The Linux Graphics Stack

Or Why X11 is Like That

Simon Kadesh

February 9, 2024

RITLUG

Introduction

- You are all looking at a graphical application

Introduction

- You are all looking at a graphical application
- How does it get displayed?

Introduction

- You are all looking at a graphical application
- How does it get displayed?
- Why is it so terrible?

Pre-history

The Teletype

- In the beginning, there was the teleprinter
- Electromechanical device for sending textual messages

The Teletype

- In the beginning, there was the teleprinter
- Electromechanical device for sending textual messages
- Then computers got invented – teleprinters turned out to be a convenient interface for them

The Glass Teletype

- Printers are kind of terrible:

The Glass Teletype

- Printers are kind of terrible:
  - They're slow

The Glass Teletype

- Printers are kind of terrible:
  - They're slow
  - They jam

The Glass Teletype

- Printers are kind of terrible:
  - They're slow
  - They jam
  - Paper costs money

The Glass Teletype

- Printers are kind of terrible:
  - They're slow
  - They jam
  - Paper costs money
- What if we could display text... on a screen

The Terminal

- Display-based terminals are great – they solve all kinds of problems with the teletype as a user interface

The Terminal

- Display-based terminals are great – they solve all kinds of problems with the teletype as a user interface
- They also open up room for new inventions

The Terminal

- Display-based terminals are great – they solve all kinds of problems with the teletype as a user interface
- They also open up room for new inventions
    - Movable cursors

The Terminal

- Display-based terminals are great – they solve all kinds of problems with the teletype as a user interface
- They also open up room for new inventions
  - Movable cursors
  - Deleting

The Terminal

- Display-based terminals are great – they solve all kinds of problems with the teletype as a user interface
- They also open up room for new inventions
  - Movable cursors
  - Deleting
  - Dinging

The Terminal

- Display-based terminals are great – they solve all kinds of problems with the teletype as a user interface
- They also open up room for new inventions
  - Movable cursors
  - Deleting
  - Dinging
- With a nice enough terminal, working with a computer is almost not terrible

The Virtual Terminal

- Computers are getting much more powerful

The Virtual Terminal

- Computers are getting much more powerful
    - We have the 386

The Virtual Terminal

- Computers are getting much more powerful
  - We have the 386
  - We have VGA

The Virtual Terminal

- Computers are getting much more powerful
  - We have the 386
  - We have VGA
  - We have hardware text mode

The Virtual Terminal

- Computers are getting much more powerful
  - We have the 386
  - We have VGA
  - We have hardware text mode
- How will we take advantage of it?

The Virtual Terminal

- Computers are getting much more powerful
  - We have the 386
  - We have VGA
  - We have hardware text mode
- How will we take advantage of it?
- We can simulate a terminal!
- Better yet, we can simulate lots of terminals

The Framebuffer Device

- Problem: Not all systems have a hardware text mode

The Framebuffer Device

- Problem: Not all systems have a hardware text mode
- Solution: Make a software text mode – thus emerged the framebuffer (`fbcon`)

The Framebuffer Device

- Problem: Not all systems have a hardware text mode
- Solution: Make a software text mode – thus emerged the framebuffer (`fbcon`)
- We can also use the framebuffer for other things, like drawing images, or displaying user interfaces
- Generally this is a bad idea, as the framebuffer does not support hardware acceleration

The Point

- There are a few problems with the interfaces described thus far:
  - They're all either hardware or in-kernel
  - Most of them only display text
  - The one option with graphics is slow
- We can do better

The X Window System

The X Window System

- In the year 1984, MIT researchers are working on "Project Athena" to improve the available computer tools on campus
- This project ends up spawning the X Window System
- The people at MIT and DEC found that X was better than existing graphical interfaces in many ways
  1. X was hardware independent
  2. X was vendor independent
  3. X was network transparent
  4. X was fast

X11

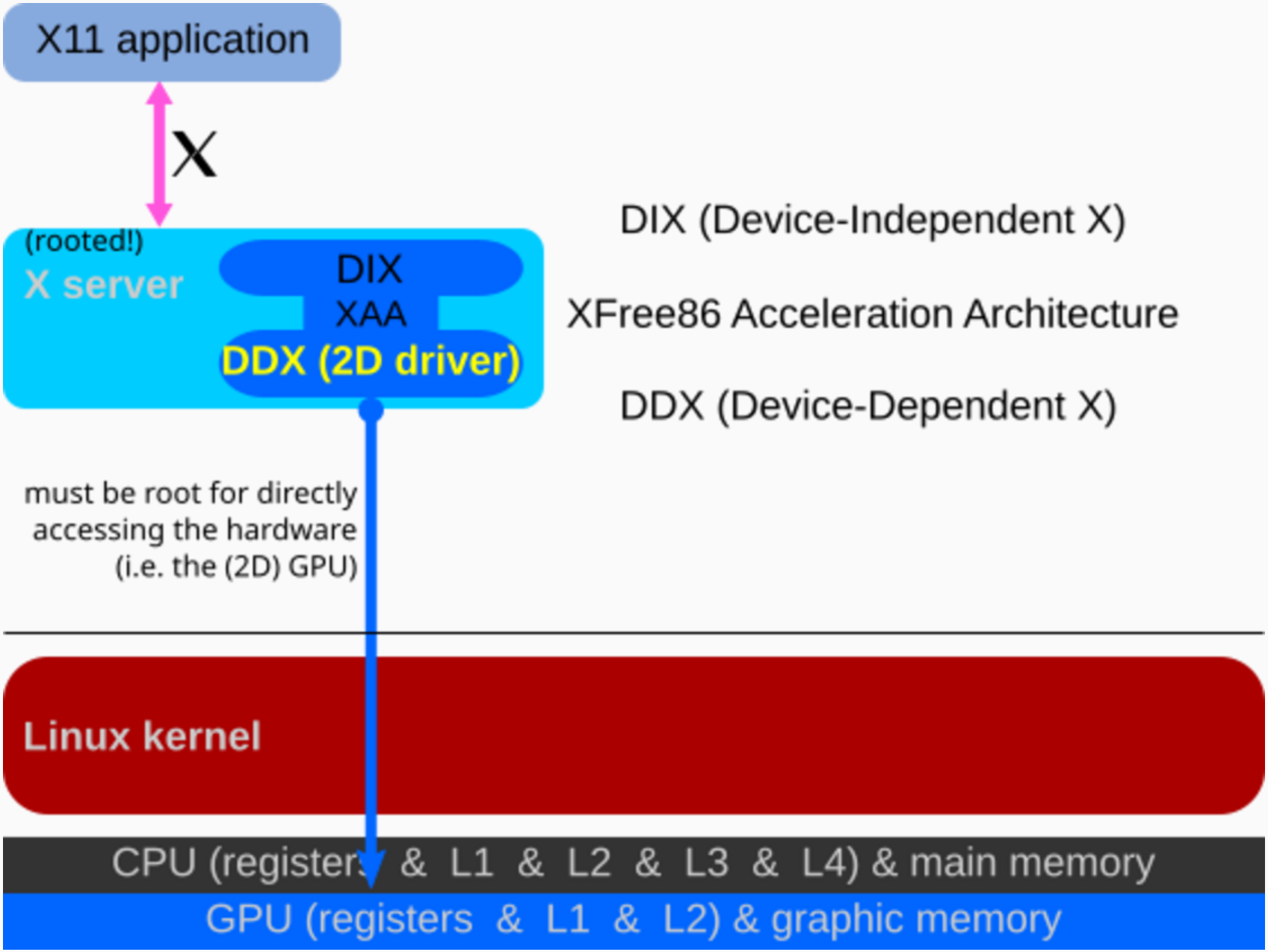
- Protocol update over X10 intended to increase hardware independence
- Early X11 implementations were hardware terminals (e.g., VT1000)
- X11R5 included an implementation of the X11 server, called X386, for PC compatibles

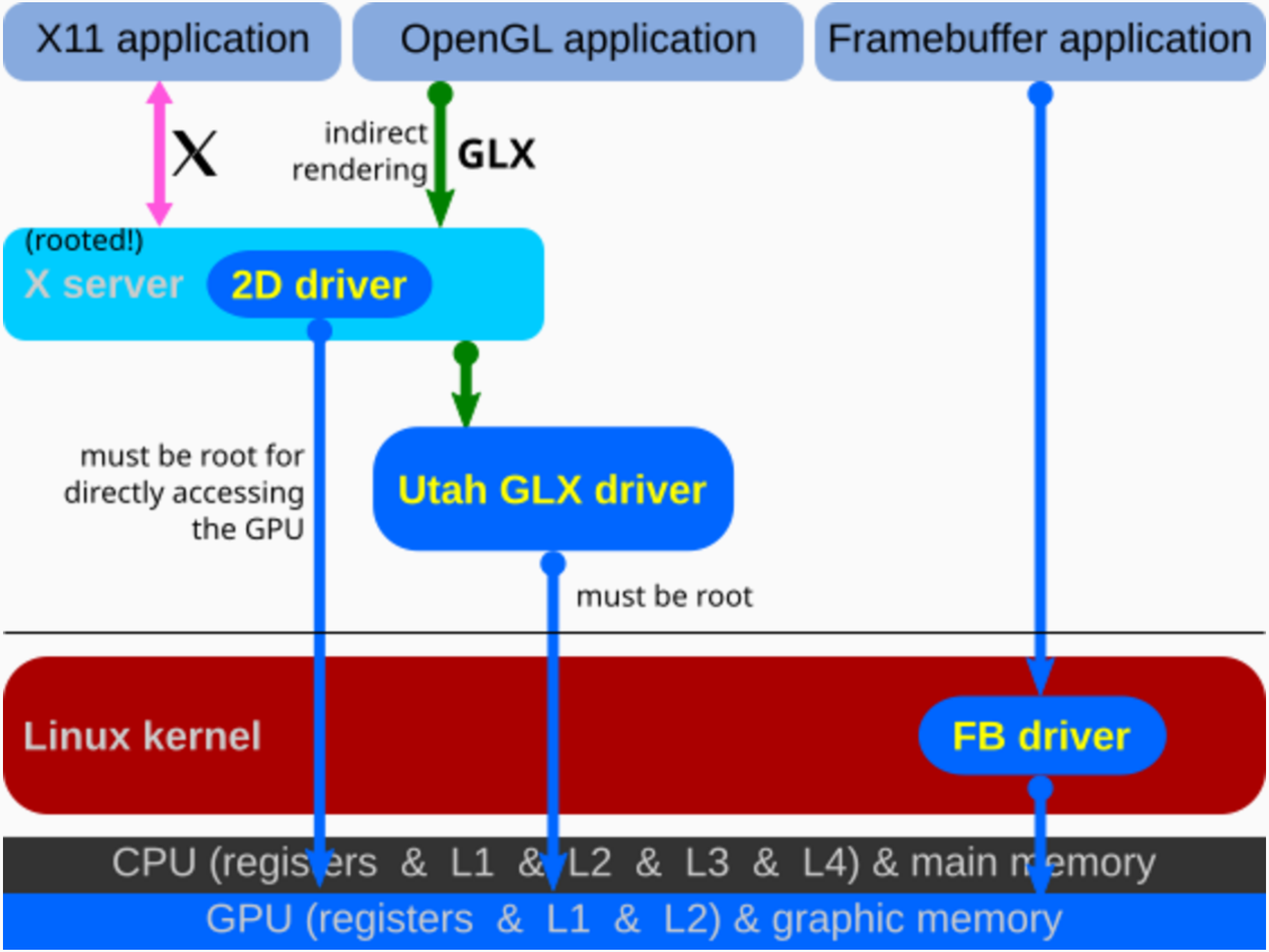
X386 & XFree86

- X386 was somewhat buggy
- Furthermore, the X Consortium stops existing in 1995, and The Open Group isn't doing much to develop X
- XFree86 introduces a number of useful features
- In particular, XFree86 splits the X server into a main server which loads drivers, rather than a monolith for each possible driver

Why everything is so terrible

- The goal of X was to decouple client software from the hardware on which it runs
- Initially this was achieved by having a different X server for each possible hardware
- XFree86 changed this, by making one server that could load multiple drivers
  - Consequentially, graphics drivers exist within the X server
  - Leads the ongoing problem of having the graphics subsystem tightly coupled to the X server


X11 application
X
(rooted!)
X server
DIX
XAA
DDX (2D driver)
DIX (Device-Independent X)
XFree86 Acceleration Architecture
DDX (Device-Dependent X)
must be root for directly accessing the hardware (i.e. the (2D) GPU)
Linux kernel
CPU (registers & L1 & L2 & L3 & L4) & main memory
GPU (registers & L1 & L2) & graphic memory


X11 application
OpenGL application
Framebuffer application
X
indirect rendering
GLX
(rooted!)
X server
2D driver
must be root for directly accessing the GPU
Utah GLX driver
must be root
Linux kernel
FB driver
CPU (registers & L1 & L2 & L3 & L4) & main memory
GPU (registers & L1 & L2) & graphic memory

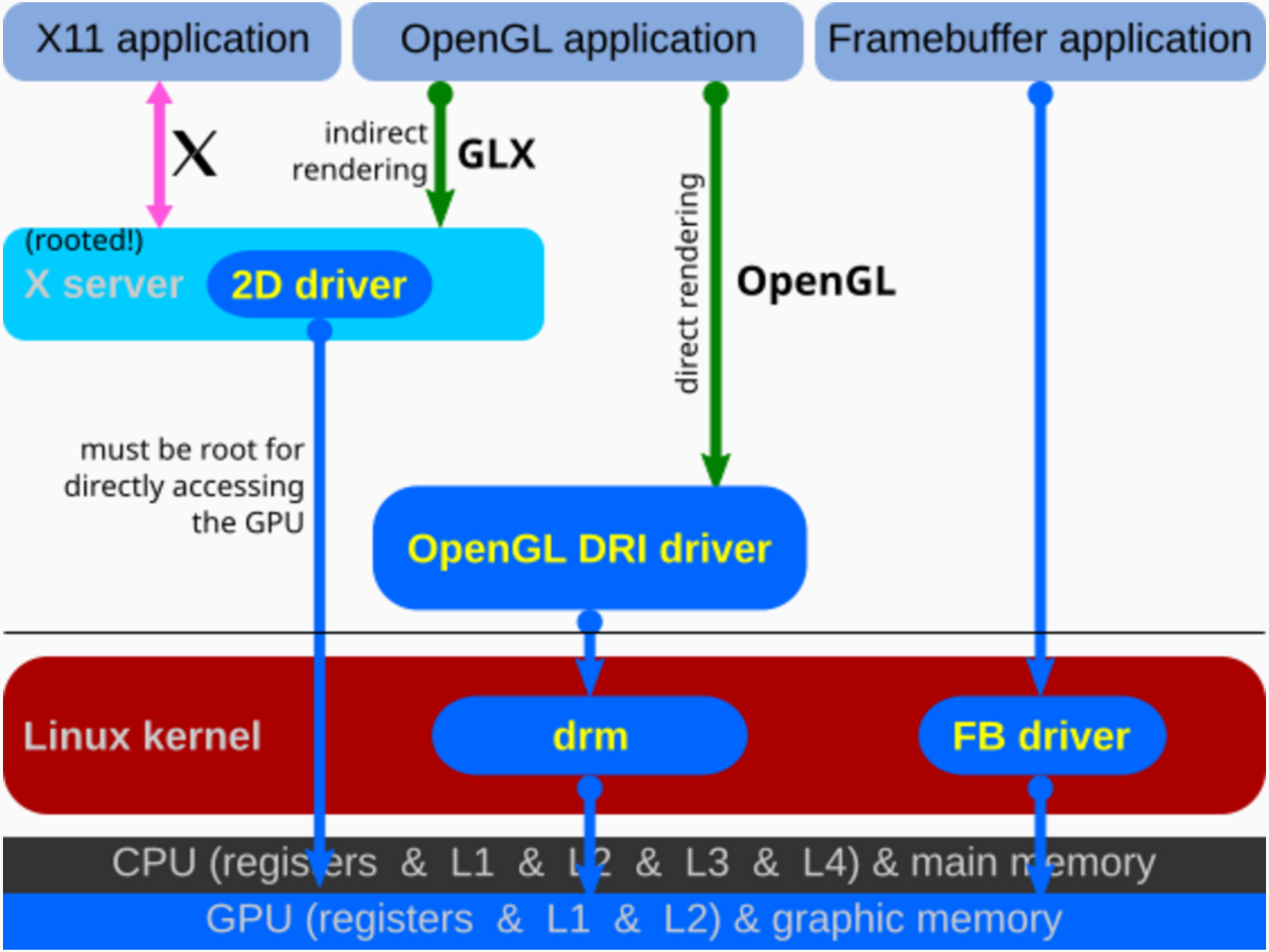
Modernity

Mesa

- Open source implementation of graphics APIs such as OpenGL, and Vulkan
- Translates to hardware specific operations on the physical GPU

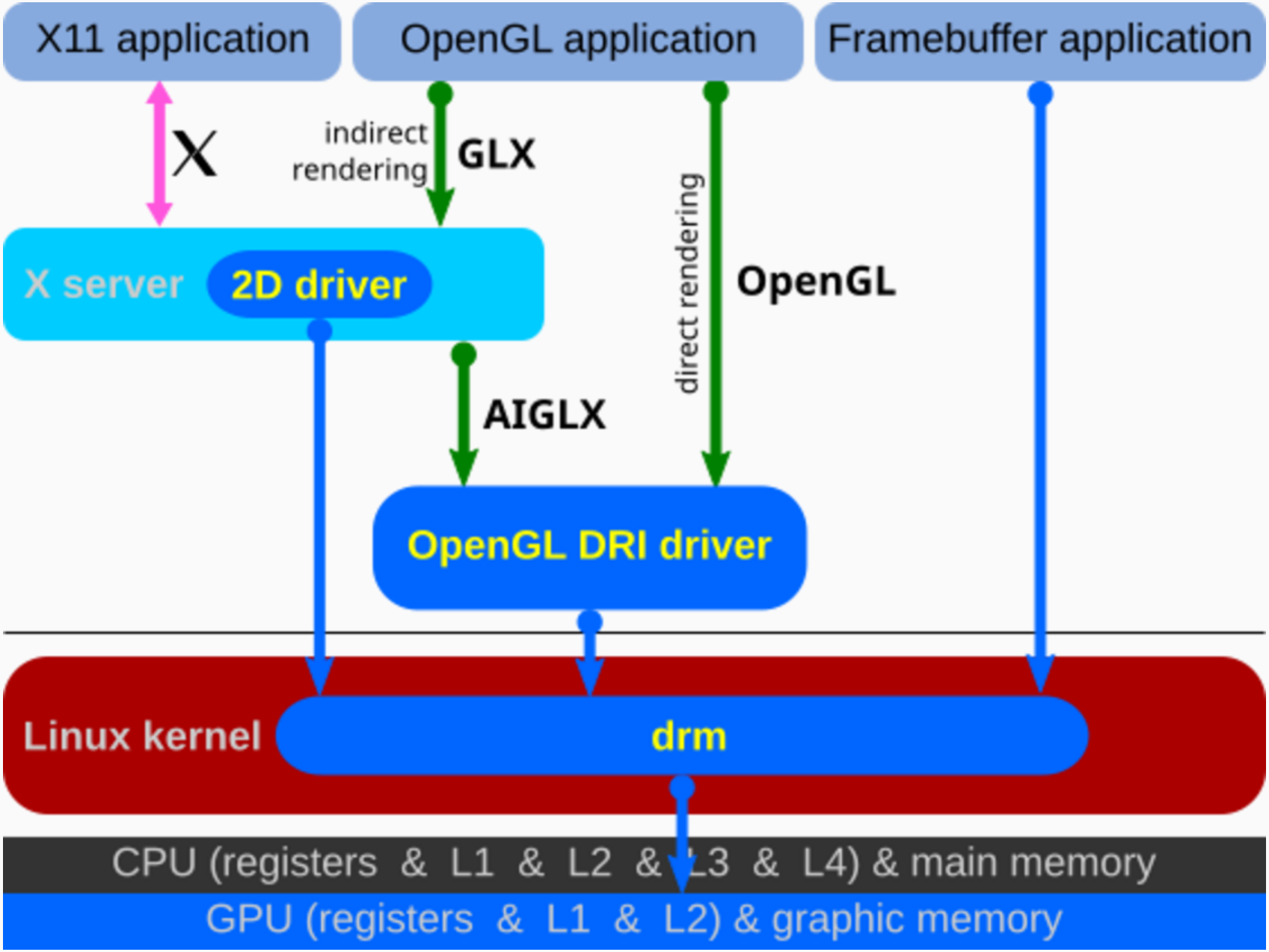
DRI

- Device Dependent X is kind of terrible
  - Graphics drivers in the X server
  - Only the X server can do rendering or control the display
  - Everything else needs to ask X as an intermediary
- DRI was designed to solve both problems – by putting graphics drivers in kernel, we reduce coupling, and allow the kernel to multiplex access to many different programs at the same time.
- Unfortunately DRI fails to completely remove the scourge of X.org interdependency – it's still ultimately an X driver


X11 application
OpenGL application
Framebuffer application
X
indirect rendering
GLX
direct rendering
OpenGL
(rooted!)
X server
2D driver
must be root for directly accessing the GPU
OpenGL DRI driver
Linux kernel
drm
FB driver
CPU (registers & L1 & L2 & L3 & L4) & main memory
GPU (registers & L1 & L2) & graphic memory

DRM/KMS

- DRM is the part of the DRI project that manages rendering
  - Give individual programs direct OpenGL contexts, rather than rendering through the X server
- There's still a problem: We still depend on X to set things up and control the display
- This is where KMS comes in – we move the modesetting into the kernel


X11 application
OpenGL application
Framebuffer application
X
indirect rendering
GLX
direct rendering
OpenGL
X server
2D driver
AIGLX
OpenGL DRI driver
Linux kernel
drm
CPU (registers & L1 & L2 & L3 & L4) & main memory
GPU (registers & L1 & L2) & graphic memory