cairo-gl

Eric Anholt

What raster backends are there?

- xlib: built on XRender
- xcb: built on XRender, but better.
- image: built on pixman
- glitz: built on an abstraction of XRender, now deleted
- drm: experiment by Chris Wilson to do custom Cairo drivers

Why not Xlib?

- Half of it was never implemented
- Slow

Why not image?

- Most of us want to look at the graphics we draw
- The cost of uploading image results to the screen

Why not glitz?

- Glitz abstracted my abstraction layer
- Few performance wins, some performance hits
- Unmaintained

Why not drm?

- I'm a driver developer.
- I'm lazy.
- It's really hard.

Introducing cairo-gl

- Direct-rendered cairo on proven hardware abstraction
- Everyone needs a good GL driver anyway
- Implementation is simple.
  - cairo-gl: 6362 LOC (+43889 LOC in i965 3D driver)
  - cairo-xlib: 8317 LOC (+3956 LOC in i965 2D driver)
  - cairo-drm-i965: 15982 LOC
- Maybe we can make cairo as fast as everyone hopes it can be.

How does it work

- Cairo has 5 basic operations:
  - paint
  - mask
  - fill
  - stroke
  - glyphs

mask

- Basic operation for cairo backends
- Source is a color, gradient, or texture
- Mask is a color, gradient, or texture
- Multiply source by mask alpha and blend
- First operation supported by cairo-gl

paint

- This is mask without a mask
- We don't implement it so cairo makes it into a mask call

fill and stroke

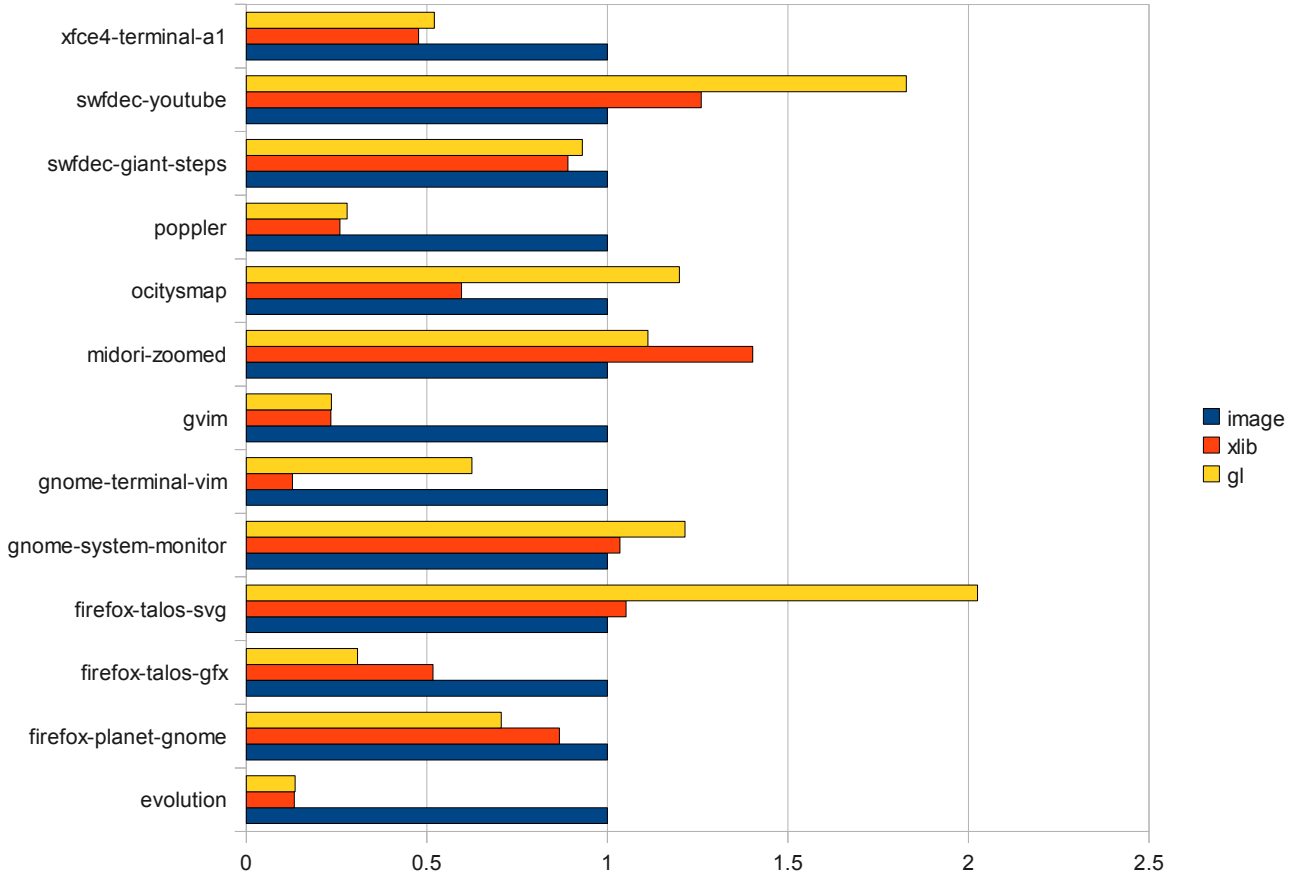
- Cairo computes a polygon outline
- Cairo tesselates the polygon outline to spans
  - x1
  - x2
  - y
  - coverage
- Cairo-gl takes spans and emits rectangles with coverage in the color attribute
- The GPU likes this: lots of vertices with the same state

glyphs

- App uses pango or something to lay out text
- Cairo gets a list of glyph IDs and positions to draw
- Cairo-gl loads the glyphs from the font into a texture atlas
- Emits rectangles loading the mask from the texture atlas
- Lots of vertices, the GPU is good with this

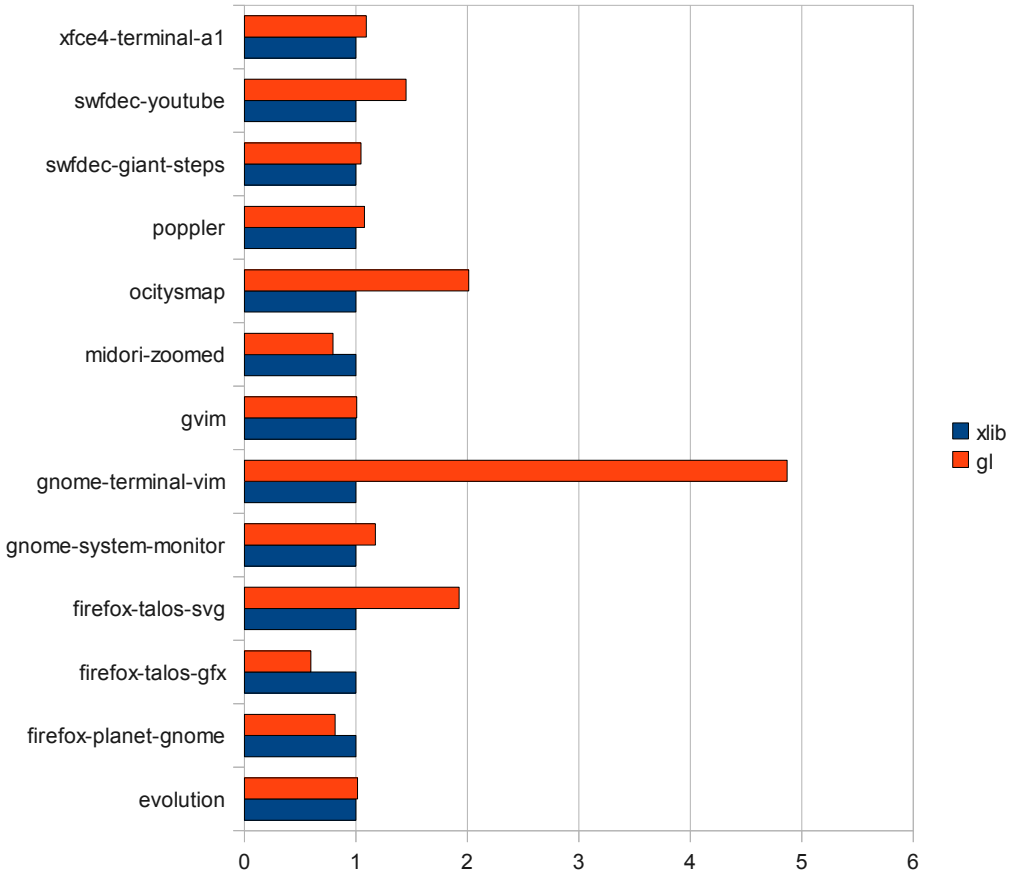
How is performance?

- bigger is better



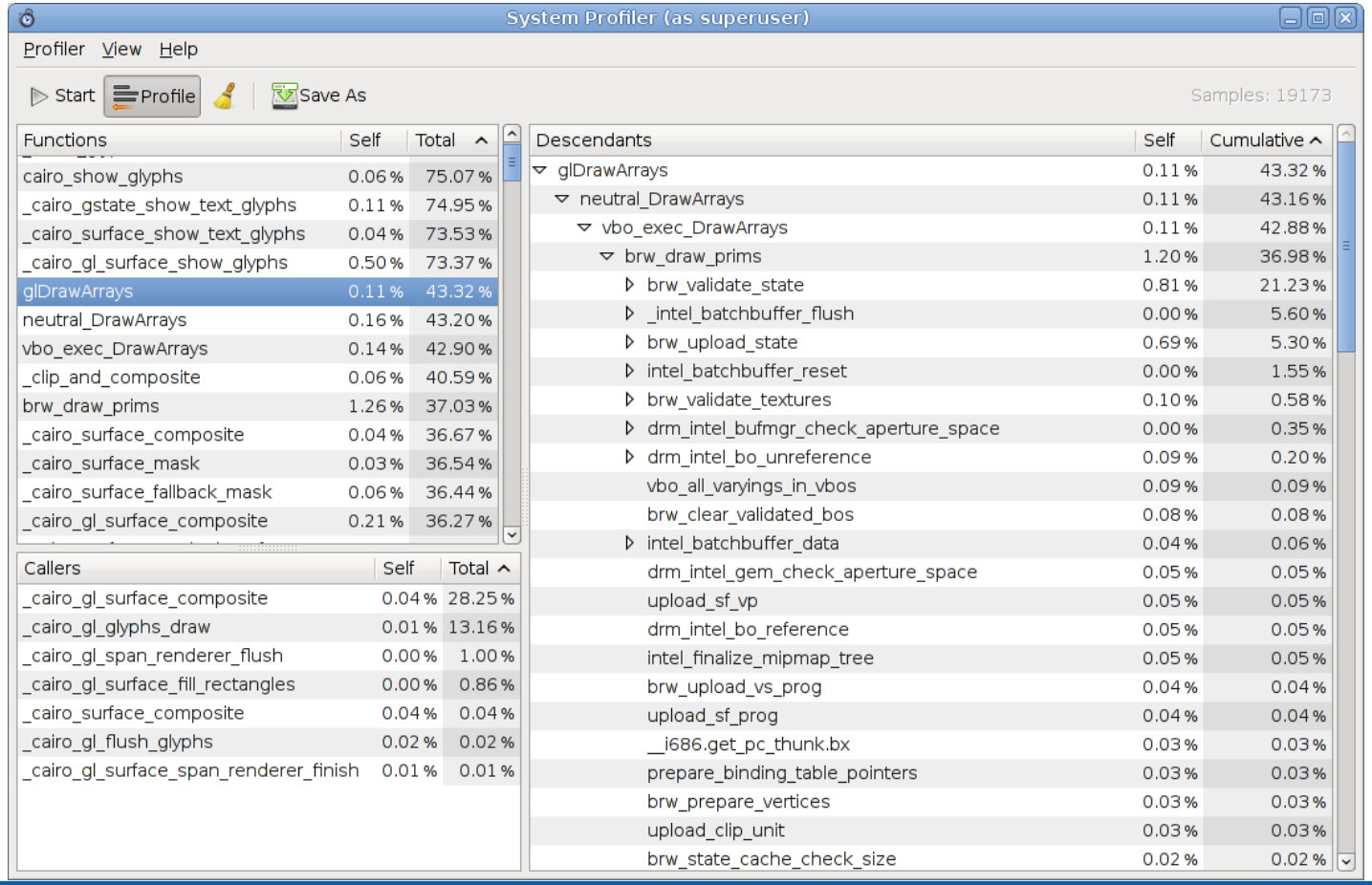
A more exciting chart

- bigger is better



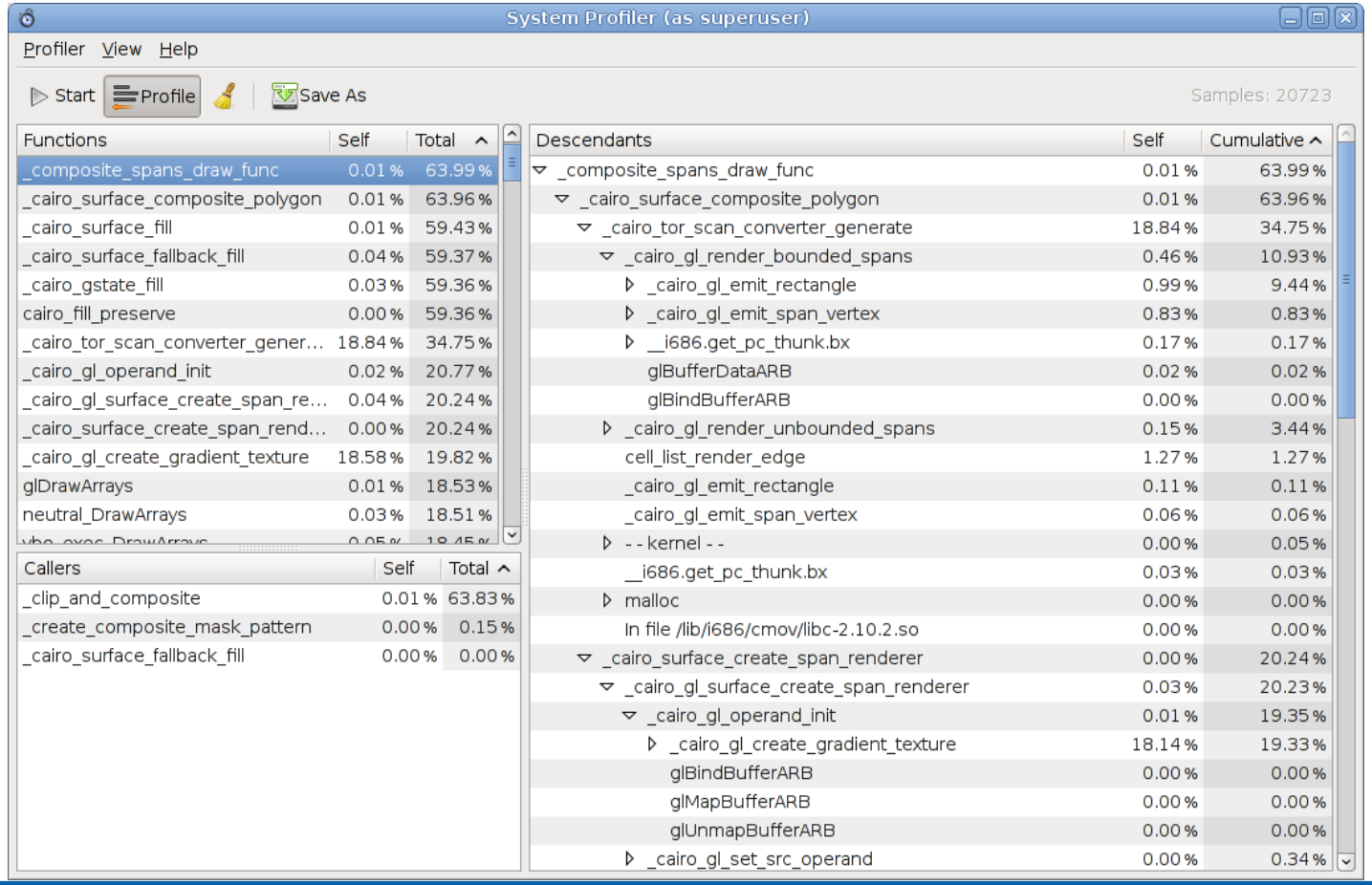
What's still painful?

- 2D is all about glyphs (firefox-talos-gfx)



What's still painful

- gradients too (firefox-talos-svg)



More experiments

- Move to a smaller transformation matrix
- Transformation matrix and other constants as attributes
- Shorter gradients shaders
- Loop-Blinn design to replace spans
- Replace QUADS with TRIANGLES (index buffer or not?)
  - ARB_primitive_restart?

Other directions

- GLES 2.0 support
- Non-Mesa GL drivers
- Large surfaces

Conclusion

- Faster than today's Xlib
- Easy code to hack on
- Not yet beating CPU drawing on integrated graphics
- Lots of room to improve