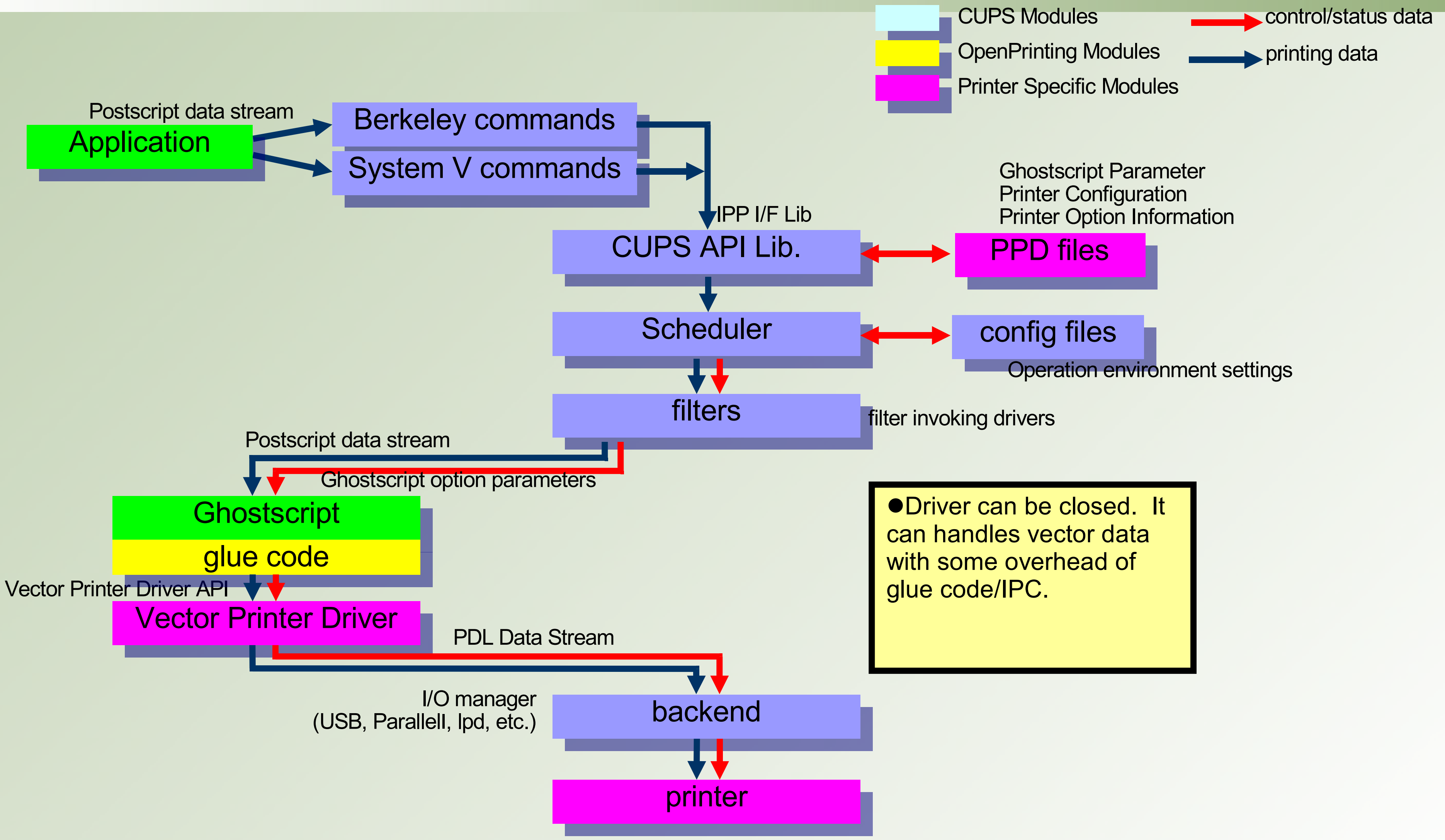
FSG OpenPrinting: Vector Printer Driver Standard

Free Standards Group
OpenPrinting Workgroup Japan/Asia
Osamu Mihara
<mihara.osamu@fxpsc.co.jp>
October 24, 2006

Vector Printer Driver API

- Send graphics commands to printer, instead of rasterized bitmap image.
- Called by render engine such as Ghostscript or X print server.
- Objectives of OpenPrinting Vector Printer Driver API
  - Performance Optimization
    - Achieve full speed printing on fast laser printers
    - Utilizes graphical acceleration feature supported by printer controllers
  - Data Size Optimization
    - Reduces size of print data using high level graphics commands.
    - Contributes to reduce network bandwidth and increase through-put
  - Print Quality Optimization
    - Utilizes printer's graphics quality enhancement technology by sending vector graphics command
    - Color Optimization
      - Driver can recognize the kind of graphics primitives and switch color scheme – natural color for bitmaps and vivid colors for graphics and text.
  - Independent Design from Rendering Engine
  - Free from Free Software License Woe
    - Vendor drivers can be provided without making source code open

Ghostscript+OpenPrinting Vector Printer Driver



●Driver can be closed. It can handles vector data with some overhead of glue code/IPC.

API Overview

API Overview

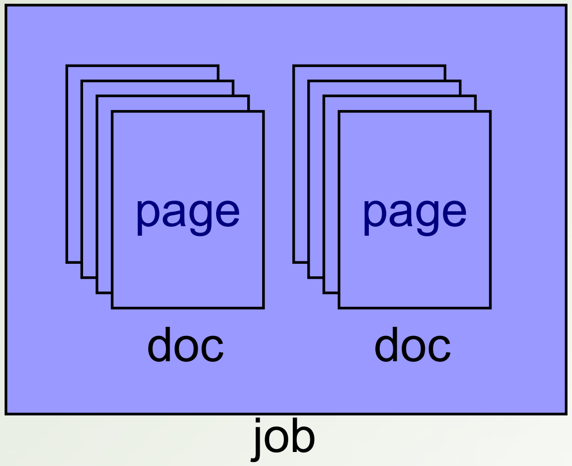
- Job Control
  - Open/Close driver
  - Set Job/Document/Page attributes
- Graphics State Operation
  - Set attributes for each graphics objects
- Drawing Operations
  - Path
  - Text
  - Bitmap Image
  - Scanline
  - Raster Image
- Stream Data (embedded PDL)

Printer Context Operations

- OpenPrinter()
  - Create printer context
  - Register API entry pointers
  - Specify file descriptor for data stream
- ClosePrinter()
  - Closes printer context
  - Driver releases all resources

Job Control Operations

- A print job consist of documents.
- A document consist of pages (document is optional unit).
- StartJob(), EndJob()
- StartDoc(), EndDoc()
- StartPage(), EndPage()
- Job, doc and page attributes are s each StartXxx() function.
- PWG/UPDF is used to describe attributes



Query Device Capabilities and Information

- QueryDeviceCapability()
  - Query if the device can do number-up, duplex, etc.
  - Information such as media size, media source and etc. which are supported by the device can be retrieved.
- QueryDeviceInfo()
  - Query current settings of the device.

Graphics State Object Operations

- Graphics State is managed as GS object
  - Operation to GS – InitGS, SaveGS, RestoreGS
- Controls to each items in GS
  - CTM (Coordinate Translate Matrix)
  - Color Space
  - Raster Operation – ROP3
  - Fill Mode – even/odd or winding
  - Alpha Constant
  - Line Style – width, dash/solid, cap, join
  - Paint Mode – opaque or transparent
  - Stroke and fill color – brush control
  - Foreground and background color – solid brush

Path Operations

- A path is a virtual track object
  - Will be visible by stroke or fill operations
  - Will be used to define clip region
- Lines, rectangles, polygons, arc/pie and Bezier are all treated as "path."
- Operations:
  - NewPath() – Declare start of a path
  - EndPath() – Declare end of a path
  - StrokePath(), FillPath(), StrokeFillPath() – make visible path
  - SetClipPath(), ResetClipPath() – defines clip region by current path

Text Operations

- Not defined in API Version 1.0
- Text Operations will includes:
  - Text Operations
    - SetFont / GetFont / SetCharset / GetCharset / SetTextAttribute / GetTextAttribute / SetFontMatrix / GetFontMatrix / GetTextExtent / DrawText / PathText
  - Font Downloading Operations
    - BeginFontDownload / TransferFontHeader / TransferFontData / EndFontDownload / DeleteFont

Bitmap and Scanline Operations

- Bitmap is a bit oriented image data drawn in rectangle region
  - DrawImage()
  - StartDrawImage(), TransferDrawImage(), EndDrawImage()
- Scanline is a horizontal line defined by start and end point pairs.
  - Used to draw graphics rendered by renderer
  - StartScanLine(), ScanLine(), EndScanLine()

Raster Image Operation

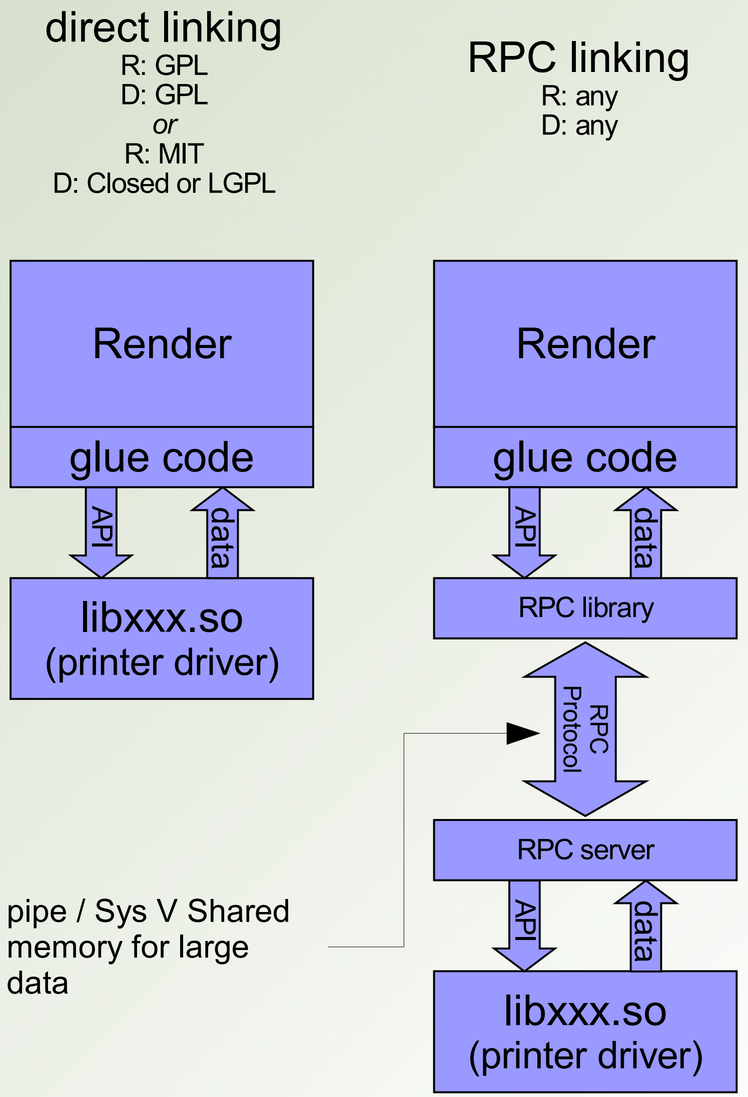
- If the device does not any graphic primitives, raster image can be sent by these operation.
- StartRaster(), TransferRasterData(), EndRaster()

Stream Data Operations

- Direct PDL embedding is possible by these operation.
- Can be used for “form printing”, eps embedding, or direct device control.
- StartStream(), TransferStreamData(), EndStream()

Linking with Render

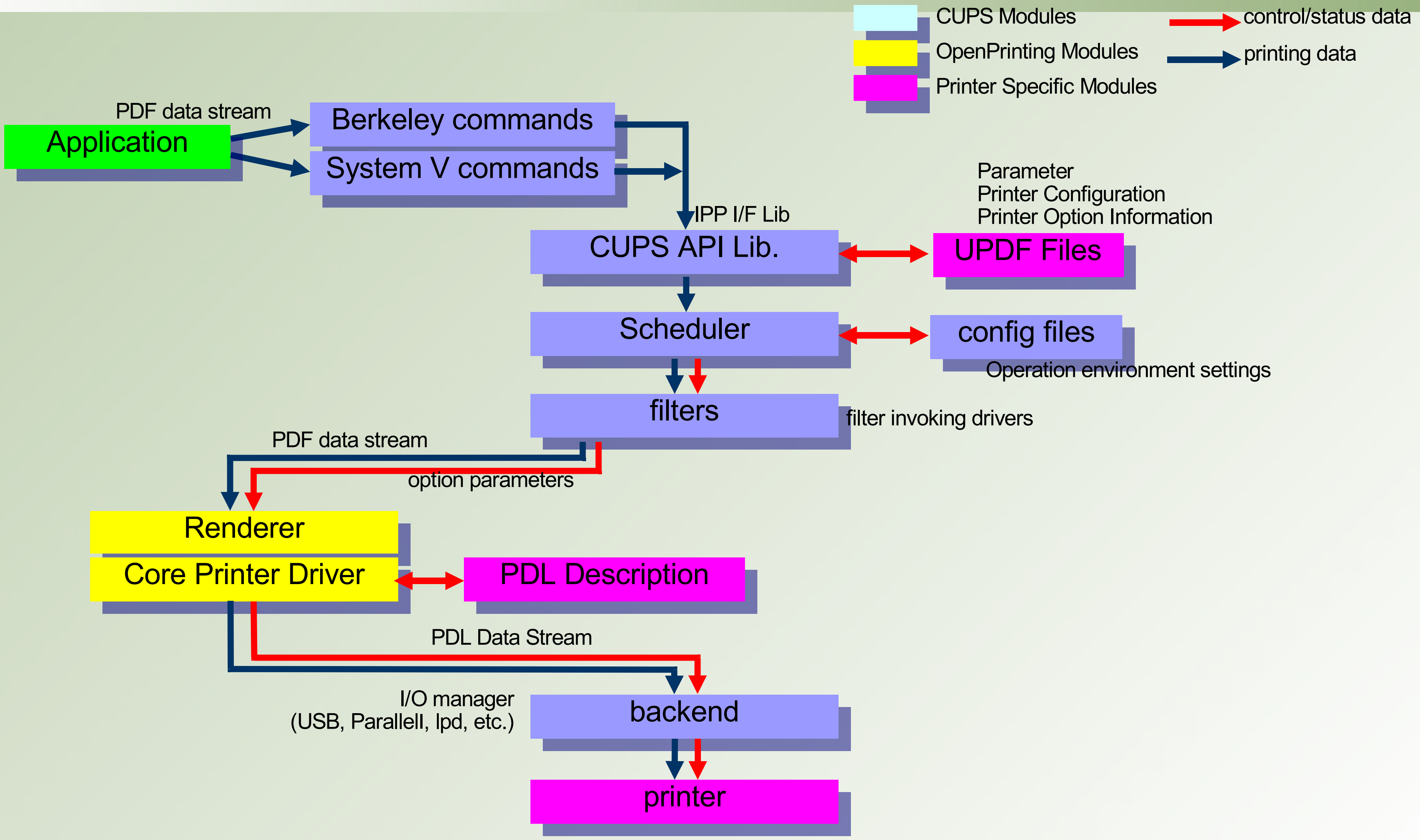
- Printer driver is provided as a dynamic library.
- Driver can be linked dynamically or via RPC (uses pipe and Sys V shared memory).



Update for Version 1.0

- Currently working for formal release as Version 1.0.
- Changes from 0.2:
  - Document License: FDL to MIT
  - Symbols have “fsgpd” prefixes.
  - Tentative font operation is removed (no font support yet – sorry!)
  - OpenPrinter() now handles API spec version.
  - Change of parameters of raster functions (DrawImage(), StartDrawImage())
  - Scheme for Job/Doc/Page attribute: support of UPDF become mandatory.
  - Support of KRGB for inkjet devices
  - Many other fixes.
- GS meta driver (opvp) will be updated when Version 1.0 when it is available. Driver developers are encouraged to apply version 1.0.

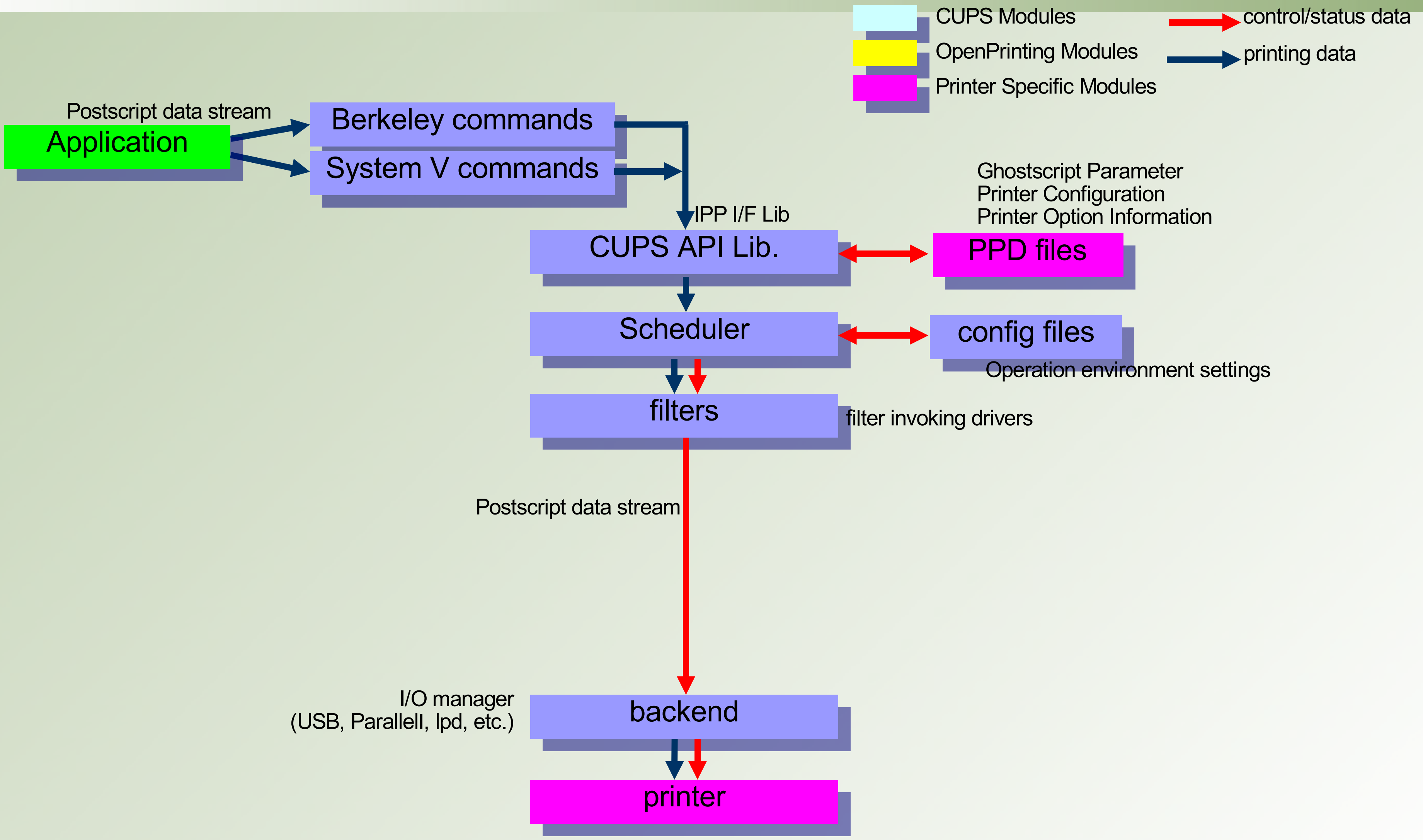
Future: Core Printer Driver



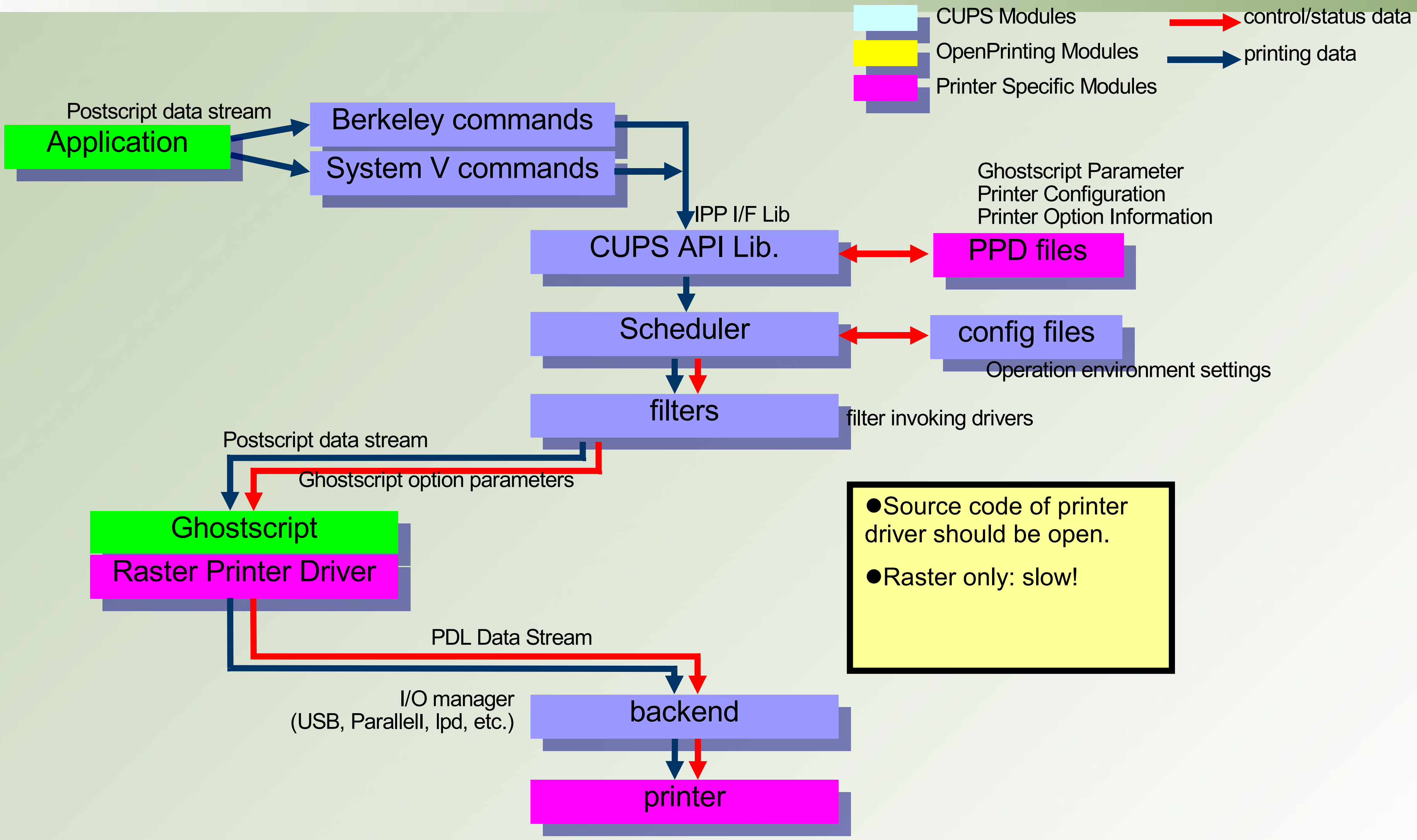
Thank You!

Appendix
Printer Driver on Linux Platform

PostScript Printer

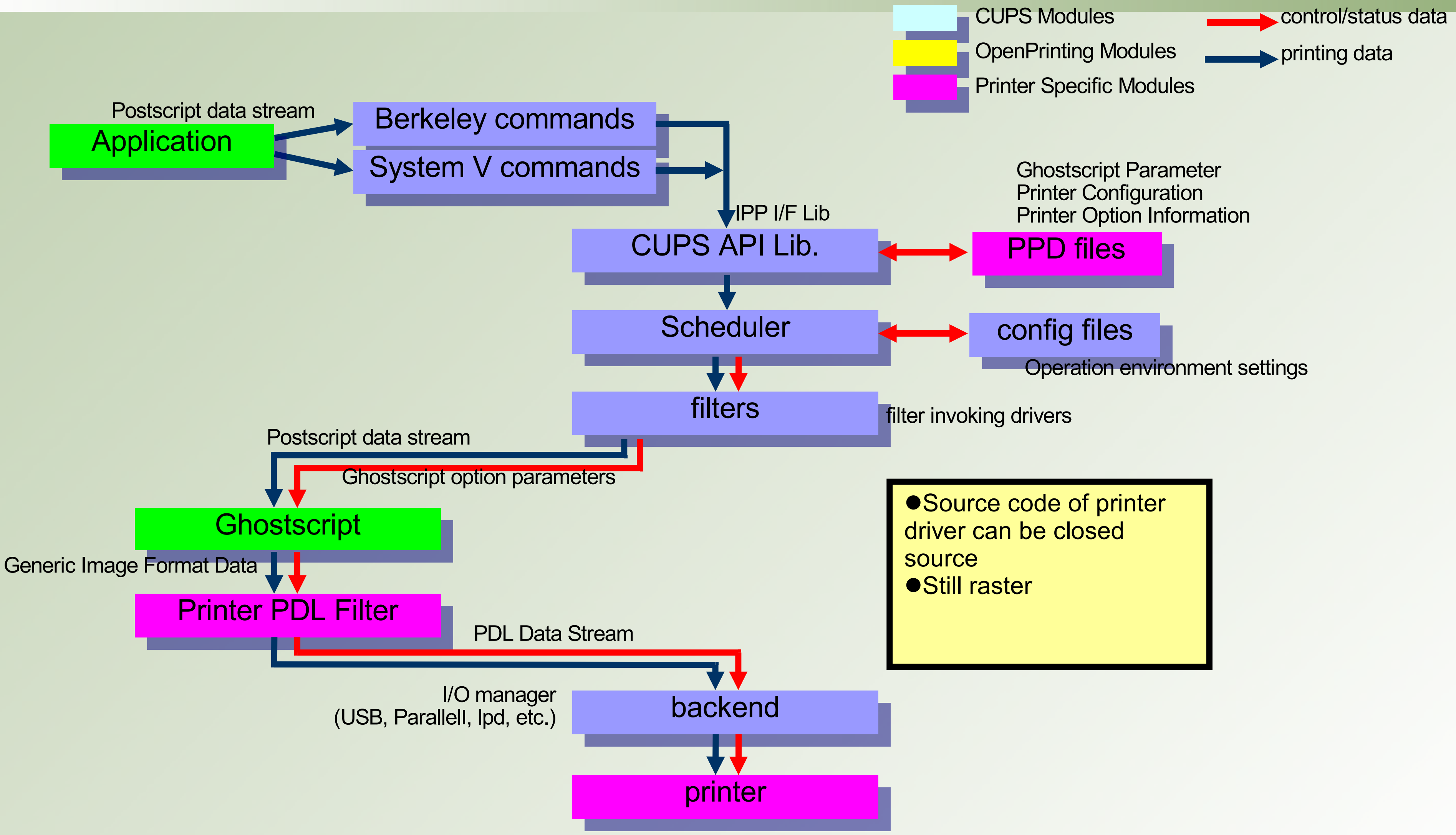


Ghostscript+Raster Printer Driver



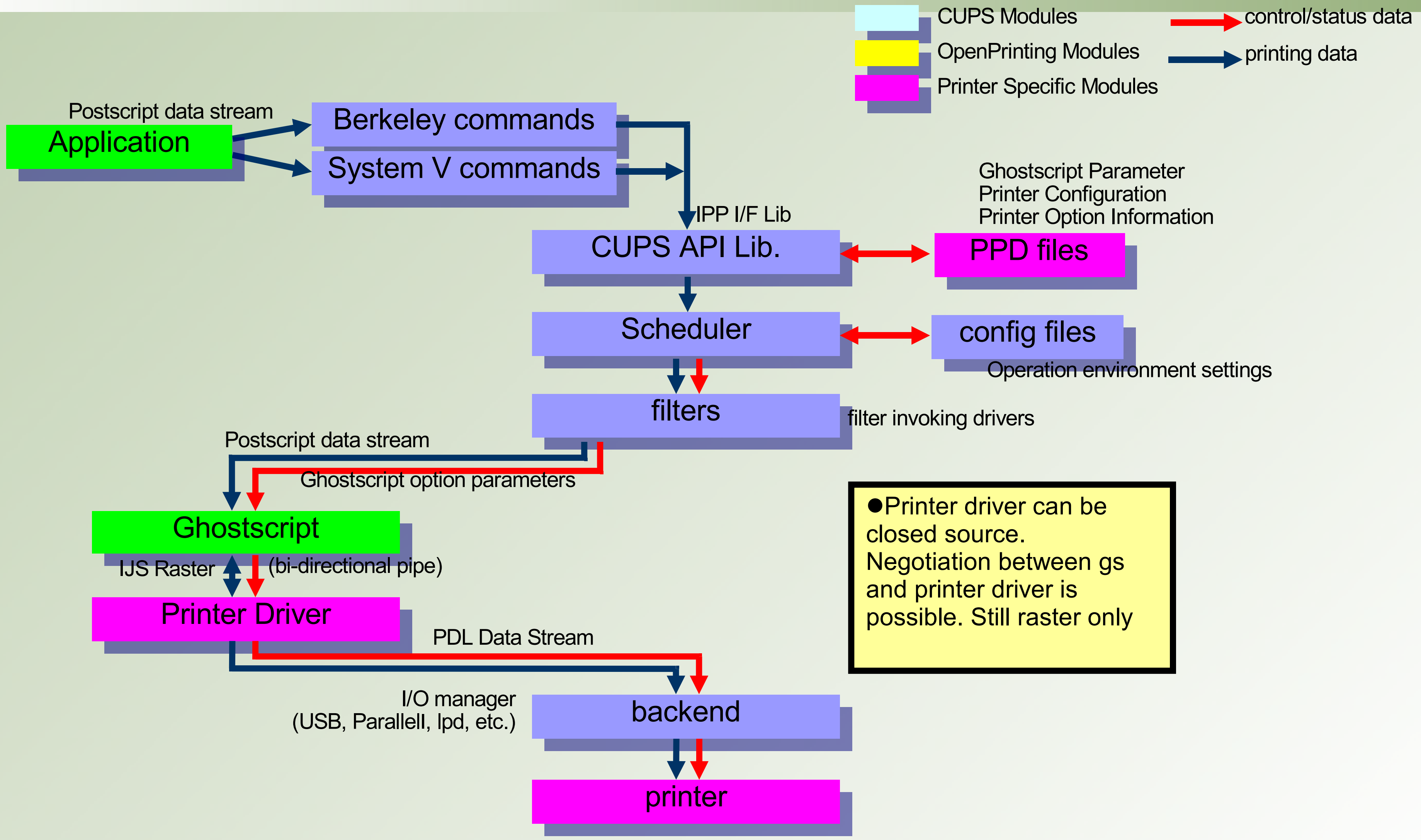
- Source code of printer driver should be open.
- Raster only: slow!

Ghostscript+Filter Program



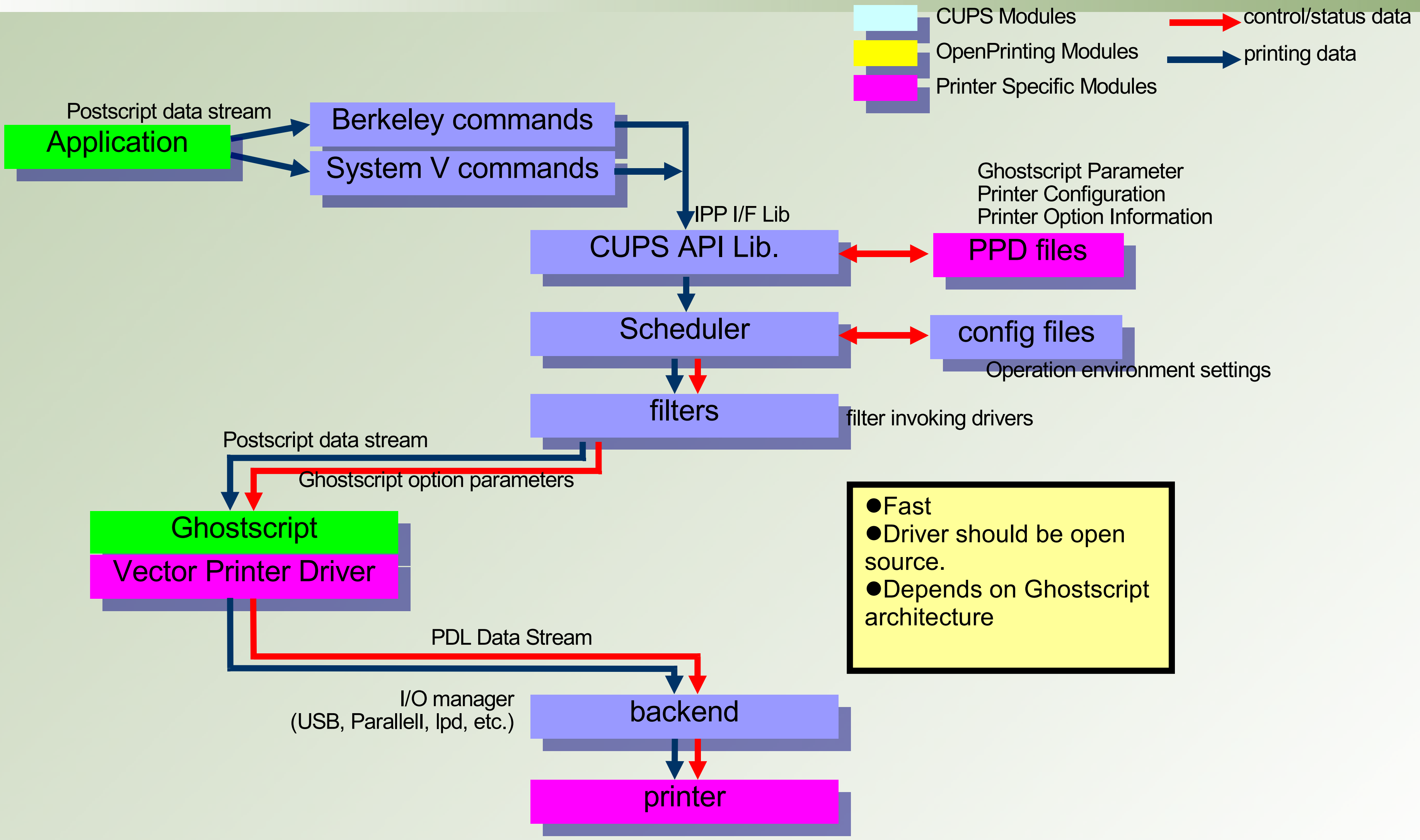
- Source code of printer driver can be closed source
- Still raster

Ghostscrpt+IJS



●Printer driver can be closed source. Negotiation between gs and printer driver is possible. Still raster only

Ghostscript+Vector Printer Driver



- Fast
- Driver should be open source.
- Depends on Ghostscript architecture