Android Container

Cloud

简介

Android Cloud 作用：

automate app running at scale

stream games to millions

secure mobile apps

项目设计

云手机项目在商业上要想成功，首先需要和物理手机比拼单位成本，让客户愿意用，这就需要在服务器上做好性能优化，达到高虚拟机密度。然后，做好服务器的可扩展性管理，通过多租来获得更好的收益和更低的成本。

项目的启动资金较低，因为 amazon 提供 arm 虚拟机租用。可先在云上部署，待盈利明确后再考虑部署自己的机房。

渠道上，已有现成成熟渠道。

技术功能实现上，我一个多月独立完成。基于以前多年的 Android 底层和 Cloud 经验，并借助 Anbox,k8s 等开源项目，做起来比较快。

硬件平台：Ampere ARM server / Amazon ARM cloud instance

x86 平台需要有 hudini 模块对部分 apps 的 ARM ndk 代码进行 binary translation，某些情况下会有较大性能开销，故不选用 x86 平台。

Kunpeng920 ARM server 不支持 ARM 32bit instruction，需要商业版的转码软件 Exagear。故不选用。选用 Ampere ARM server。

OS: Ubuntu18 server ARM version. 很好用。软件生态比较成熟。

Android 虚拟化/容器化

主要工作量在于处理好 HAL 层的 graphics,gralloc,input,audio,network 等虚拟化。有 Android emulator 的大量代码可以复用。

Fundamental issue: one vs. multiple Android instances

- Resource access and usage

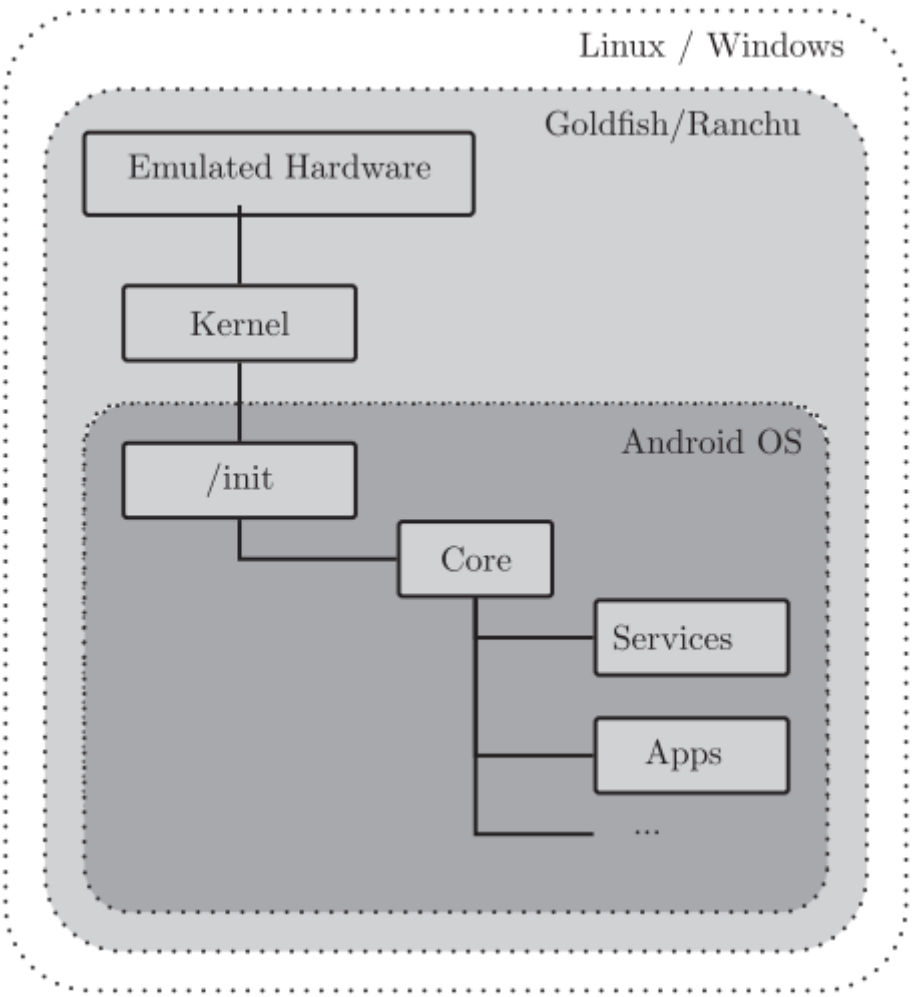
– Android’s resource usage assignment for process groups formed with cgroup

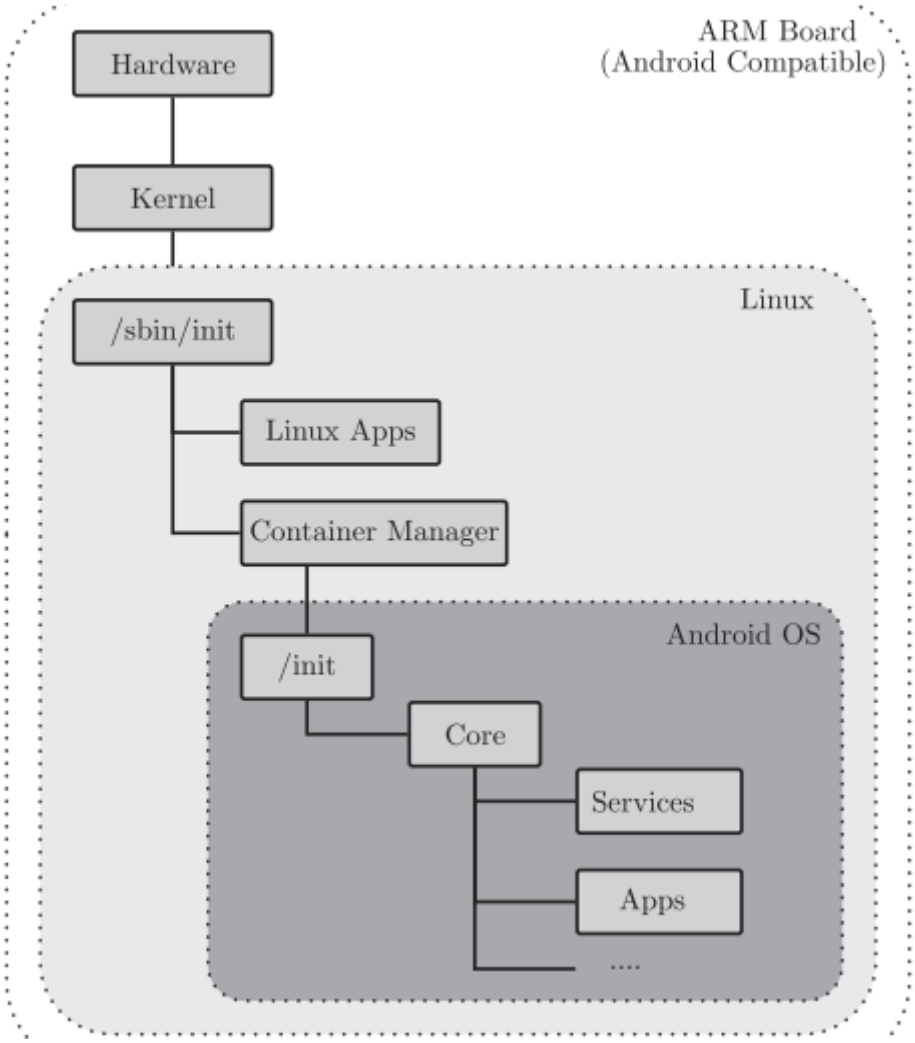
– Android’s use of SELinux

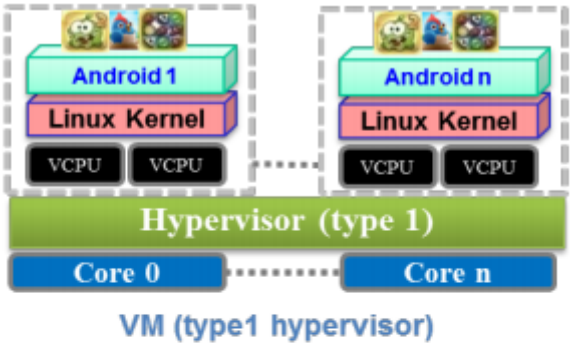
– Android’s permission settings for procfs (/proc) and sysfs(/sys)

- Device virtualization – Binding of virtual and local/remote physical peripheral devices – Fair sharing of local physical peripheral devices among host and containers

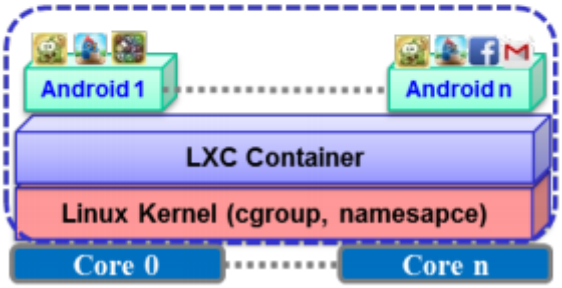
Commen design for Android emulator VS Android container


Linux / Windows
Goldfish/Ranchu
Emulated Hardware
Kernel
Android OS
/init
Core
Services
Apps
...



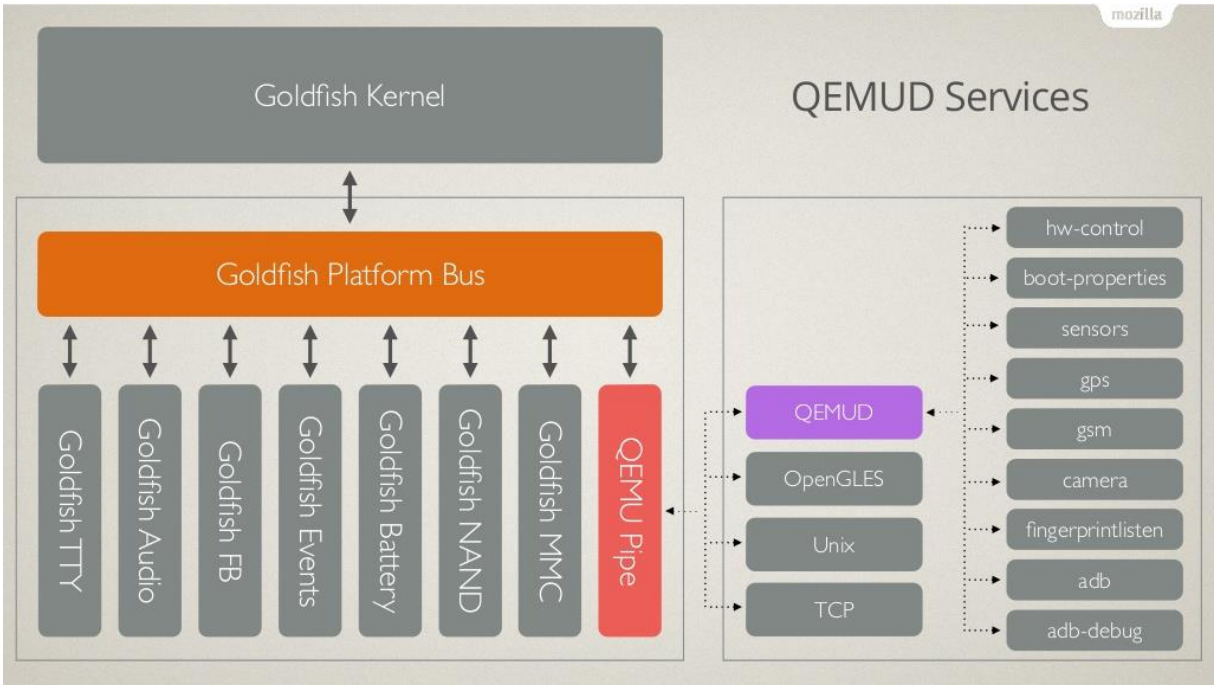


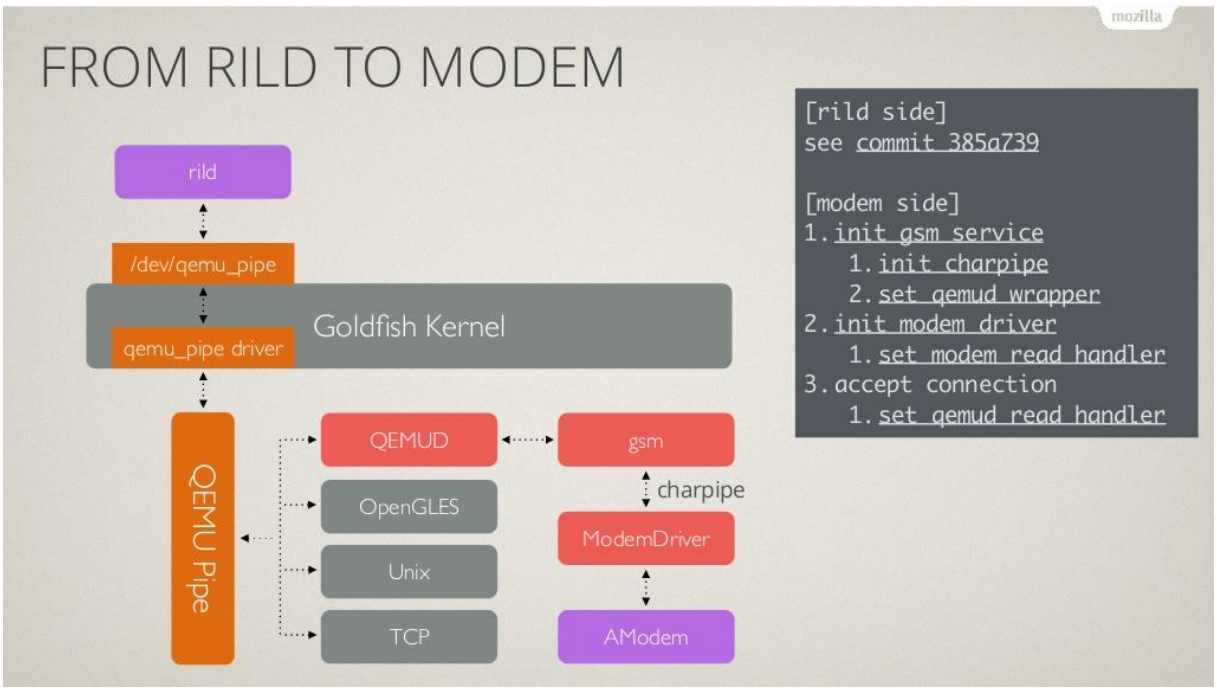
VM (type1 hypervisor)



LXC Container

The Android inside the container has no direct access to any hardware. All hardware access is going through the anbox daemon on the host. We're reusing what Android implemented within the QEMU-based emulator for OpenGL ES accelerated
rendering.

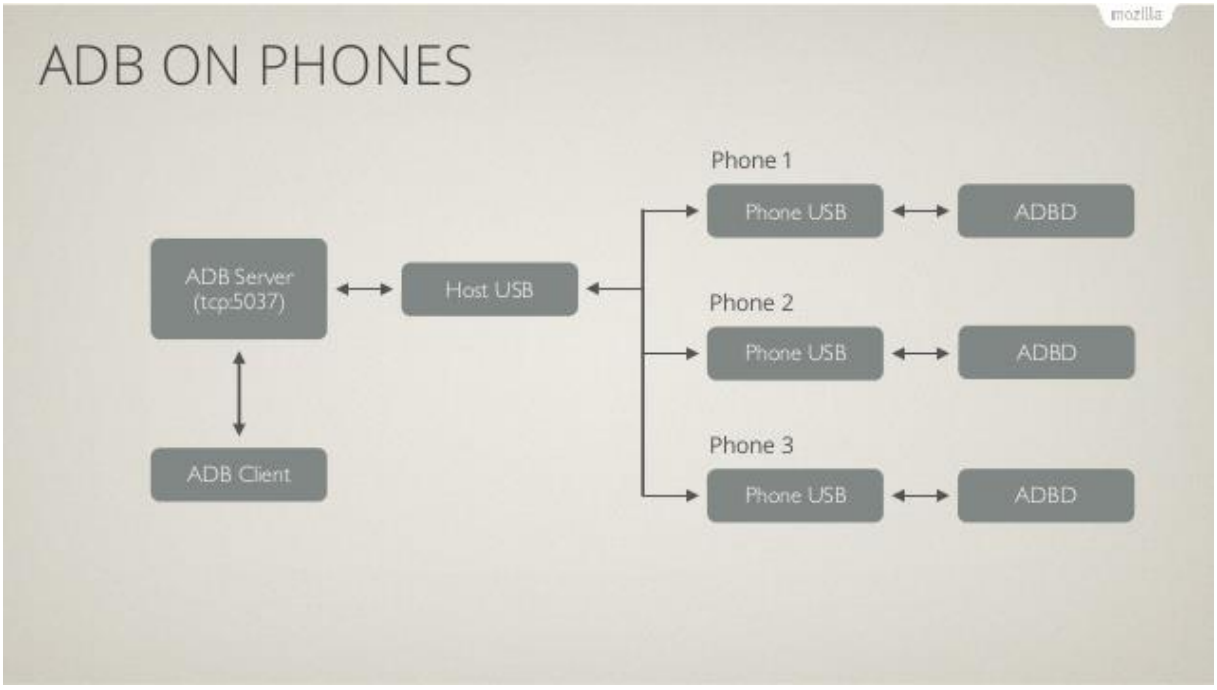


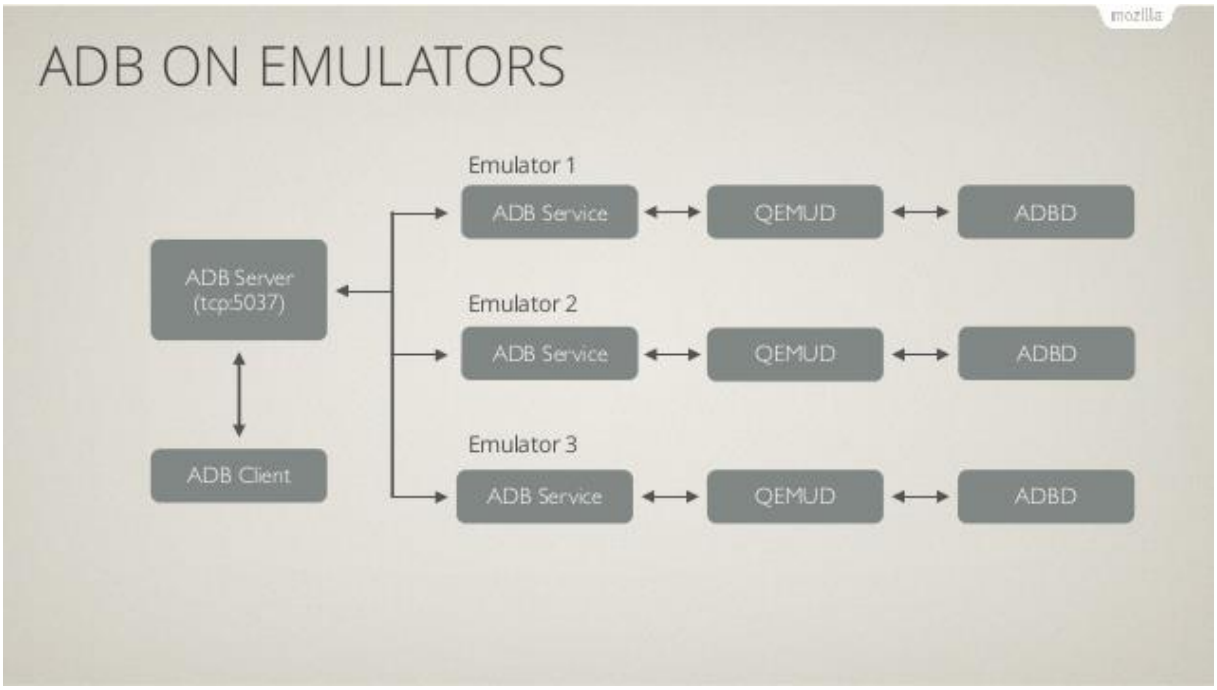


用容器的时候，并不需要 qemu_pipe driver。在 Anbox 里 qemu_pipe 只是一个 socket 文件，作为 volume 映射到容器，而不是 Ranchu/Goldfish 里的一个驱动。

--volume=$SOCKETDIR/$instance/sockets/qemu_pipe:/dev/qemu_pipe
--volume=$SOCKETDIR/$instance/sockets/anbox_audio:/dev/anbox_audio:rw
--volume=$SOCKETDIR/$instance/sockets/anbox_bridge:/dev/anbox_bridge:rw
--volume=$SOCKETDIR/$instance/input/event0:/dev/input/event0:rw
--volume=$SOCKETDIR/$instance/input/event1:/dev/input/event1:rw

ADB 虚拟





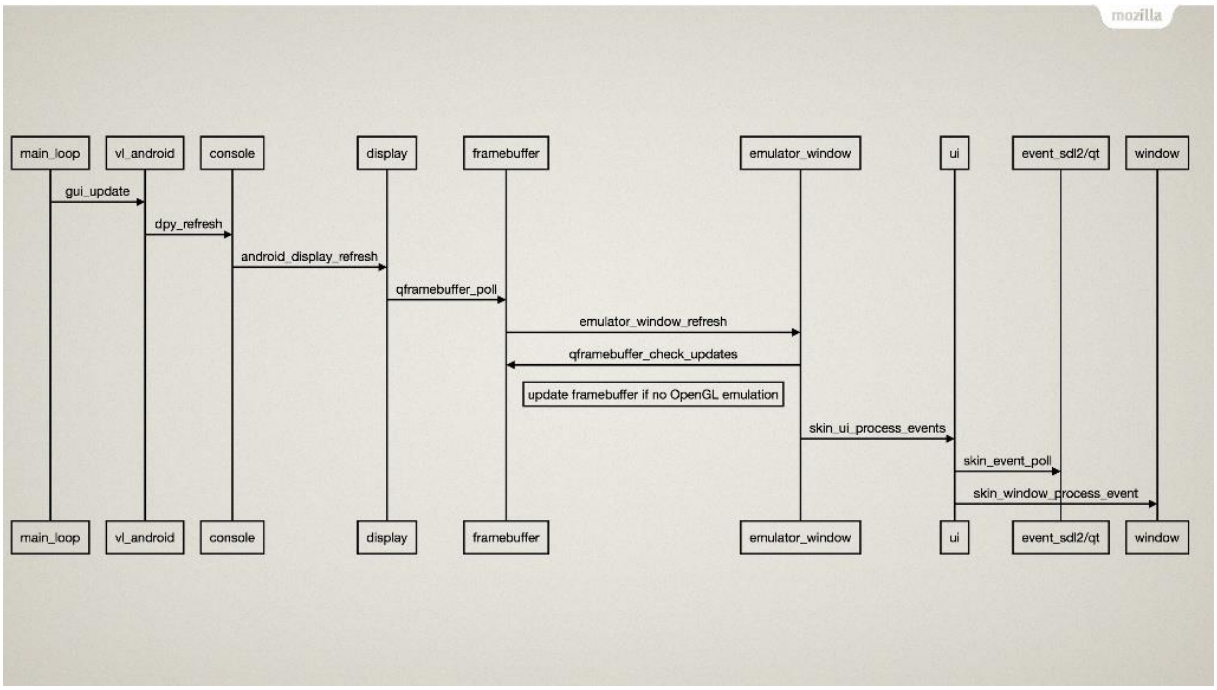
但是这种从 qemud 走的 ADB 连接，很容易在大量数据传输的时候断掉。直接把 ADB port 作为 container port 映射，稳定性更好。

Console 虚拟

Anbox 的 UI 实现基于 SDL2+Xwindow

SDL（Simple DirectMedia Layer）作为免费的跨平台多媒体应用编程接口，已经被人们广泛用于开发二维游戏，其优秀的消息框架支持、文件支持和声音支持都使得它能与微软 DirectX 匹敌。现在很多 game 游戏里面，都采用 SDL+OpenGL ES 的模式来绘制 3D 界面。 可以让 SDL 使用 OpenGL ES 的函数接口来渲染 3D。

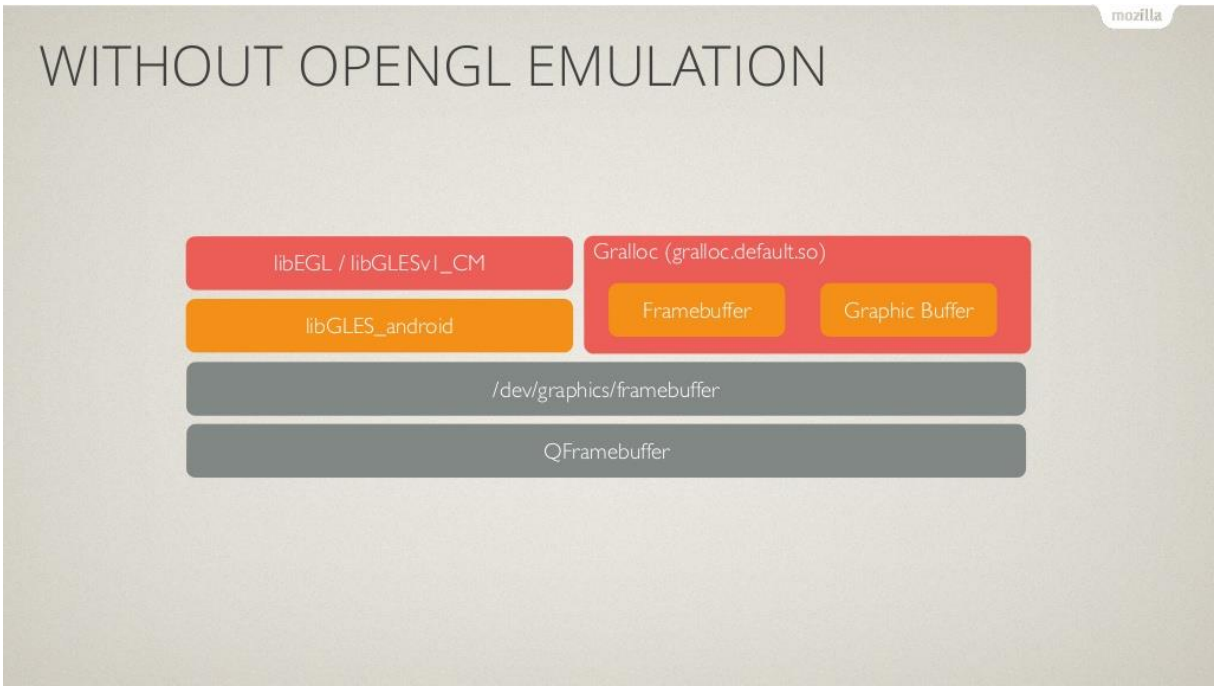
SDL 做的工作就是用 X11 创建窗口，用 EGL 创建 Surface 并绑定，最后就可以用 OpenGL 或者 GLES 去 render.

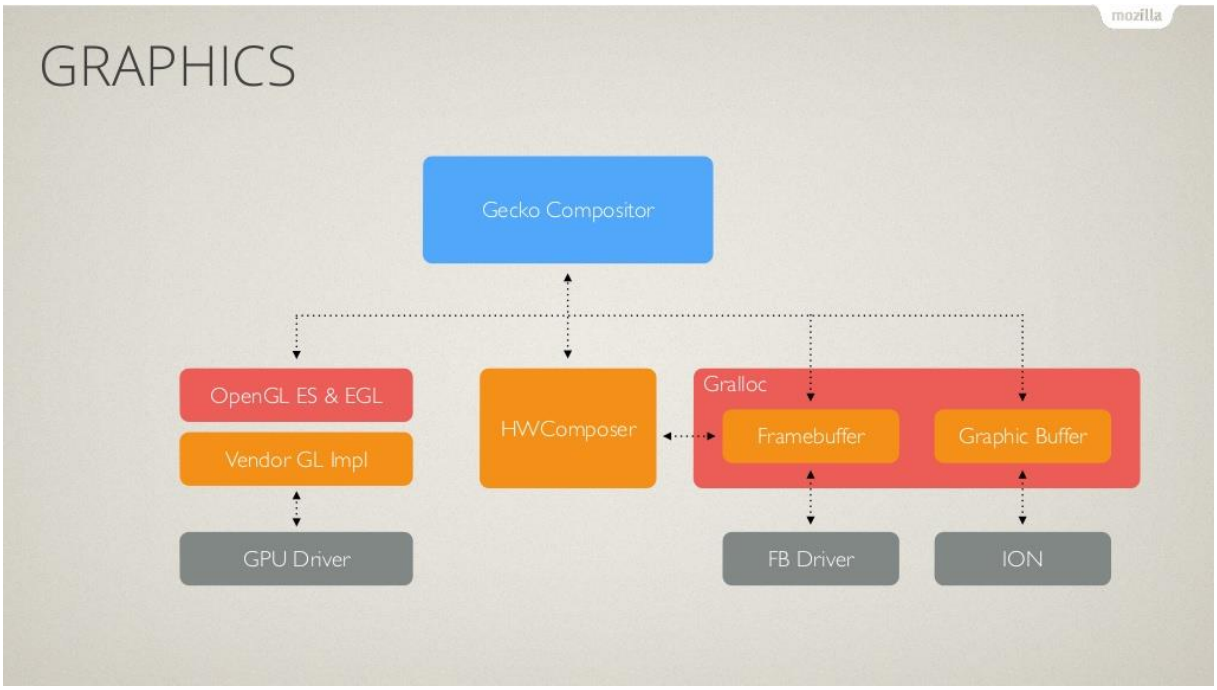


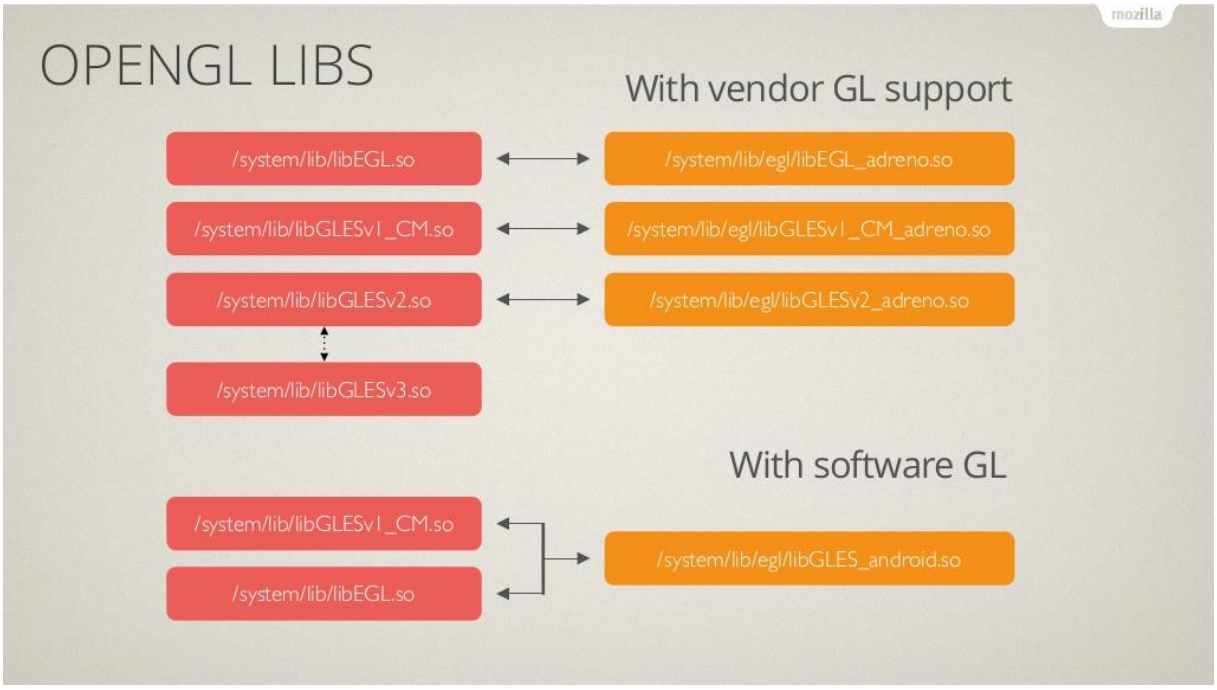
不过这种实现，在使用 MobaXterm 做 Xserver 远程显示时，由于 SDL2 的兼容问题，会失败。只能使用 VNCServer 做远程显示。

Graphics 虚拟

虚拟前：

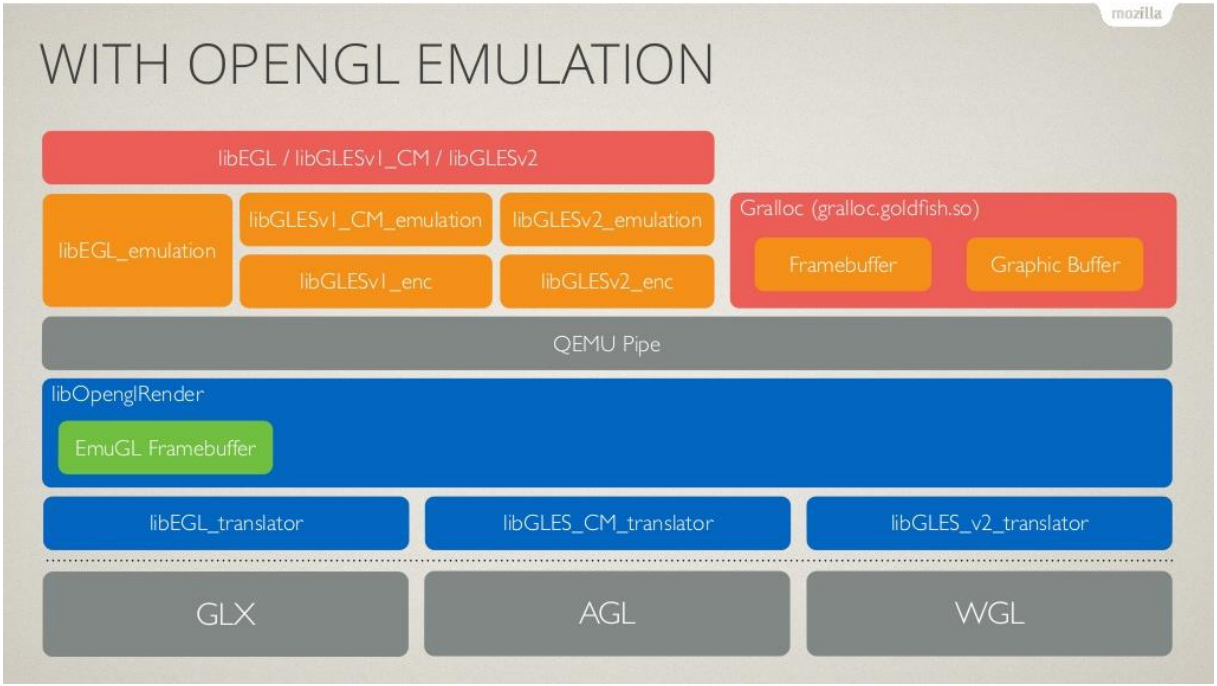


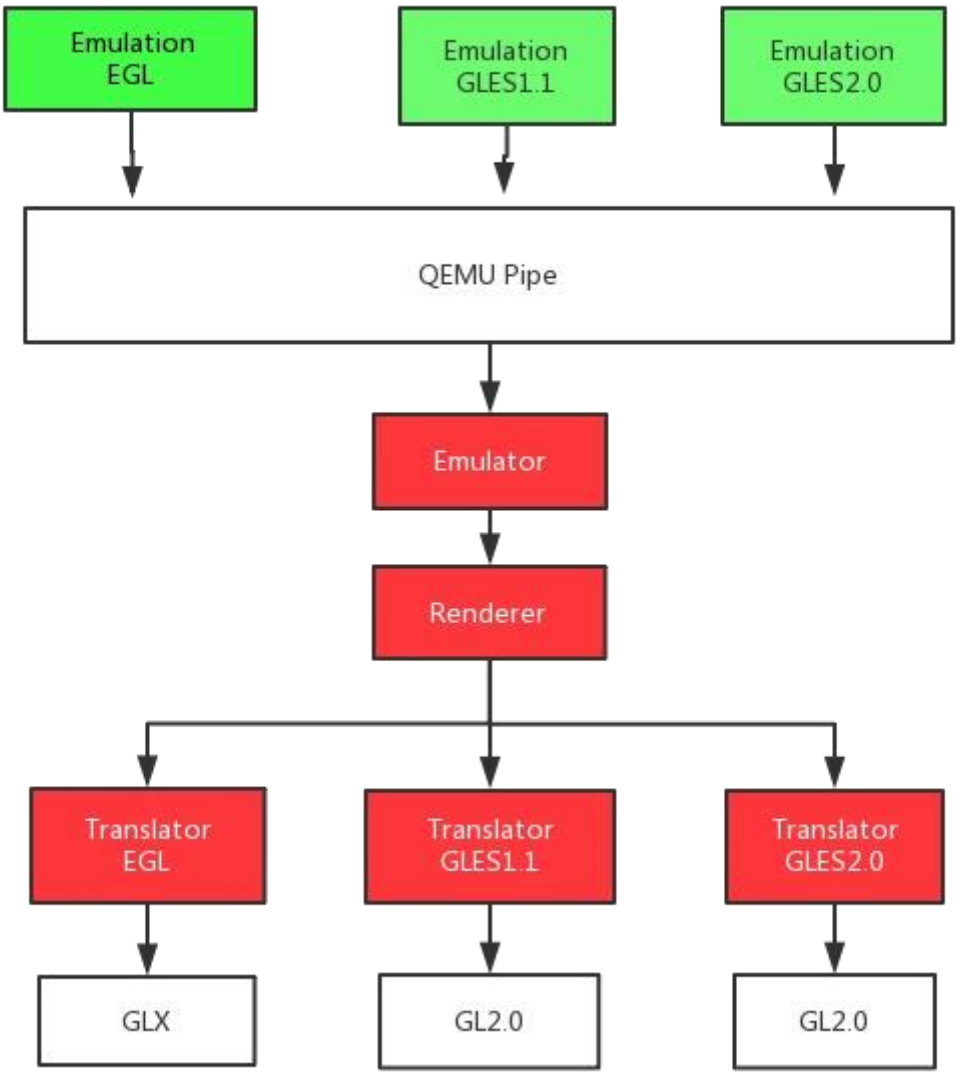

mozilla
GRAPHICS
Gecko Compositor
OpenGL ES & EGL
Vendor GL Impl
GPU Driver
HWComposer
Gralloc
Framebuffer
Graphic Buffer
FB Driver
ION


mozilla
OPENGL LIBS
With vendor GL support
/system/lib/libEGL.so
/system/lib/egl/libEGL_adreno.so
/system/lib/libGLESv1_CM.so
/system/lib/egl/libGLESv1_CM_adreno.so
/system/lib/libGLESv2.so
/system/lib/egl/libGLESv2_adreno.so
/system/lib/libGLESv3.so
With software GL
/system/lib/libGLESv1_CM.so
/system/lib/libEGL.so
/system/lib/egl/libGLES_android.so

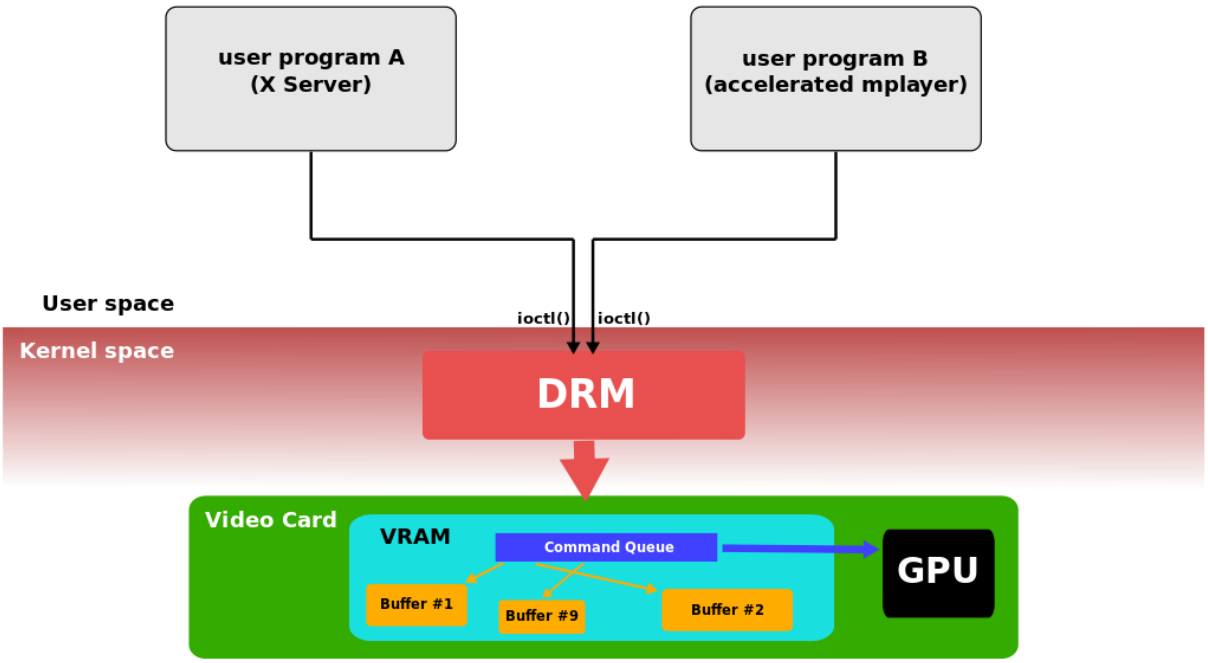
虚拟后：

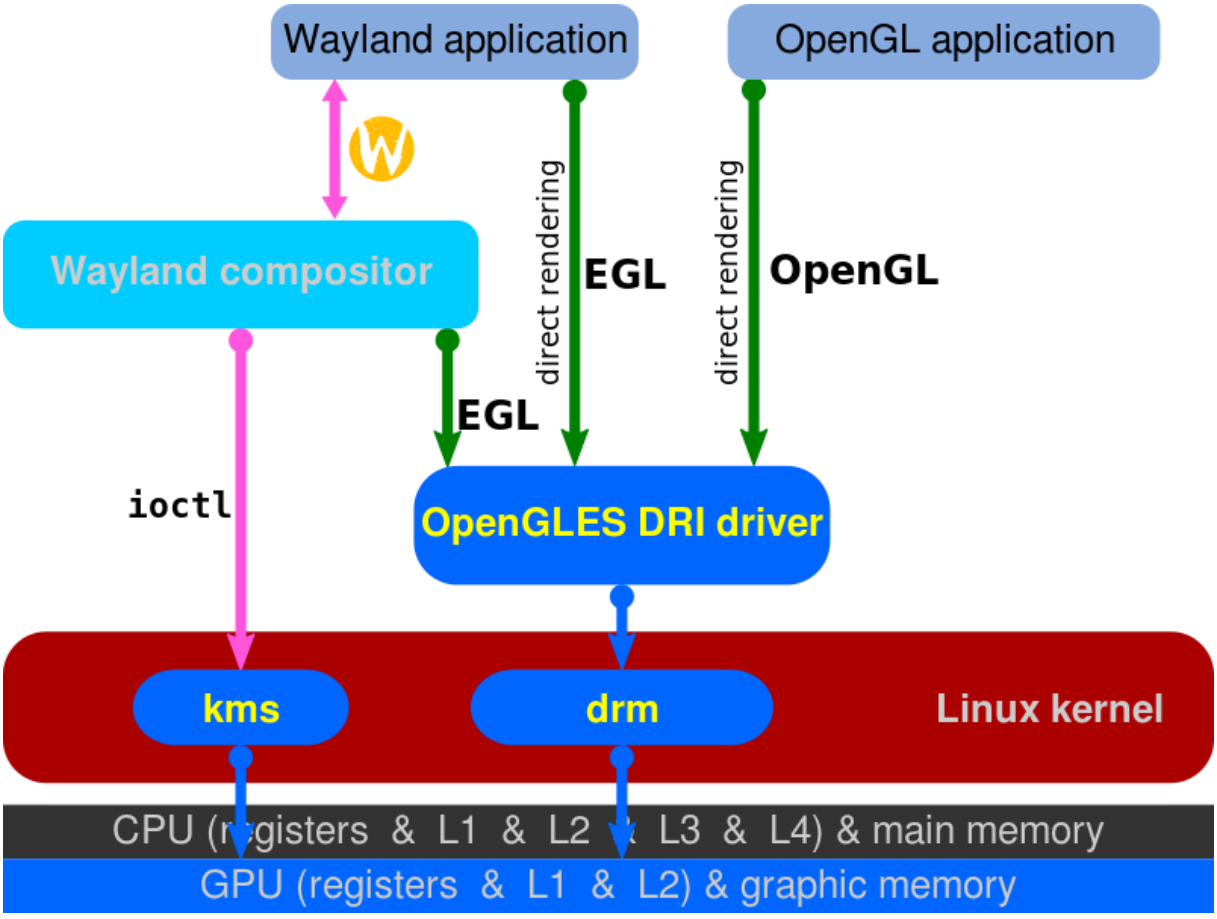
WITH OPENGL EMULATION

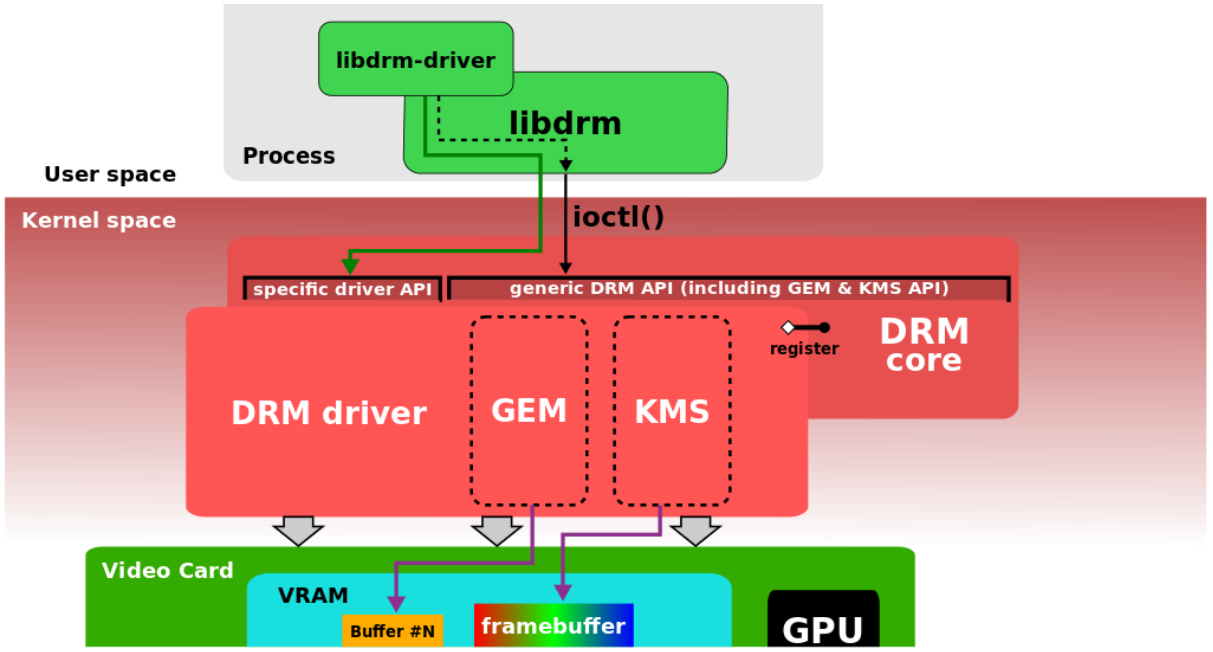




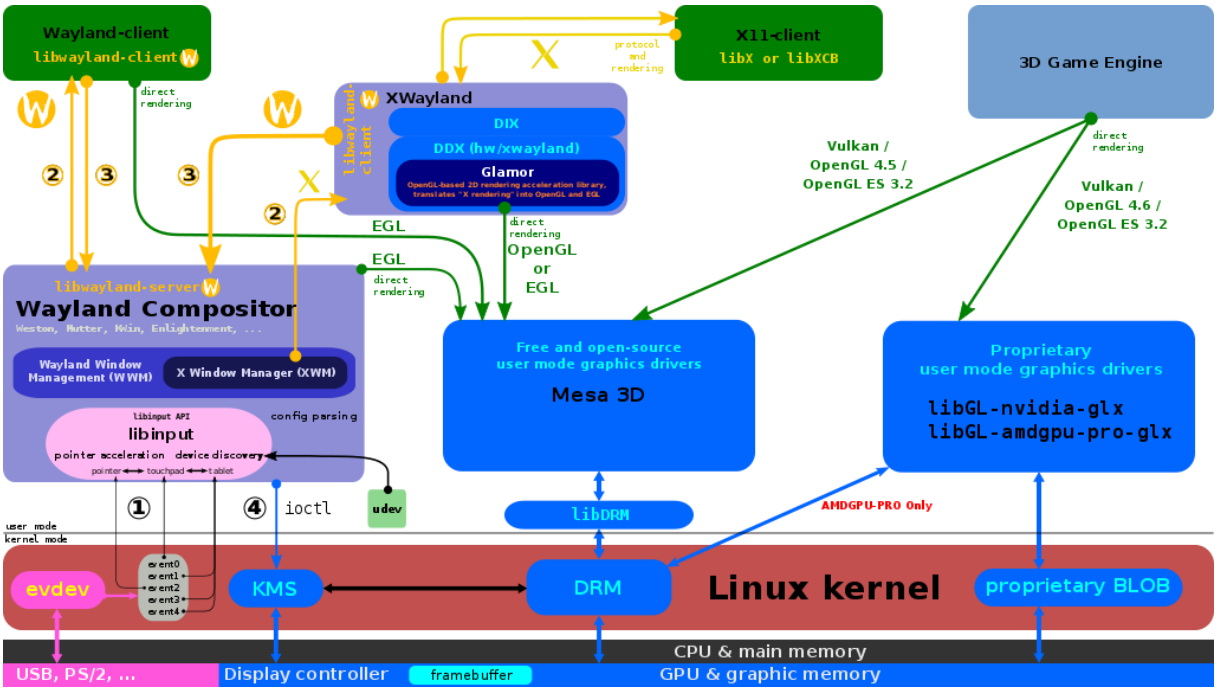
(NOTE: 'GLX' is for Linux only, replace 'AGL' on OS X, and 'WGL' on Windows)







Version 1



The X window system is a computer software system and network protocol that provides a basis GUIs for networked computers. It creates a hardware abstraction layer.

GLX enables programs wishing to use OpenGL to do so within a window provided by the X Window System.

GLX consists of three parts:

- An API that provides OpenGL functions.

- An extension of the X protocol, which allows the client to send 3D rendering commands - An extension of the X server that receives the rendering commands from the client and passes them on to the installed OpenGL library

If client and server are running on the same computer and an accelerated 3D graphics card is available, the former two components can be bypassed by DRI. The client program is then allowed to directly access the graphics hardware.

Direct Rendering Infrastructure (DRI) is an interface used in the X Window System to allow user applications to access the video hardware without requiring data to be passed through the X server.

Camera 虚拟

Anbox 复用了 android emulator 里关于 camera 虚拟的代码。可在其基础上增加图片图像的用户灌入。

Whenever you enable camera support in Anbox, you will get a video stream socket that can be eligible to receive raw color-format(rgba) based video streaming and display in the camera preview.

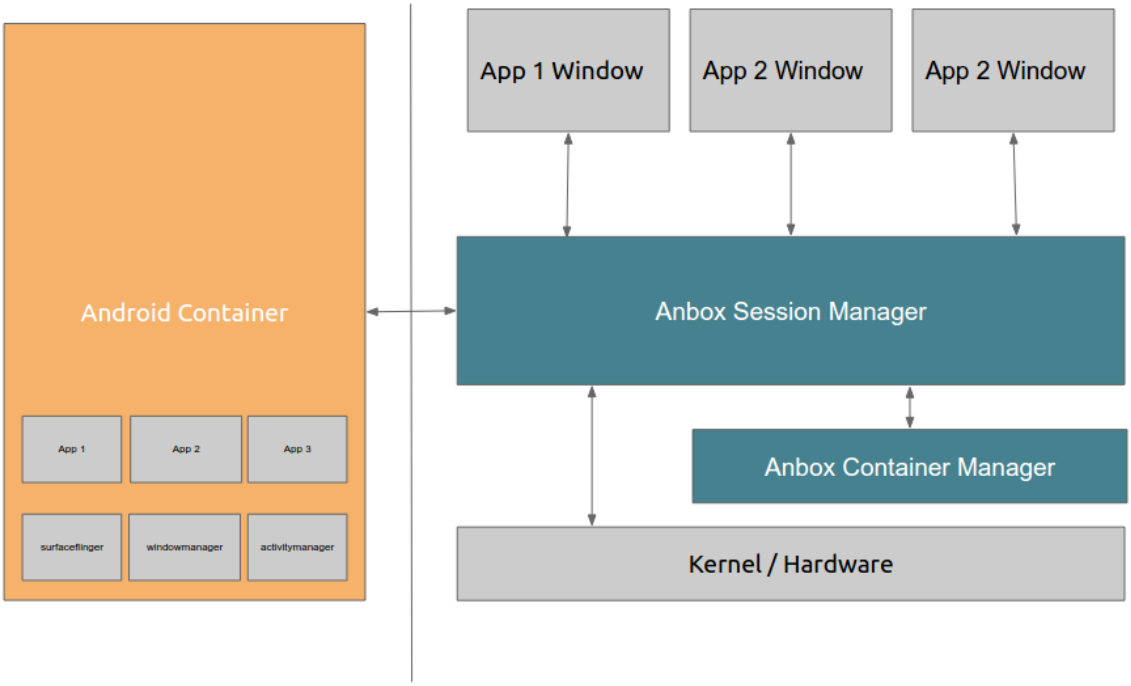
NOTE: the returned socket path is not fixed. It varies when you toggle camera support in Anbox via the above API

For example, you have a mp4 video file available in the container, to stream video content to Anbox:

Similar to uploading a static image to anbox, the video frame size must match the one of the resolution you got from the camera information API. For example, if you get 1280(w) x 720(h) resolution from the response of the camera info API, and the size of the video frame encoded in the uploaded video file is 320x640, you have to scale the video frame to the required size in some manners, otherwise you may get artifacts.

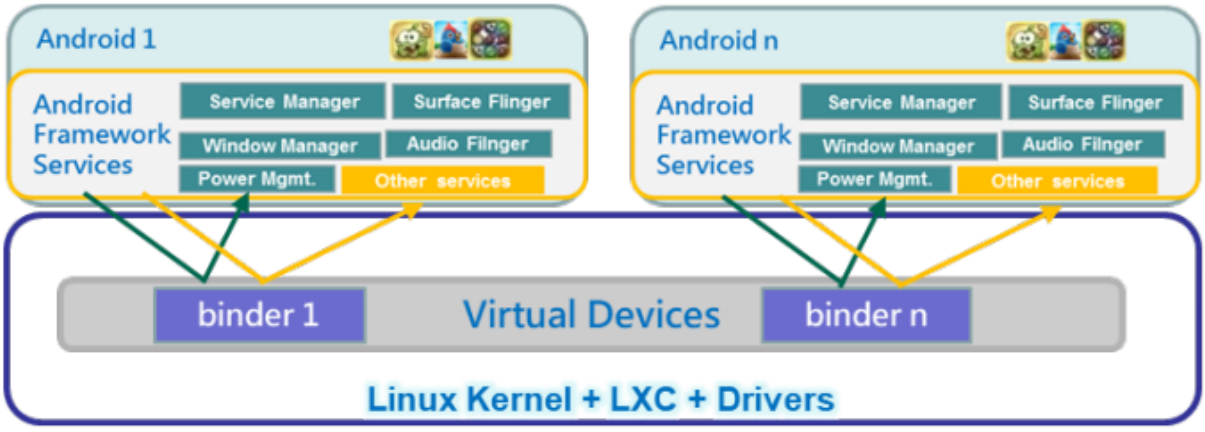
With ffmpeg, you can do:

Windows Manager

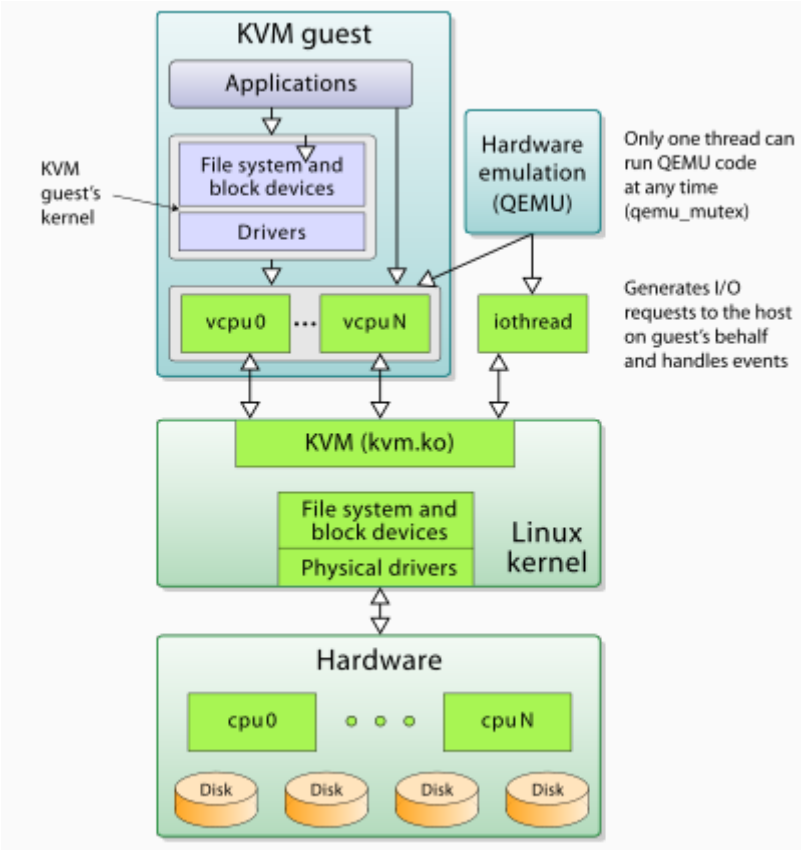


Binder 设备的虚拟与隔离

Linux kernel 的 binder driver 已支持多个 Android instrances。直接使用即可。



Android Qemu2



Kubernetes

常用 debug 命令

k8s 层面的：

kubectl run -i --tty busybox --image=busybox --restart=Never -- sh

Docker 层面的：

docker run -it busybox sh

k8s 报错日志查看

看系统日志 cat /var/log/messages
用 kubectl 查看日志
注意：使用 Kubelet describe 查看日志，一定要带上 命名空间，否则会报如下错误
[root@node2 ~]# kubectl describe pod coredns-6c65fc5cbb-8ntpv
Error from server (NotFound): pods "coredns-6c65fc5cbb-8ntpv" not found

kubectl describe pod kubernetes-dashboard-849cd79b75-s2snt --namespace kube-system kubectl logs -f pods/monitoring-influxdb-fc8f8d5cd-dbs7d -n kube-system

kubectl logs --tail200 -f kube-apiserver -n kube-system |more

kubectl logs --tail200 -f podname -n jenkins

用 journalctl 查看日志非常管用 journalctl -u kube-scheduler journalctl -xefu kubelet
journalctl -u kube-apiserver journalctl -u kubelet |tail journalctl -xe

用 docker 查看日志 docker logs c36c56e4cfa3 (容器 id)

K8s 网络模型

https://zhuanlan.zhihu.com/p/61677445

Flannel android 访问外网方法：

1, ifconfig eth0 10.244.2.111

ip route add default via 10.244.2.1

执行 ifconfig 后，用 ip route show table eth0 发现 eth0 的误导路由已被清空。执行 ip route add 后的 default 路由将被放到 table main 里，开始发挥作用。

2, 直接修改 eth0 talbe 里的路由：

ip route del default table eth0

ip route add default via 10.244.2.1 dev eth0 table eth0

以前 ping baidu 不通的原因可能是：eth0 table 里有个默认路由，被匹配上，但是又不转发到跳转 gateway 10.244.2.1,而是发到本地 eth0, 而本地 eth0 网络里是找不到 baidu/8.8.8.8 等地址的 ARP 信息的。必须发到跳转 gateway，才不会查 ARP 信息。

ip route show table

eth0

0.0.0.0 dev eth0 proto static scope link

default dev eth0 proto static scope link

10.244.2.0/24 dev eth0 proto static scope link

路由表按最具体到最不具体的顺序使用。

但是，在 Linux 上，它比您预期的要复杂一些。首先，存在多个路由表，并且何时使用哪个路由表取决于许多规则。

要获得完整图片：

$ ip rule show

0: from all lookup local

32766: from all lookup main

32767: from all lookup default

$ ip route show table local

broadcast 127.0.0.0 dev lo proto kernel scope link src 127.0.0.1

local 127.0.0.0/8 dev lo proto kernel scope host src 127.0.0.1

local 127.0.0.1 dev lo proto kernel scope host src 127.0.0.1

broadcast 127.255.255.255 dev lo proto kernel scope link src 127.0.0.1

broadcast 192.168.0.0 dev eth0 proto kernel scope link src 192.168.1.27

local 192.168.1.27 dev eth0 proto kernel scope host src 192.168.1.27

broadcast 192.168.1.255 dev eth0 proto kernel scope link src 192.168.1.27

$ ip route show table main

default via 192.168.1.254 dev eth0

192.168.0.0/23 dev eth0 proto kernel scope link src 192.168.1.27

$ ip route show table default

该 local 表是特殊的路由表，其中包含本地和广播地址的高优先级控制路由。

该 main 表是包含所有非策略路由的普通路由表。这也是您可以查看是否只是简单执行 ip route show（或简称为执行 ip ro）的表。我建议不再使用旧 route 命令，因为它仅显示 main 表格，并且其输出格式有些陈旧。

default 如果以前的默认规则未选择该数据包，则该表为空并且保留用于后处理。

模拟器检测手段

https://bbs.pediy.com/thread-264183.htm

https://www.zhihu.com/question/21355176/answer/114345450

https://www.cnblogs.com/Tesi1a/p/7624061.html

远程桌面

下面选用 XFCE + XRDP:

ubutnu 14.04 的远程桌面连接不支持 gnome 版本,从 13.03 之后都不支持了,所以需要更换为 xfce 界面.

sudo apt-get install tightvncserver xrdp

sudo apt-get install xubuntu-desktop

echo xfce4-session >~/.xsession

sudo vim /etc/xrdp/xrdp.ini

在文件末尾添加以下行：

exec startxfce4

修改 /etc/xrdp/startwm.sh 文件

sudo gedit /etc/xrdp/startwm.sh

在. /etc/X11/Xsession 上边添加 xfce4-session

```
#!/bin/sh

if [ -r /etc/default/locale ]; then
  . /etc/default/locale
  export LANG LANGUAGE
fi

xfce4-session
. /etc/X11/Xsession
```

修改配置文件 /etc/X11/Xsession

在文件最上面添加 xfce4-session

```
#!/bin/sh
#
# /etc/X11/Xsession
#
# global Xsession file -- used by display managers and xinit (startx)

# $Id: Xsession 967 2005-12-27 07:20:55Z dnusinow $
xfce4-session
set -e

PROGNAME=Xsession
```


重新启动 xrdp 服务

sudo service xrdp restart

XRDP 客户端用 remmina

"What is XDG_RUNTIME_DIR?", it is an environment variable that is set automatically when you log in. It tells any program you run where to find a user-specific directory in which it can store small temporary files.

echo $DISPLAY

And you can check what sudo thinks your DISPLAY is by typing:

sudo -s
echo $DISPLAY
exit

Again, let's check if it is properly set in sudo:

echo $XAUTHORITY
sudo -s
echo $XAUTHORITY
exit

If XAUTHORITY is pointing to a file in your home directory for you, but it's blank when you run sudo, then that's the problem.

FIX: Save the Environment Variables

So, what's the fix? If either the , you can tell sudo to preserve the environment by using the -E option, like so:

```
sudo -E evince
```

A better way: env_keep

You might well ask, Wait, if -E makes everything magically work, then why isn't it the default for sudo? The answer is that it is a potential security hazard. Environment variables affect the way programs work and you don't want them all being exported from a user account to the root. The "correct" way to do it is to add the line

```
Defaults env_keep += "DISPLAY XAUTHORITY"
```

to the sudoers file using visudo. You can check what environment variables sudo preserves by running:

```
sudo sudo -V
```

(Yes, you type sudo twice). I recommend putting the line not in the default sudoers file (/etc/sudoers), but in a local file that won't get overwritten when you upgrade your system. You can do that like so:

```
sudo visudo -f /etc/sudoers.d/local
```

But wait, what if none of the above works?

I think this is a fairly thorough answer, but if you're still having trouble, there's one other thing I'd suggest. You can use xhost to grant access to a specific user on the local host (your machine) like so,

```
xhost si:localuser:root
```

In this case, we're specifying root as the username, since that's the account that sudo runs programs as.

What is SPURV?

SPURV is our experimental containerized Android environment, and this is a quick overview of what it is.

It's aptly named after the first robotic fish since a common Android naming scheme is fish-themed names. Much like its spiritual ancestor Goldfish, the Android emulator.

Other Android Compatibility Layers

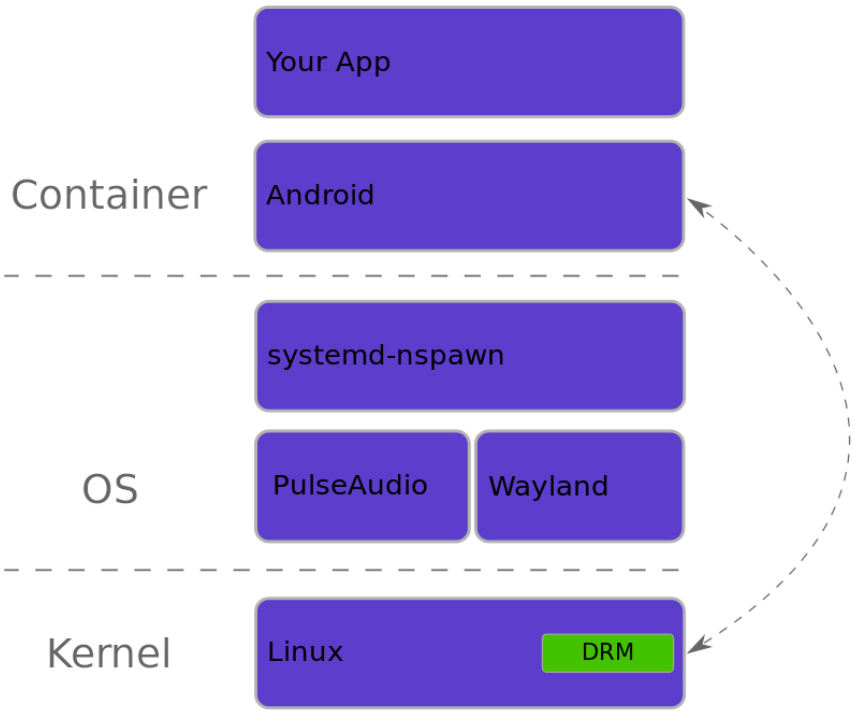
This means that Anbox which is LXC based, is different from SPURV in terms of how hardware is accessed. The hardware access that Anbox provides in indirect, and through the Qemu Pipes functionality, which is something it adopted from the Android (goldfish) emulator.

Shashlik and Genimobile are Android on Linux integration layers both based on Qemu, which means even better security properties than Anbox and certainly SPURV, but at the cost of an even larger performance penalty.

Direct Hardware Access

SPURV is different from other Linux desktop integrations for Android since it offers direct hardware access to the Android application. This is a choice we made for performance reasons. But has drawbacks, especially when it comes to security.

Using direct hardware access does however grant us increased GPU and CPU performance, which is important since we're targeting embedded platforms which can have very limited resources.



Components

SPURV consists of a few different parts, all living in the same project.

SPURV Audio

SPURV HWComposer