Ветка «u-boot-2023.07.y» отслеживает внешнюю ветку «u-boot-2023.07.y» из «origin».
Переключено на новую ветку «u-boot-2023.07.y»
```

Используется стабильный релиз uBoot и ARM Trusted Firmware

Собирать самостоятельно не надо, все есть в составе дистрибутива: uboot-images

Подготовка образа SD-карты

truncate -s 2G sd.img

losetup /dev/loop0 sd.img

fdisk /dev/loop0

dd if=u-boot-rockchip.bin of=/dev/loop0 seek=64

```
[root@soc image]# cp ../u-boot/u-boot-rockchip.bin .
[root@soc image]# dd u-boot-rockchip.bin ^C
[root@soc image]# dd if=u-boot-rockchip.bin of=/dev/loop0 seek=64
18214+0 записей получено
18214+0 записей отправлено
9325568 байт (9,3 MB, 8,9 MiB) скопирован, 0,136495 s, 68,3 MB/s
[root@soc image]#
```

```
[root@soc image]# fdisk /dev/loop0

Добро пожаловать в fdisk (util-linux 2.37.2).
Изменения останутся только в памяти до тех пор, пока вы не решите записать их.
Будьте внимательны, используя команду write.

Устройство не содержит стандартной таблицы разделов.
Создана новая метка DOS с идентификатором 0x93ae4fa4.

Команда (m для справки): n
Тип раздела
   p   основной (0 primary, 0 extended, 4 free)
   e   расширенный (контейнер для логических разделов)
Выберите (по умолчанию - p):p
Номер раздела (1-4, default 1):
Первый сектор (2048-4194303, default 2048): 20480
Last sector, +/-sectors or +/-size{K,M,G,T,P} (20480-4194303, default 4194303):

Создан новый раздел 1 с типом 'Linux' и размером 2 GiB.

Команда (m для справки): p
Диск /dev/loop0: 2 GiB, 2147483648 байт, 4194304 секторов
Единицы: секторов по 1 * 512 = 512 байт
Размер сектора (логический/физический): 512 байт / 512 байт
Размер I/O (минимальный/оптимальный): 512 байт / 512 байт
Тип метки диска: dos
Идентификатор диска: 0x93ae4fa4

Устр-во      Загрузочный начало   Конец Секторы Размер Идентификатор Тип
/dev/loop0p1             20480 4194303 4173824     2G           83 Linux
```

HighLoad++

Установка ОС в каталог

rpm --root /root/nanoPi-M4/image/mnt/ --initdb

Перед установкой rpm-пакетов рекомендуется отключить установку слабых зависимостей, например, к ним относится grub, который не нужен в данной инсталяции. Для этого необходимо добавить строку в файл /etc/yum.conf:

install_weak_deps=False

Теперь необходимо установить минимальный набор пакетов:

dnf install --installroot /root/nanoPi-M4/image/mnt dnf yum alsa-utils haveged wpa_supplicant vim net-tools iproute iputils NetworkManager openssh-server openssh-clients passwd hostname bluez pulseaudio-module-bluetooth parted linux-firmware gdisk ntp bc systemd-timesyncd openScaler-repos

Chroot’имся в новую ОС

Добавляем репозиторий:

/etc/yum.repos.d/extra.repo

[extra]
name=extra packages
baseurl=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/extra/$basearch
enabled=1
gpgcheck=1
gpgkey=http://repo.openscaler.ru/openScaler-22.03-LTS-SP2/OS/$basearch/RPM-GPG-KEY-openScaler

```
[root@soc image]# blkid
/dev/mapper/openscaler-swap: UUID="44f608f2-89f5-4a92-bc01-4361fee8818f" TYPE="swap"
/dev/loop0p1: UUID="251c6b52-70b3-4707-84b2-9f78325f6c84" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="93ae4fa4-01"
/dev/mapper/openscaler-root: UUID="ba71612b-11c4-4c6a-976e-7efd9242a777" BLOCK_SIZE="4096" TYPE="ext4"
/dev/sda2: UUID="9ea19dd3-f013-431b-a969-4871c0702868" BLOCK_SIZE="4096" TYPE="ext4" PARTUUID="a0a3f5cc-1d1a-41e5-ba4b-6269a7e0dcfd"
/dev/sda3: UUID="lRcRZD-wfaF-G40E-jZvc-J8JJ-4I4m-Ib5qkG" TYPE="LVM2_member" PARTUUID="d69e931b-3ff8-4ec3-a62a-ba64e8fc4968"
/dev/sda1: UUID="1F2B-6182" BLOCK_SIZE="512" TYPE="vfat" PARTLABEL="EFI System Partition" PARTUUID="fe3770b9-e07c-4181-bc50-423b9c5012c8"
```

cat /etc/fstab

UUID=251c6b52-70b3-4707-84b2-9f78325f6c84 / etx4 defaults 1 1

Установка ядра и загрузчика

dnf install rockchip-kernel dracut uboot-tools

Подготовим initrd-образ и сделаем его доступным для u-boot:

```
dracut --kver 5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64
mkimage -A arm64 -T ramdisk -C none -n uInitrd -d /boot/initramfs-
5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64.img /boot/uInitrd-
5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64
cd /boot
ln -s uInitrd-5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64 uInitrd
```

```
bash-5.1# mkimage -A arm64 -T ramdisk -C none -n uInitrd -
Image Name:   uInitrd
Created:      Fri Nov  3 08:52:21 2023
Image Type:   AArch64 Linux RAMDisk Image (uncompressed)
Data Size:    20873500 Bytes = 20384.28 KiB = 19.91 MiB
Load Address: 00000000
Entry Point:  00000000
```

```
bash-5.1# mkimage -C none -A arm64 -T script -d /boot/boot.cmd /boot/boot.scr
Image Name:
Created:      Fri Nov  3 09:52:58 2023
Image Type:   AArch64 Linux Script (uncompressed)
Data Size:    681 Bytes = 0.67 KiB = 0.00 MiB
Load Address: 00000000
Entry Point:  00000000
Contents:
   Image 0: 673 Bytes = 0.66 KiB = 0.00 MiB
```

cat /boot/boot.cmd

```
################################################
# DO NOT EDIT THIS FILE boot.scr
# Please edit boot.cmd
# And run: mkimage -C none -A arm64 -T script -d /boot/boot.cmd
/boot/boot.scr
################################################
echo "Boot script loaded from ${devtype}"
setenv bootargs "root=/dev/mmcblk1p1 ro rootfstype=ext4 earlyprintk
rootwait console=ttyS2,1500000 console=tty1 consoleblank=0"
load ${devtype} ${devnum} ${kernel_addr_r} ${prefix}Image
load ${devtype} ${devnum} ${ramdisk_addr_r} ${prefix}uInitrd
load ${devtype} ${devnum} ${fdt_addr_r} ${prefix}/dtb/rk3399-nanopi-m4.dtb
fdt addr ${fdt_addr_r}
booti ${kernel_addr_r} ${ramdisk_addr_r} ${fdt_addr_r}
```

Пробуем взлететь

minicom - наш друг

```
Hit any key to stop autoboot:  0
** Booting bootflow 'mmc@fe320000.bootdev.part_1' with script
Boot script loaded from mmc
27509248 bytes read in 1166 ms (22.5 MiB/s)
20873564 bytes read in 886 ms (22.5 MiB/s)
55893 bytes read in 9 ms (5.9 MiB/s)
Working FDT set to 1f00000
Moving Image from 0x2080000 to 0x2200000, end=3d00000
## Loading init Ramdisk from Legacy Image at 06000000 ...
   Image Name:   uInitrd
   Image Type:   AArch64 Linux RAMDisk Image (uncompressed)
   Data Size:    20873500 Bytes = 19.9 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 01f00000
   Booting using the fdt blob at 0x1f00000
Working FDT set to 1f00000
   Loading Ramdisk to efb32000, end f0f1a11c ... OK
   Loading Device Tree to 00000000efb21000, end 00000000efb31a54 ... OK
Working FDT set to efb21000

Starting kernel ...

[    0.000000] Booting Linux on physical CPU 0x0000000000 [0x410fd034]
[    0.000000] Linux version 5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64 (abu
[    0.000000] Machine model: FriendlyElec NanoPi M4
[    0.000000] efi: UEFI not found.
```

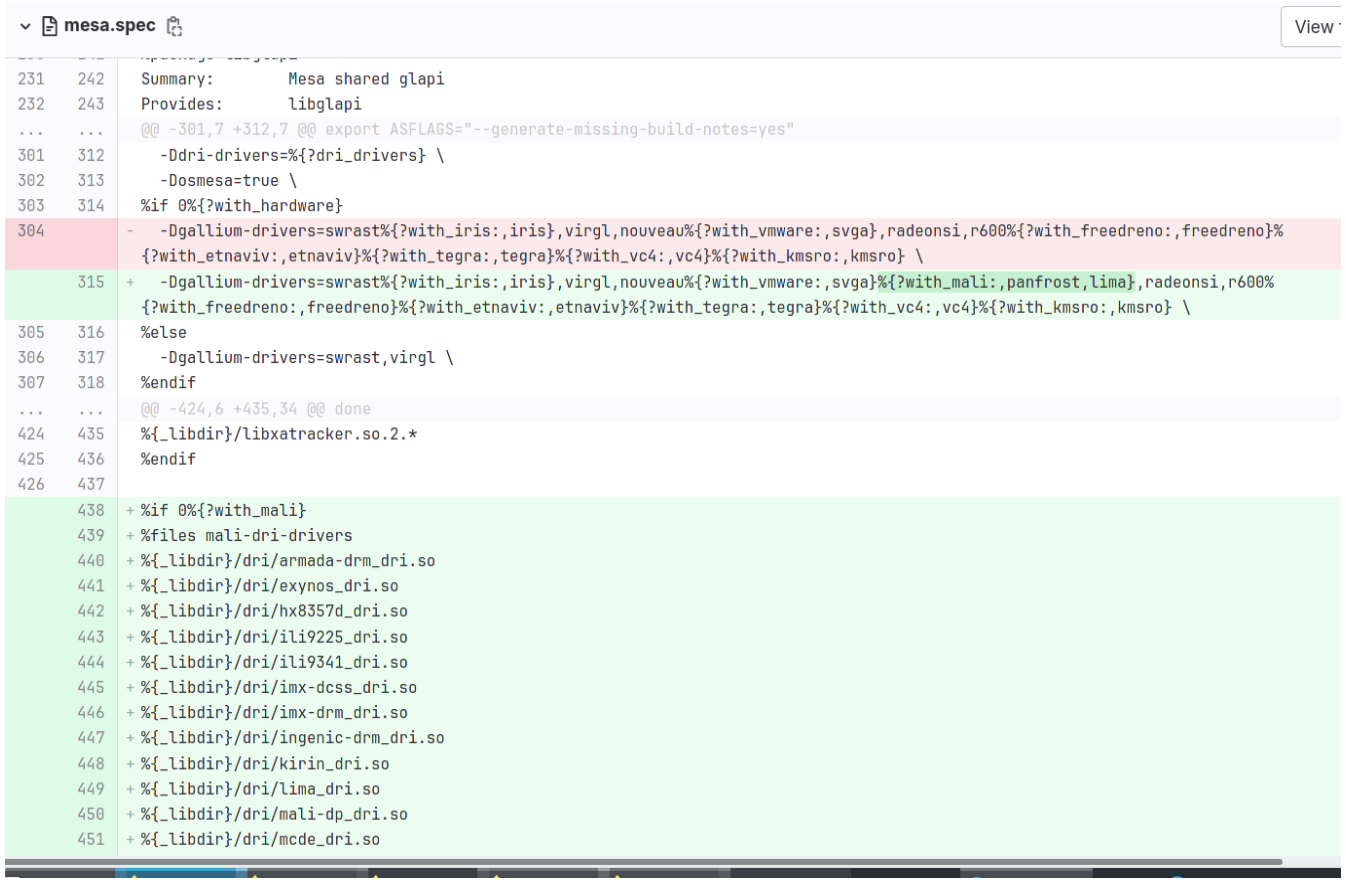
```
ПАРАМЕТРЫ: I18n
Дата компиляции Apr 20 2023, 23:14:00.
Port /dev/ttyUSB0, 12:39:54

Нажмите CTRL-A Z для получения подсказки по клавишам


U-Boot TPL 2023.07.02 (Nov 03 2023 - 09:46:42)
Channel 0: LPDDR3, 933MHz
BW=32 Col=10 Bk=8 CS0 Row=15 CS1 Row=15 CS=2 Die BW=16 Size=2048MB
Channel 1: LPDDR3, 933MHz
BW=32 Col=10 Bk=8 CS0 Row=15 CS1 Row=15 CS=2 Die BW=16 Size=2048MB
256B stride
Trying to boot from BOOTROM
Returning to boot ROM...

U-Boot SPL 2023.07.02 (Nov 03 2023 - 09:46:42 +0300)
Trying to boot from MMC1
spl_load_fit_image: Skip load 'atf-5': image size is 0!


U-Boot 2023.07.02 (Nov 03 2023 - 09:46:42 +0300)

SoC: Rockchip rk3399
Reset cause: POR
Model: FriendlyElec NanoPi M4
DRAM:  4 GiB (effective 3.9 GiB)
PMIC:  RK808
Core:  277 devices, 25 uclasses, devicetree: separate
MMC:   mmc@fe310000: 3, mmc@fe320000: 1, mmc@fe330000: 0
Loading Environment from MMC... Card did not respond to voltage select! : -110
*** Warning - No block device, using default environment

rk3399_vop vop@ff8f0000: failed to get ahb reset (ret=-524)
rk3399_vop vop@ff8f0000: failed to get ahb reset (ret=-524)
In:    serial
Out:   serial
Err:   serial
Model: FriendlyElec NanoPi M4
Net:
Error: ethernet@fe300000 address not set.
No ethernet found.

Hit any key to stop autoboot:  0
```

Графика и драйверы


```
egrep -i 'mali|rockchip' /var/log/Xorg.0.log
[     14.647] Current Operating System: Linux openScaler-
nanoPi-M4 5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64 #1
SMP PREEMPT Mon Oct 30 12:18:44 UTC 2023 aarch64
```

```
dnf install mesa-mali-dri-drivers
```

```
egrep -i 'mali|rockchip' /var/log/Xorg.0.log
[     14.562] Current Operating System: Linux openScaler-
nanoPi-M4 5.10.0-153.12.0.93.os2203sp2.rockchip.aarch64 #1
SMP PREEMPT Mon Oct 30 12:18:44 UTC 2023 aarch64
[     18.840] (II) modeset(0): glamor X acceleration enabled on
Mali-T860 (Panfrost)
[     19.114] (II) modeset(0): [DRI2]   DRI driver: rockchip
[     19.115] (II) modeset(0): [DRI2]   VDPAU driver: rockchip
[     19.146] (II) AIGLX: Loaded and initialized rockchip
```



mesa.spec

```
231 242  Summary:        Mesa shared glapi
232 243  Provides:       libglapi
... ...  @@ -301,7 +312,7 @@ export ASFLAGS="--generate-missing-build-notes=yes"
301 312    -Ddri-drivers=%{?dri_drivers} \
302 313    -Dosmesa=true \
303 314  %if 0%{?with_hardware}
304      -   -Dgallium-drivers=swrast%{?with_iris:,iris},virgl,nouveau%{?with_vmware:,svga},radeonsi,r600%{?with_freedreno:,freedreno}%
         {?with_etnaviv:,etnaviv}%{?with_tegra:,tegra}%{?with_vc4:,vc4}%{?with_kmsro:,kmsro} \
     315 +   -Dgallium-drivers=swrast%{?with_iris:,iris},virgl,nouveau%{?with_vmware:,svga}%{?with_mali:,panfrost,lima},radeonsi,r600%
         {?with_freedreno:,freedreno}%{?with_etnaviv:,etnaviv}%{?with_tegra:,tegra}%{?with_vc4:,vc4}%{?with_kmsro:,kmsro} \
305 316  %else
306 317    -Dgallium-drivers=swrast,virgl \
307 318  %endif
... ...  @@ -424,6 +435,34 @@ done
424 435  %{_libdir}/libxatracker.so.2.*
425 436  %endif
426 437
     438 + %if 0%{?with_mali}
     439 + %files mali-dri-drivers
     440 + %{_libdir}/dri/armada-drm_dri.so
     441 + %{_libdir}/dri/exynos_dri.so
     442 + %{_libdir}/dri/hx8357d_dri.so
     443 + %{_libdir}/dri/ili9225_dri.so
     444 + %{_libdir}/dri/ili9341_dri.so
     445 + %{_libdir}/dri/imx-dcss_dri.so
     446 + %{_libdir}/dri/imx-drm_dri.so
     447 + %{_libdir}/dri/ingenic-drm_dri.so
     448 + %{_libdir}/dri/kirin_dri.so
     449 + %{_libdir}/dri/lima_dri.so
     450 + %{_libdir}/dri/mali-dp_dri.so
     451 + %{_libdir}/dri/mcde_dri.so
```

Спасибо!

Павлов Владимир

Варенов Дмитрий

https://t.me/openscaler

HL 2023 HighLoad++