deepin/UOS 操作系统架构之图形栈

徐小东

统信软件

2023-03-15

目录

1. 图形栈概述
2. 内核层
3. 用户层
4. 参考资料

Section 1

图形栈概述

图形栈整体架构

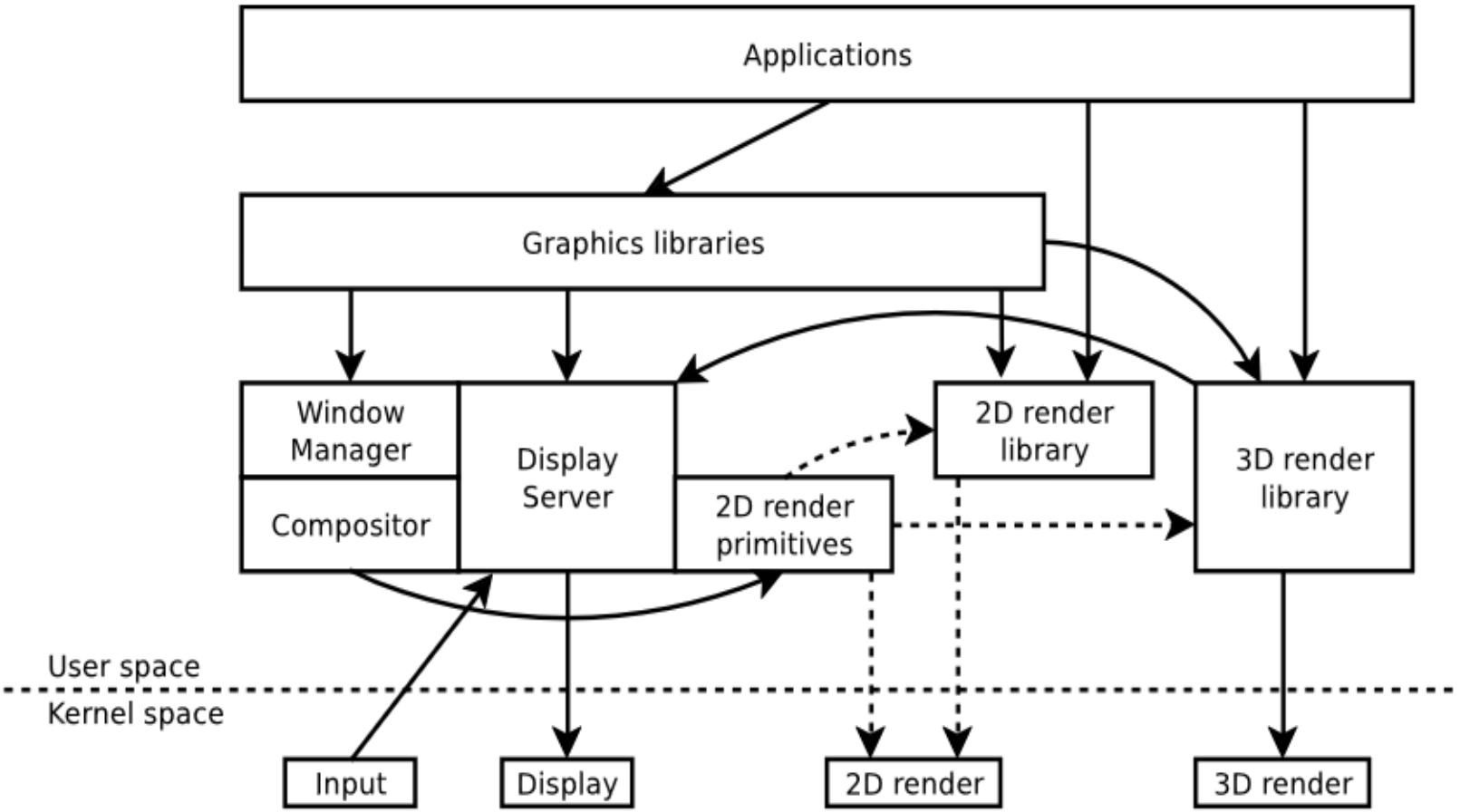


图 1: 图形栈整体架构

Linux 内核

- 输入子系统
  - 支持鼠标、键盘、触摸屏等设备
  - 统一的 evdev 事件接口
- 帧缓冲设备（fbdev）子系统
  - 用于在屏幕上显示像素缓冲的遗留接口
  - 不支持热插拔
  - 通过特定的驱动接口添加扩展特性
- 直接渲染管理器（DRM）子系统
  - 统一的显示配置接口：内核模式设置（KMS）
  - 允许一起同步更改（DRM atomic）
  - 通过特定的驱动接口暴露渲染设备（DRM render）
  - 提供内存管理机制（DRM GEM）

底层用户库

- 输入
  - libevdev (C)：evdev 系统调用接口的包装
  - libinput (C)：输入设备管理，使用 libevdev
- 显示/渲染
  - libdrm (C)：DRM 系统调用的包装
- 2D 渲染
  - Pixman (C)：优化的像素级别操作
  - Cairo (C)：优化的矢量图形绘制
- 3D 渲染
  - Mesa 3D (C)：OpenGL 的开源实现

X Window

- 概述
  - X Window/X11 是显示协议
  - 通过许多协议扩展来扩展功能
  - X.org 是 X11 服务器参考实现
  - 需要外部的窗口管理器来处理多个应用程序
  - 由服务器或窗口管理器（利用合成扩展）来合成
- 窗口管理器
  - KWin：源于 KDE 桌面环境
  - Compiz：支持 3D 合成
- 显示客户端库
  - Xlib (C)：遗留的 X11 客户端协议库
  - XCB (C)：更新的 X11 客户端协议库

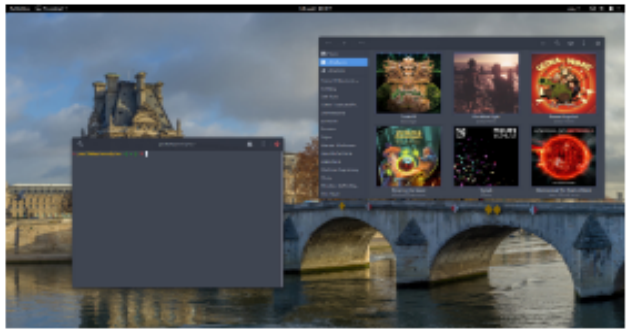
Wayland

- 概述
  - Wayland 是一个显示协议（包括 core 和 extensions）
  - 使用更现代和简约的范式替换 X11
  - 合成器（服务端）处理输入、窗口、合成及显示
- Wayland 合成器
  - 利用 libwayland-server 基础协议库
  - Weston/libweston：参考实现
  - Sway/wlroots：平铺式窗口管理器及基础库
  - KWin：KDE 合成器
- 显示客户端库
  - 利用 libwayland-client 基础协议库

GNOME-Shell (green) displaying the top-bar and background

Lollipop (green) with window decorations (red)

GNOME-Terminal (green) with window decorations (red)



The composited result

图 2: 合成过程

上层图形化库及桌面环境

- 应用程序很少直接使用 X11 或 Wayland
- 绘制并管理用户界面十分复杂
- 广泛使用的上层图形化库
  - GTK (C)：基于 Widget 的 UI 库
  - Qt（C++）：基于 Widget 的 UI 库
  - DTK（C++）：基于 Qt5 开发的一整套 UI 图形库
- 桌面环境组合相关的库和组件，具有一致的外观
- 桌面环境
  - DDE：使用 DTK
  - GNOME：使用 GTK
  - KDE：使用 Qt

Section 2

内核层

帧缓冲设备 - fbdev 概述

- fbdev 是历史遗留的 Linux 显示子系统
- 包含许多显示相关的特性
  - 访问帧缓冲
  - 原子操作
  - 绘制光标
  - 电源管理
- 初始状态可通过命令行的 `video` 选项配置
- 通过 `/dev/fb*` 设备文件用于用户空间
- 内核用 fbcon 来提供图形化控制台

帧缓冲设备 - fbdev 限制

- 针对简单的使用场景设计
- 输出设置主要是静态的
- 不支持缓冲分配和管理
- 主要采用于 1990 ~ 2000 年间

DRI

- Direct Rendering Infrastructure，直接渲染基础架构
- 在 X Window 下直接访问图形化硬件的框架
- 组成
  - 内核：DRM
  - X 服务器
  - 客户端库
- 主要用于为 Mesa 3D 创建快速的 OpenGL 实现以提供硬件加速

DRM

- Direct Rendering Manager，直接渲染管理器
- Linux 内核中负责显卡 GPU 的接口子系统
- 用户空间程序用于：
  - 使用它向 GPU 发送命令和数据，并执行操作
  - 使用 DRM API 命令 GPU 进行 3D 硬件加速渲染
  - 视频解码
  - GPGPU 计算

DRM 架构

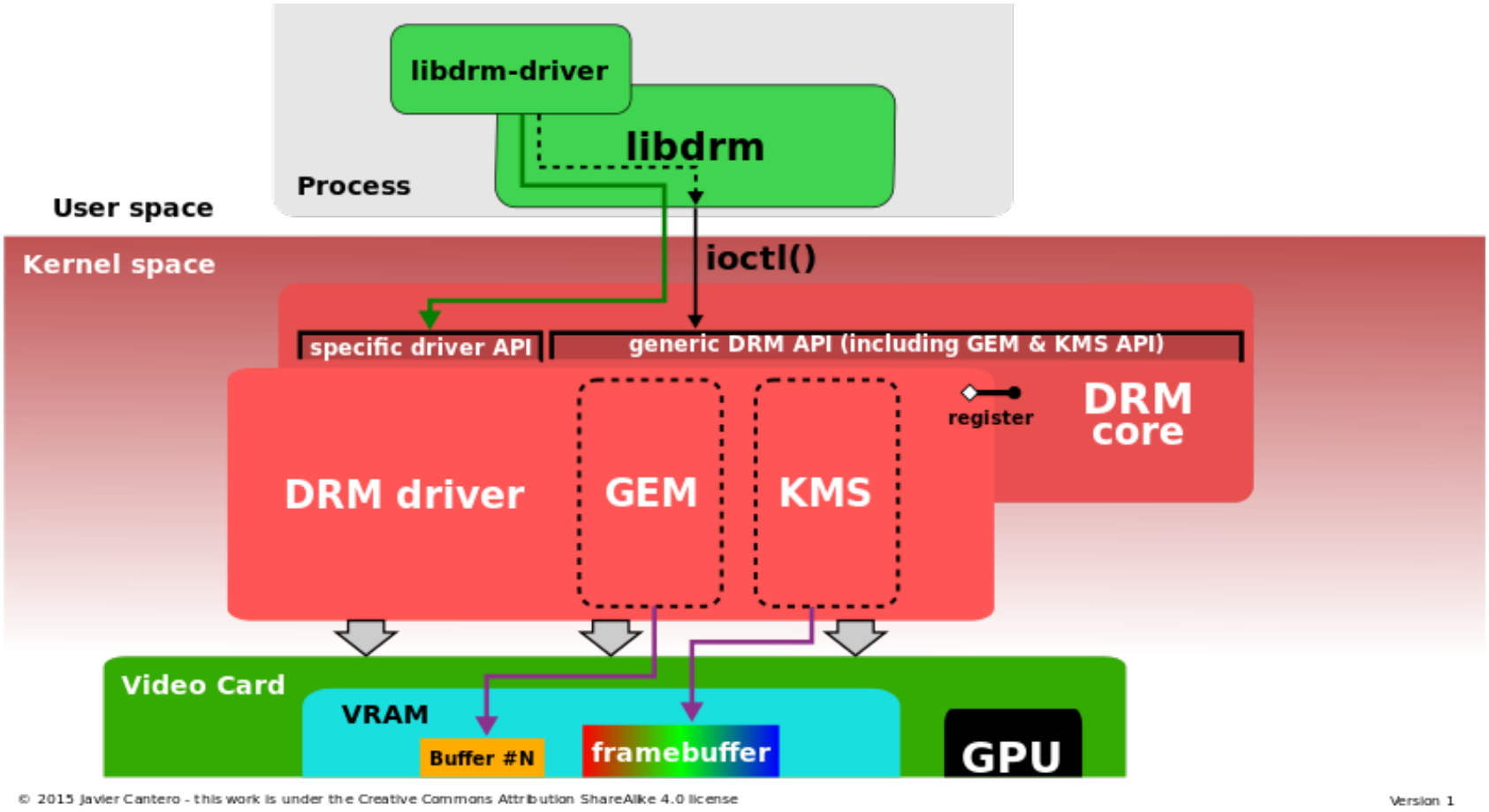


图 3: DRM 架构

- DRM 核心：提供基础框架，硬件无关的功能，如：注册不同的 DRM 驱动
- DRM 驱动：支持的各种 GPU 硬件类型
  - GEM：Graphics Execution Manager，提供内存管理
  - KMS：Kernel Mode Setting，设置显示模式（屏幕分辨率、颜色深度和刷新率）

DRM 设备文件

- 用主/次编号标识的 Unix 风格设备文件
- 存在两种类型的 DRM 设备
  - 主要节点：`/dev/dri/card*`，用于 KMS 接口的显示操作
  - 渲染节点：`/dev/dri/renderD*`，用于特定驱动接口的渲染操作
- DRM 设备也可直接由内核使用
  - fbdev 兼容层提供 `/dev/fb*` 节点
  - 由 fbcon 用来提供虚拟控制台
- 用户空间打开设备节点需要权限
  - 一般允许 `video` 组

Section 3

用户层

libdrm

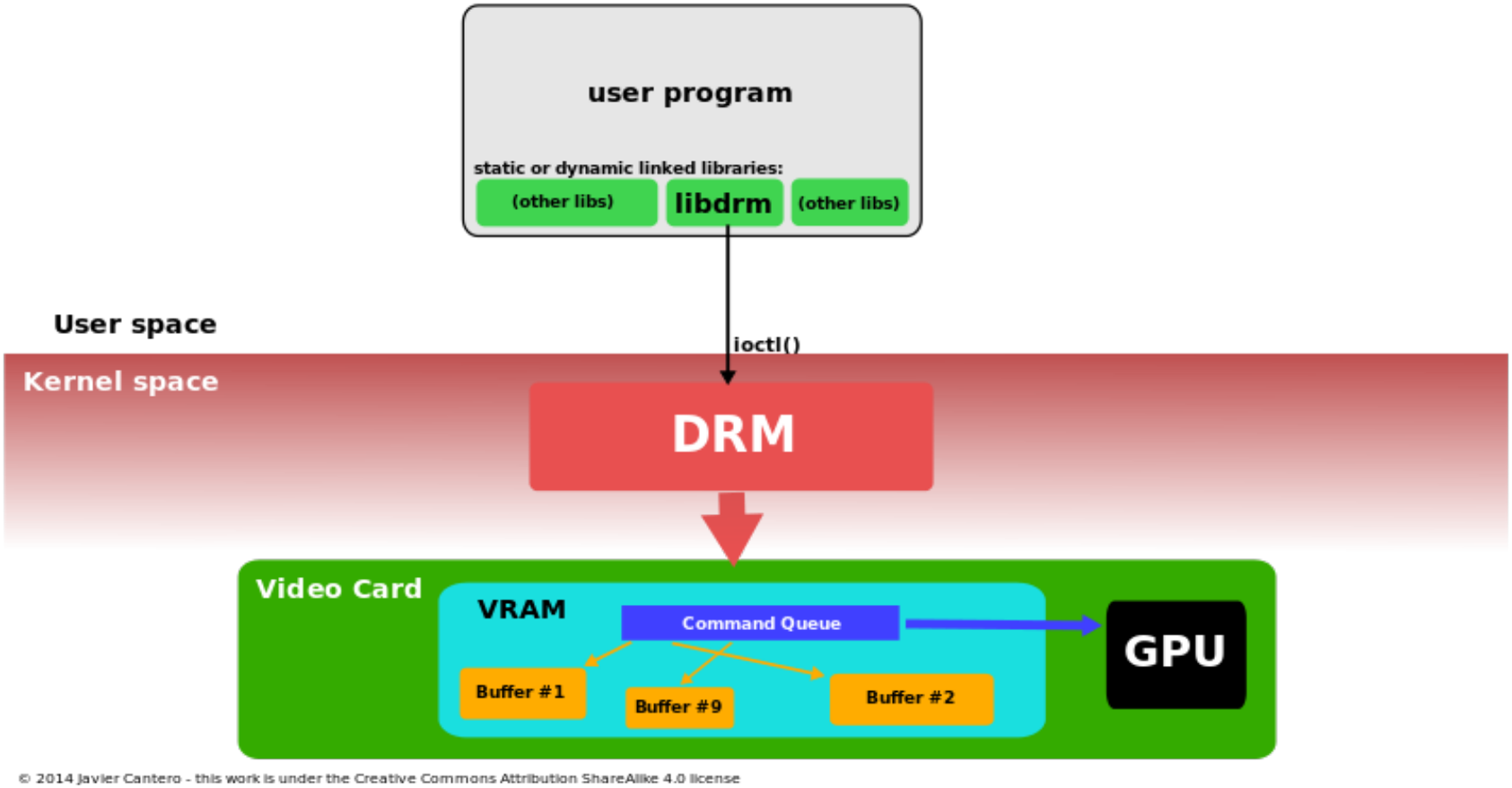


图 4: 进程使用 DRM

- 使用 libdrm 从用户空间访问 DRM 设备
- 通过 `ioctl` 暴露便利的接口、辅助函数及数据结构
  - 对于 KMS 支持：`libdrm.so` 库
  - 对于特定的硬件渲染驱动：专门的库，如：`libdrm_nouveau.so`
- 用于与 DRM 交互的用户空间项目：
  - Xorg
  - Weston
  - Mesa 3D

Mesa 3D - 标准的 3D 渲染 API：OpenGL

- 3D 渲染 API，针对 GPU 硬件加速设计
  - 通用 API，但特定于硬件实现
  - 由 Silicon Graphics 于 1992 年启动，现由 Khronos Group 管理
- OpenGL 提供 3D 图形的高级方法
- Stateful 及 context-based 的编程模型
- OpenGL 随附 GL Shading Language (GLSL)

Mesa 3D - 标准的 3D 渲染 API：OpenGL ES 和 EGL

- OpenGL ES 作为嵌入式设备的简化版而引入
- 使用 GLSL shader 及与 OpenGL 相同的编程模型
- EGL 作为标准化的窗口集成 API 而引入
  - 使用原生的系统显示服务器连接
  - 替换 X11 的 GLX，并由 Wayland 所采用

Mesa 3D - 标准的 3D 渲染 API：Vulkan

- Vulkan 是针对 GPU 访问的低级通用 API
- 适合 3D 渲染和计算
- 使用 Standard Portable Intermediate Representation (SPIR-V) shader
- Direct 编程模型，具有低级内存管理
- API 提供 window system integration (WSI)

Mesa 3D 概述

- Mesa 是自由软件的 3D 图形实现
- 实现支持渲染 API
  - OpenGL (4.6) 和 OpenGL ES (3.2)
  - Vulkan (1.1)
  - Direct 3D (version 9)
- 实现窗口系统集成
  - EGL：Wayland、X11、Android
  - Vulkan WSI：Wayland、X11
  - GLX：X11
- 支持其他 GPU 相关特性
  - 视频解码加速：VDPAU、VAAPI、OMX
  - 计算（GPGPU）：OpenCL

Mesa 3D 架构

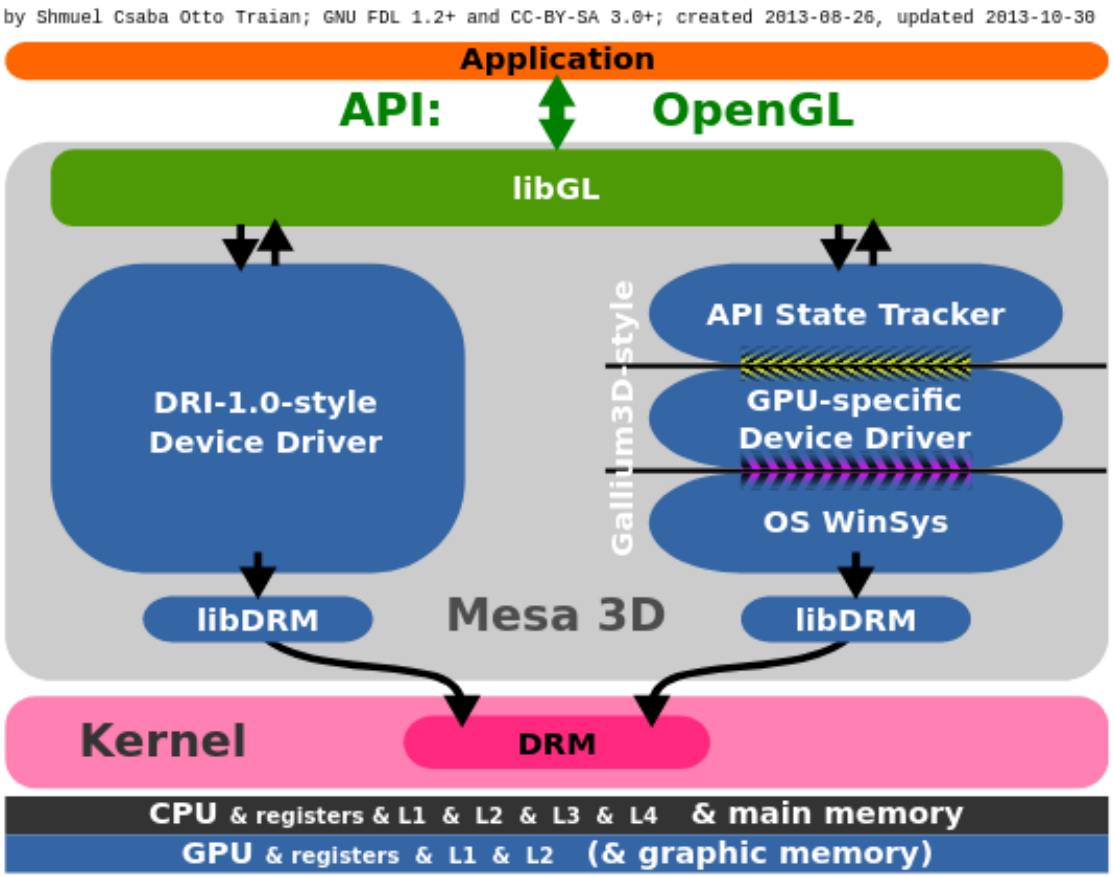


图 5: Mesa 3D 架构

- 传统的 mesa 驱动有明显的代码重复
  - API 状态跟踪
  - 编译器实现
- Gallium 3D 接口
  - API 状态跟踪：维护 API 的使用状态
  - Drivers：实现着色器编译及硬件配置
  - Winsys：实现低级内核接口

Mesa 3D 硬件支持现状

- Intel HD/Iris Graphics
  - 平台：Intel
  - Mesa 驱动：i965 (classic), iris (Gallium)
  - DRM 驱动：i915
- Nvidia pre-NV110
  - 平台：Tegra
  - Mesa 驱动：nouveau (Gallium)
  - DRM 驱动：nouveau
- AMD Radeon GCN-ish
  - 平台：AMD
  - Mesa 驱动：deonsi (Gallium)
  - DRM 驱动：amdgpu

X Window - X11 协议

- X11 核心协议由 Xorg 实现
  - 基于 packet 的异步系统：Request、Reply、Event 等
  - 可用于本机（UNIX 套接字）或通过网络（TCP/IP）
- 将 drawables 暴露给客户端以便传输和绘制像素数据到服务器
  - Windows：由应用程序所有的显示缓冲区
  - Pixmaps：可复制到 windows 的非显示后备存储
- Windows 表示为树状结构
  - 由 X 创建根窗口开始
  - 上层窗口和子窗口由客户端创建
- graphics context (GC) 允许请求基本的绘制和字体渲染
- 服务器提供 input events 给相关的客户端
  - 相对于窗口坐标的鼠标移动
  - 从原始的键码翻译按键符号

X Window - X11 协议扩展

- X11 通过对主协议扩展进化至今
  - 适用于客户端的额外接口，以匹配新的硬件特性
- XKB：复杂的键盘布局
- Xinput2：触摸板、触摸屏、多点触控支持
- XSHM：共享客户端/服务器内存，避免额外的传输/拷贝
- XRandR：显示器配置及热插拔不必重启服务器
- Composite：将窗口合成委派给合成窗口管理器
- XRender：2D 渲染 API，包含 alpha 合成、光栅化、变换、滤镜等
- Xv：视频输出格式转换

X Window - Xorg 架构

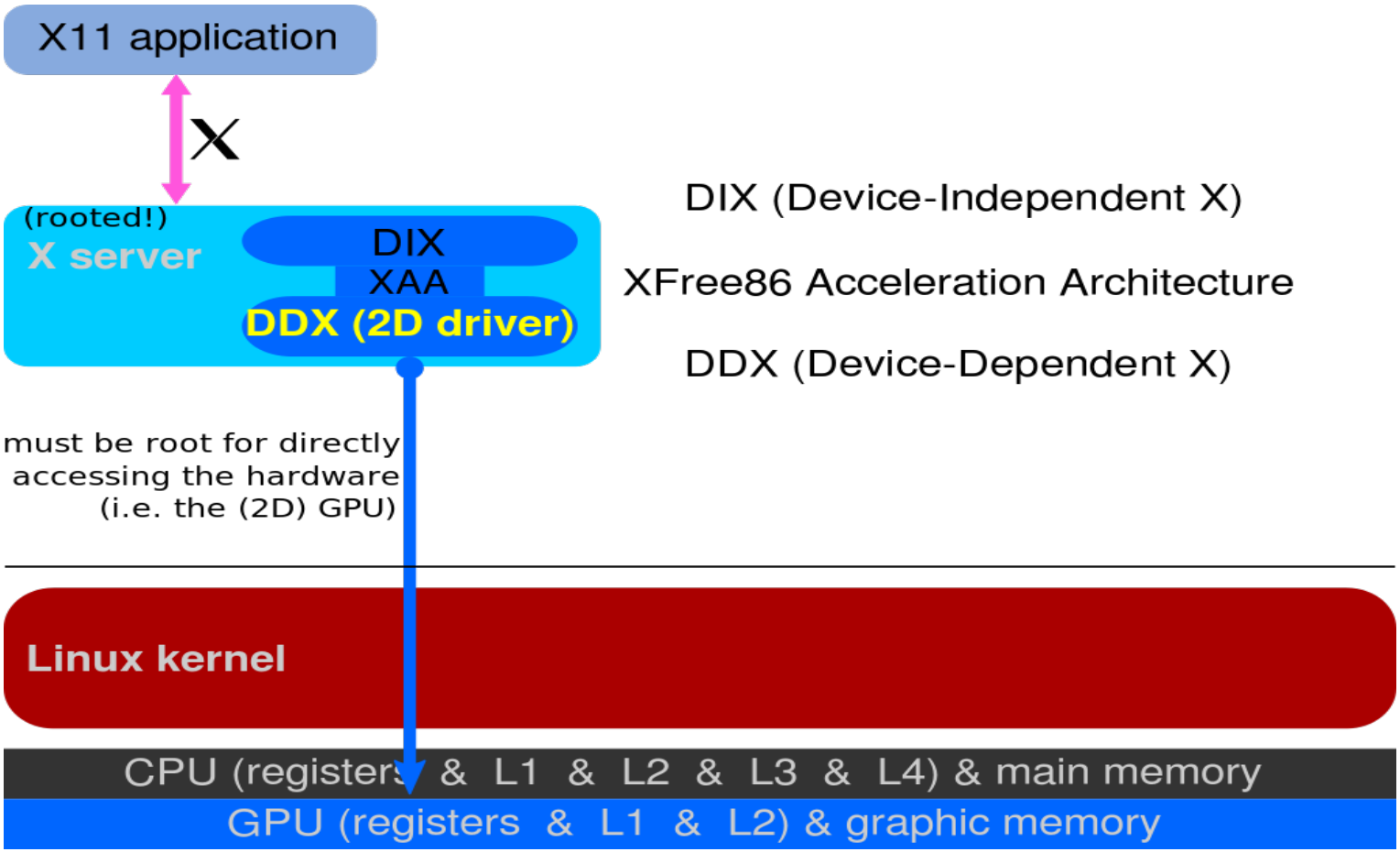


图 6: Xorg 架构

- Xorg 分为通用和特定硬件两部分
- Device-Independent-X (DIX)
  - X11 协议实现、客户端协作
  - 主事件循环及事件分发
  - 图形操作逻辑、样板及回调实现
- Device-Dependent-X (DDX)
  - 输入驱动（`xf86-input-*`）从内核抓取事件
  - 视频驱动（`xf86-video-*`）提供模式设置及 2D 加速
- EXA 在 DIX 和 DDX 之间提供 2D 加速架构
- Glamor 利用 OpenGL 3D 渲染为 DDX 提供 2D 加速

X Window - Xorg 驱动概述

- 通用的 Xorg 输入驱动
  - xf86-input-libinput 使用 libinput 来获取输入事件
  - xf86-input-evdev 直接使用 evdev 内核接口（已废弃）
- 特定的 Xorg 输入驱动
  - xf86-input-synaptics 用于笔记本触摸板
  - xf86-input-wacom 用于 Wacom 绘图板
- 通用的 Xorg 显示驱动
  - xf86-video-modesetting 用于 DRM KMS，可用 glamor 加速
  - xf86-video-fbdev 用于 fbdev 接口，无加速（遗留）
  - xf86-video-vesa 用于 Video BIOS Extension (VBE) 帧缓冲（x86）
- 特定的 Xorg 显示驱动
  - xf86-video-[intel,nouveau,amdgpu]

X Window - Xorg 架构：输入到显示往返

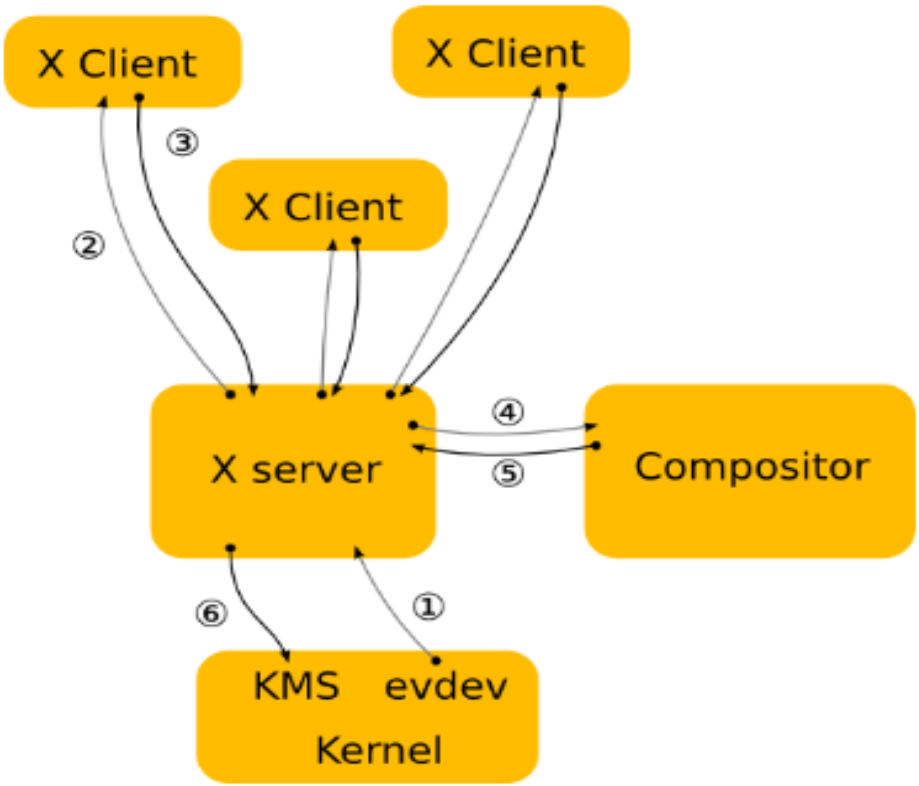


图 7: Xorg 架构

1. 服务器从内核读取输入事件
2. 判定受影响的客户端，并接收事件
3. 客户端更改并执行渲染请求
4. 服务器执行渲染并通知合成器
5. 合成器使用后备缓冲更新区域
6. 服务器从合成器缓冲更新显示缓冲

X Window - X11 的主要问题

- X11 核心协议及范式导致各种问题
  - 基于缓冲拷贝、传输及频繁重绘
  - 缺乏同步/反馈接口
  - 重度基于 packet 的协议导致延迟问题
- 因为核心协议不演进，导致扩展激增

Wayland 概述

- 作为 X Window system 的现代替代于 2008 年启动
- 简化了技术栈
  - 服务器和合成器统一为同一个组件
  - 客户端做所有渲染
  - 客户端和服务器之间共享缓冲，无传输
  - 窗口装饰可由客户端或服务器添加
- 改进了安全
  - 隔离每个客户端的输入和输出
  - 只有合成器能访问缓冲
  - 避免将合成器运行在 root 下

Wayland 协议

- Wayland 提供 client-server API
  - Wayland 通过本机 IPC 进行显示连接
  - 显示通过 WAYLAND_DISPLAY 环境变量标识
  - 异步及面向对象协议
  - 对象表示来自合成器的资源
  - 对象实现特定的接口，包含 Requests（由客户端发送）、Events（从服务器接收）
- 协议在 libwayland-client 和 libwayland-server 中实现

Wayland 扩展协议

- 合成器能暴露扩展协议（对象接口）
  - 协议（包括核心）以 XML 文件描述
  - `wayland-scanner` 工具产生客户端及服务器 C 代码及头文件
- 广泛使用的协议扩展
  - XDG-Shell：桌面 shell 集成

Wayland 架构

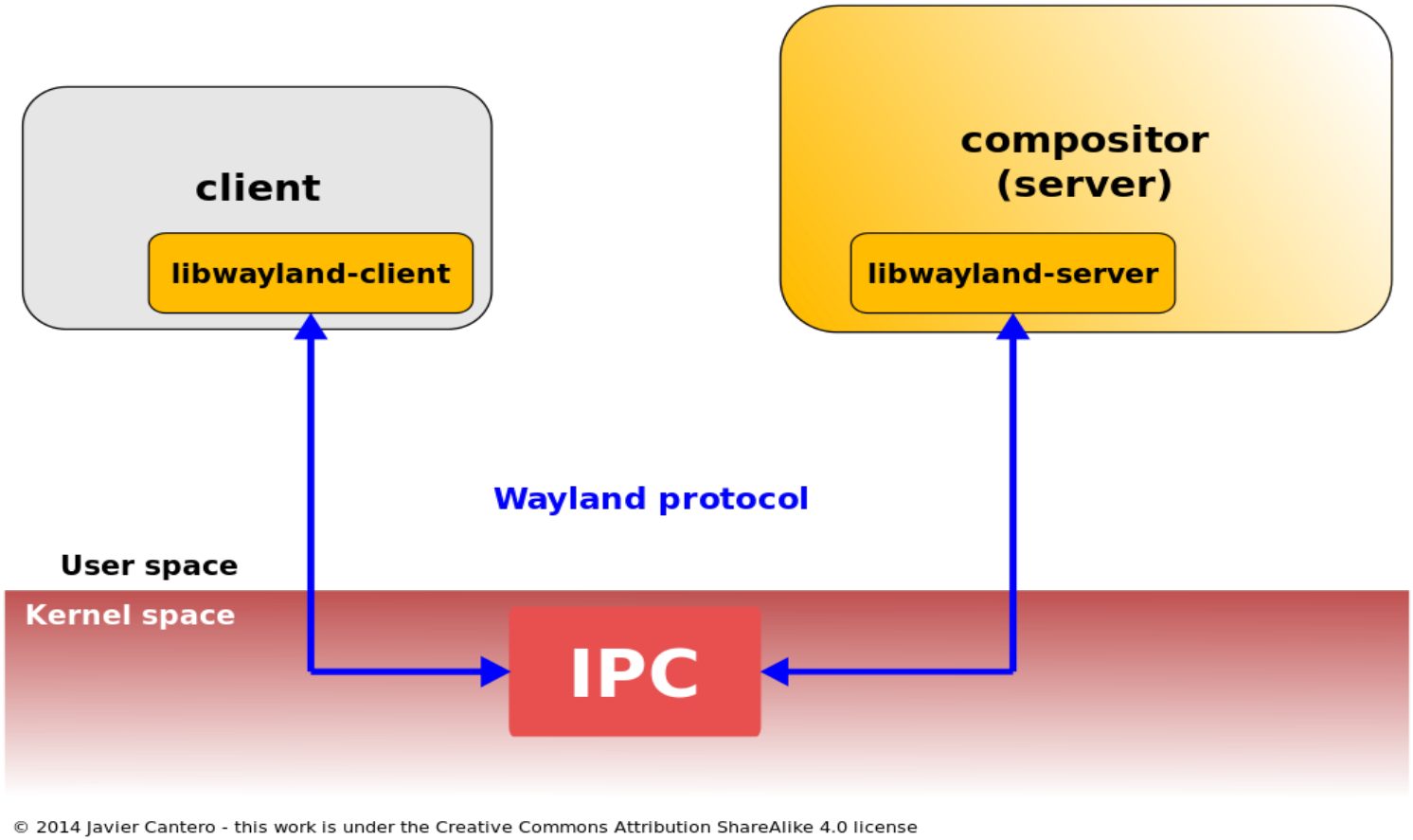


图 8: Wayland 架构

- 遵循 client-server 模型
- 客户端：请求将像素缓冲显示到屏幕上的图形化应用程序
- 服务端（合成器）：提供服务来控制像素缓冲的显示
- 两层协议
  - 底层：处理客户端和合成器进程间通信
  - 上层：处理客户端和合成器需要交换的信息

Wayland 架构：输入到显示往返

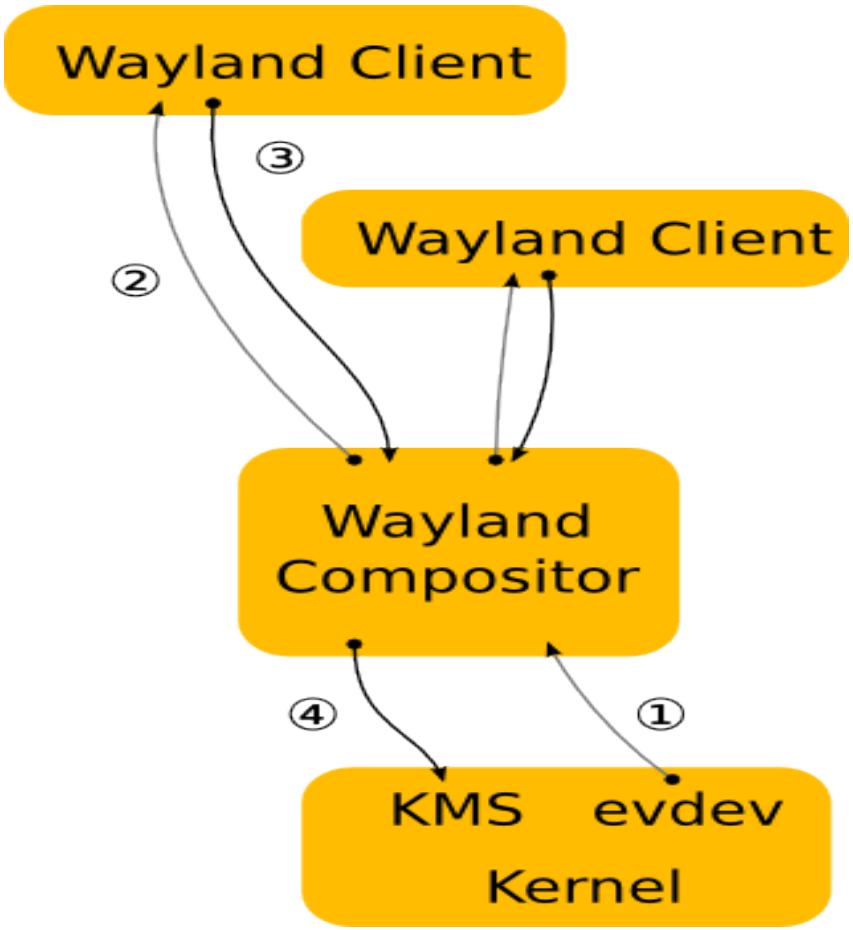


图 9: Wayland 架构

1. Linux 内核的 evdev 模块获取事件并将其发送给 Wayland 合成器
2. Wayland 合成器通过查询场景图来判定该由哪个窗口所属的客户端接收事件
3. 当 Wayland 客户端收到事件时，通过 EGL 进行直接渲染，并向合成器发送请求表明区域已得到更新
4. Wayland 合成器从客户端收集请求，然后重新合成屏幕

Wayland 现状

- Wayland 现在十分成熟、健壮、高效及使用广泛
  - 大多数界面库都支持：GTK 3、Qt 5、SDL、EFL
  - 主要的桌面环境支持：GNOME 3、KDE Plasma 5
  - 登陆管理器集成会话：`/usr/share/wayland-sessions`
  - 使用用户权限运行：`systemd-logind`
- X11 应用程序可用 XWayland 集成
  - X 服务器实现注册为 Wayland 客户端
  - Wayland 合成器实际作为 X 合成窗口管理器
- 各种合成器实现
  - 绑定到桌面环境：mutter、kde
  - libinput 创建来用于输入
  - libweston 来自 Weston 合成器核心
  - wlroots 来自 Sway 合成器核心
  - 支持 DRM KMS 显示、EGL 渲染

Section 4

参考资料

参考资料

- Wikipedia: Direct Rendering Infrastructure
- Wikipedia: Direct Rendering Manager
- Wikipedia: X Window System
- Wikipedia: X.Org Server
- Wikipedia: Wayland
- Wikipedia: Mesa