GPU 驱动的 Linux 桌面发行版适配

Icenowy Zheng

PLCT 实验室

2023-03-17

内容大纲

- 自底向上看 GPU 驱动
- 开源 GPU 驱动的组成
- 厂商 GPU 驱动适配问题

第一部分

自底向上看 GPU 驱动

内核驱动

“众所周知”

只有内核才有直接访问硬件的权限——所谓“用户空间驱动”的存在永远基于内核提供的框架。

- 显示管线的配置
- 显存管理
- GPU 任务的提交
- 提供额外特性的接口（如硬件编解码等）

API 实现库

3D 渲染	OpenGL ES/OpenGL/Vulkan
2D 渲染	OpenVG （及 3D 渲染库）
资源管理	GBM/EGL/GLX
硬件解码	VA-API/VDPAU/NV{ENC,DEC}
通用计算	OpenCL/Vulkan/CUDA

（可选）DDX （Device-Dependent X）

注意

DDX 仅在 XFree86 X 服务器下适用。

显示相关的 DDX 实现了 X11 中，与硬件相关的部分加速特性。

DDX 实现的特性举例

- DRI 认证/缓冲分配（对于 X 客户端调用图形加速所必须）
- Xv 叠加层、XRender 2D 加速等特异性加速

（可选）实用工具

八仙过海，各显神通。

举例

状态监测类 rocm-smi, nvidia-smi

性能分析类 rocprof, nvprof

第二部分

开源 GPU 驱动的组成

内核接口：FBDEV (Framebuffer Device)

注意：已废弃

内核 FBDEV 子系统已废弃，将不会接受新设备驱动支持；DRM 设备仍可提供 FBDEV 兼容设备节点。

真 · 亮机接口。
仅支持 2D 显示一个简单的 Framebuffer，没有任何现代化的加速功能。
现在仍旧提供的一大原因是 FBCON (Framebuffer Console)。

内核接口：DRM (Direct Rendering Manager)

最早为了 3D 渲染加速创造的接口；
后来随着 KMS (Kernel Mode Setting) 的引入，也开始能控制屏幕显示；
目前开源显示和 GPU 驱动的金标准。

不同驱动间的接口差异

DRM 驱动的 KMS 部分基本是标准化的（也允许在此基础上扩展）；但是显存管理、任务管理等均为驱动私有 ioctl 接口，由各个驱动自行定义。

API 实现库：Mesa

以一个软件的类 OpenGL® 实现开局；
后来获得了 DRI 的加成，能够使用图形硬件；
再后来有了基于可编程管线的 Gallium ，能够让不同驱动之间共享大量基础设施；
时至今日，支持的 API 不断扩张，支持了：

- GBM/GLX/EGL
- OpenGL (ES 和桌面)/Vulkan
- OpenCL
- VA-API/VDPAU
- 以及其他……

DDX：xf86-video-modesetting

“那个”通用的 DDX 。
现已成为 X.Org Server 代码库的一部分。
基于通用的 API 实现：

- DRM KMS 内核接口：实现显示管理、Vsync
- GBM：实现显存管理
- OpenGL (ES)：实现 2D 加速（含 2D 渲染加速和视频叠加）

其中基于 OpenGL (ES) 实现 X11 2D 加速的模块被称为 Glamor 。

DDX：其他

其他许多开源驱动同样也都有对应的 DDX：

- xf86-video-intel
- xf86-video-ati
- xf86-video-amdgpu
- xf86-video-nouveau

特异 DDX 的主要意义是 2D 加速。

第三部分

厂商 GPU 驱动适配问题

闭源内核驱动：也是一件精细活

众所周知

Linux 内核不维护内核内部 API 的稳定性。
内核会检查 .ko 文件中的版本信息及其暴露的 API，并会拒绝不与当前内核匹配的 .ko 文件。

发行闭源驱动的方式：

- 直接丢 .ko 出去——被三体人锁死到对应的内核二进制，需要发行版内核的源码，并与发行版发布内核的过程同步
- 将重要文件编译成 .o 发布——由于内核 API 不稳定，这些文件中对内核 API 的调用依然不保证跨内核版本可用
- 同上，但精心设计 .o 的导入/导出接口——这种情况能够构造出可以跨内核版本的闭源驱动

闭源内核驱动：许可证问题

众所周知

Linux 内核使用 GPLv2 许可证释出，有授权传递性问题。

虽然闭源驱动并未被 Linux 赶尽杀绝，但它们的生存环境依然十分艰难……

GPL 符号限制：EXPORT_SYMBOL_GPL

Linux 内核将大量接口以 EXPORT_SYMBOL_GPL 这个宏导出，而这样的接口将只能被在元数据中声明 GPL 兼容许可证的模块调用。与之相对的则是 EXPORT_SYMBOL 宏。

闭源内核驱动：接口限制来源

Linux 内核中的 GPL 符号限制源于 GPL 中 *derivative work* 的概念，在后文中，我们将被限制为 GPL 的符号称为“仅 GPL 符号”。

仅 GPL 符号一旦被一个模块调用，则该模块会被认为受到 GPL 的授权传递性影响，从而将受到 GPL 的约束，因此只有声明了 GPL 兼容许可证的模块才可以调用仅 GPL 符号。

一般来说，GPL 符号限制应用于 Linux 中一些较有特色的子系统（虽然此方面并无明确标准）。

闭源内核驱动：观赏厂商的表演

为了绕过 Linux 内核所做的限制，各种闭源 GPU 驱动厂商采用了一些计策，常见的有如：

计无付之 基于内核允许闭源模块调用的接口苦苦编写驱动

掩耳盗铃 在内核模块元数据中声明许可证为 GPL，即便给出的驱动包含闭源部分

画地为牢 将闭源部分作为一个单独的模块，编写额外的声明为 GPL 的内核模块作为包装，这一包装模块通过同时引用闭源模块和仅 GPL 符号完成驱动的工作

闭源内核驱动：绕过手段真的合理吗？

掩耳盗铃

当分发这一“声称为 GPL”的内核模块时，如何满足 GPL 对源码可用性的要求呢？

画地为牢

曾被某些厂商使用（现已转投下一页所讲方式实现需要 GPL 的功能）。

此方法已失效：自 Linux 5.9 起，如果一个 GPL 兼容的模块（“包装模块”）引用了非 GPL 兼容模块，则前者也将被视为非 GPL 兼容模块，从而无法使用仅 GPL 符号。

Greg Kroah-Hartman 锐评：

> *Ah, the proven-to-be-illegal "GPL Condom" defense :)*

内核驱动：开源与闭源的微妙平衡

嵌入式 GPU

嵌入式 GPU 的通常做法是将需要保密的逻辑编写为用户空间驱动，同时编写开源（并在芯片 BSP 中附带）的内核驱动配合闭源的用户空间驱动的工作。
ARM Mali GPU 同时在他们的开发者网站上公开提供未适配特定 SoC 的内核驱动以作为参考。

NVIDIA 的 open-gpu-kernel-modules

自 Turing (GeForce 系列对应 RTX20/GTX16) 起，NVIDIA 在 GPU 中嵌入了 GSP 处理器，从而能够将需要保密的逻辑移动到 GSP 固件中，而内核驱动本身则不必闭源。

内核驱动：警惕无序开发 /dev 的行为

另外值得注意的是，很多厂商驱动会在标准的 /dev/fbX 和 /dev/dri/ 之外，创建额外的私有设备节点。
请注意这些私有设备节点的权限，未正确配置可能导致只有 root 用户可以使用 GPU。
编写额外 Udev 规则可以配置这些设备节点的权限，一般来说归属于 root:video 且权限掩码为 660 。
若额外的设备节点接口设计不合理，可能成为无需提权即可利用的安全漏洞。

API 实现库：李代桃僵

部分厂商驱动直接提供了 lib{GL,EGL,GLESv2,gbm}.so 用于替换系统原有的 Mesa 库文件；例如 NVIDIA 的 340.xx 旧卡驱动。这种替换系统库文件的厂商驱动危害如下：

污染包管理器	若不通过包管理器安装，则有驱动被覆盖之虞；若通过包管理安装，则需要在打包时引入替换机制。
污染二进制文件	由于系统原有的动态库被覆盖，在链接可执行文件时，可能污染可执行文件的 ABI 。
无法共存	由于替换了库文件，多个这类驱动显然无法共存。

API 实现库：奇技淫巧之侵入 Mesa

因为历史原因，Mesa 内部也不是一块钢板——它内部有一个 DRI 接口，提供 *_dri.so 动态文件供 X 服务器实现 AIGLX 。
部分闭源驱动，为了规避替换 Mesa 的库文件的诸多弊端，选择提供一个接入 DRI 接口的库文件。
然而很不幸的是， DRI 接口从未获得妥善的文档；而且随着 Mesa 之外的 DRI 接口使用逐渐消亡，DRI 接口也逐渐转向 Mesa 内部接口，并被破坏性地简化。
也就是说，使用 DRI 接口是一种未定义的操作，且随着发行版对 Mesa 的更新，这一方法很可能于可预见的将来失效。

API 实现库：ICD (Installable Client Driver) 革命

首先由 OpenCL 扩展 cl_khr_icd 提出并定义，随后被 Vulkan 所继承；类似的机制之后还被 NVIDIA 实现于 GL(ES)，即 GLVND。

ICD 机制下，API 实现与接口得到了分离，应用程序只需链接到 ICD 加载器库，由后者负责在运行时选择并加载具体的驱动程序。

安装一个 ICD 驱动程序是很简捷的：提供驱动的动态库文件，并使用一个 json 配置文件记载这些动态库文件的路径和版本等信息，由此即可被 ICD 加载器库识别。

API 实现库：来自移动端的小小震撼

现如今，几乎所有 non-x86 SoC 的渲染/解码子系统和一些新兴厂商的 PCIe GPU ，均采用了某些在移动端更为流行的 GPU IP (Imagination PowerVR, ARM Mali, VeriSilicon Vivante 等)。
非常不幸地，由于这些 GPU IP 现阶段主要面向移动端（尤其是 Android ）市场，其 API 实现情况大大受移动端生态影响……

API 实现库：没有桌面 GL 的世界

移动端 GPU 驱动的通病

没有实现桌面版 GL。

此类情况下，通常需要发行版为仅使用 GLES 提供适配。
部分应用程序（如 KWin）和库（如 Qt）在编译时可以选择使用 GLES，而部分库（如 wxWidgets）并不具有可用的 GLES 实现。

GL4ES

GL4ES 是一个在 GLES 上部分实现桌面 GL API 的包装库；但是很不幸地，它的 Shader 翻译器过于幼稚，以至于大量应用程序无法运行/需要特别适配……

API 实现库：Zink，是救赎还是画饼？

Vulkan，作为一种理论上比 GL 更接近底层的图形 API，可以在其上实现 GL ——事实上 Mesa 已经提供了一个在 Vulkan 基础上实现 GL API 的驱动，也就是 Zink 。
非常不幸的是，移动端 GPU 驱动的 Vulkan 实现通常不好，在桌面 Linux 上表现尤其差……

某模拟器开发团队锐评

It is obvious that only 4 vendors have the expertise and the commitment to make Vulkan drivers work: NVIDIA, AMD, and Mesa, with a special mention for Intel, who recently stepped up their game.

API 实现库：同样的痛苦，但这次是编解码

Android 采用的编解码 API 是 OpenMAX （以下简称 OMX ），很不幸地，这一 API 在 Linux 桌面下应用甚少——Linux 桌面下主要使用的是 VA-API/VDPAU 。

GStreamer 可以提供对 OMX 的支持，但是很不幸地，大多数多媒体应用软件并不支持 OMX 。

更加糟糕的消息是，并没有可用的 API 转换层能够基于 OMX 驱动提供 VA-API/VDPAU……

DDX：快跑！

TL;DR

为了 Wayland 化的将来考虑，理论上，新的 GPU 驱动不应当使用 DDX 。

DDX 仅是 XFree86 X 服务器的一部分；也就是说，非 X11 的场景下 DDX 起不到丝毫作用。尽管 Wayland 为了兼容 X11 应用程序，提供了 Xwayland 服务器，但后者同样不支持 DDX ，而选择直接调用 Glamor 完成 2D 加速。

DDX：没有了会怎么样？

但是很不幸地，当下 xf86-video-modesetting / Glamor 亦可能遇到各种问题：

- 已在 X 服务器 master 得到修复的问题（尚未发版）：modesetting DDX 缺乏双缓冲支持，在性能不足时可能出现撕裂；Glamor 对 GLES 的支持有问题，会导致颜色不正确（蓝色变成黄色）。
- Glamor 原理导致的问题： Glamor 大量依赖于 GL 中的 FBO（Frame Buffer Object，帧缓冲区对象），以至于显示缓冲区本身都是用 EGLImage 导入并创建为 FBO 的。这对驱动程序的 FBO 处理提出了很高的要求，在不完善的驱动程序上很可能导致性能问题乃至渲染正确性问题。

DDX：ABI 版本号

对于以闭源形式提供 DDX 模块的驱动来说，还有一个额外的噩耗：
DDX 作为 XFree86 X 服务器的内部细节，并不保证 X 服务器大版本之间的接口稳定性——事实上当下 master 分支的 X 服务器已经具有了和 21.1 分支不同的 ABI 版本号 （26 vs 25）。

第四部分

结语与展望

驱动适配总结

内核驱动	两条路：封装核心代码为 .o 文件/全部开源（可以带有闭源固件）
用户空间驱动	感恩的心，感谢 ICD 机制，GL 驱动是时候适配 GLVND 了；纯 GLES 驱动仍然是修罗场
DDX	不该有，但现阶段很多时候又不得不有的东西

一言蔽之：摸着 NVIDIA 过河。

展望与许愿

能支持开源驱动的话，那是最棒的了！不过，如果条件不允许的话：

- 至少开源内核驱动
- 至少给图形库适配 GLVND 并适配桌面 GL（或妥善适配 Vulkan ）
- 至少做好 Glamor 优化

鸣谢

PLCT 实验室	提供图形栈相关工作机会，允许我参与 RevyOS GPU 适配开发。
@Rongronggg9	试讲伙伴，协助检查并清理演示文稿。
freedesktop.org	长期维护 Mesa、libglvnd 等基础库。