Hardware-Accelerated Graphics on Microkernels

Jamey Sharp, Galois

2015-09-18
what’s a microkernel?
a definition

... the near-minimum amount of software that can provide the mechanisms needed to implement an operating system (OS).
— https://en.wikipedia.org/wiki/Microkernel
what does that mean, really?

Figure 1: monolithic vs. microkernel (from Wikipedia, public domain)
who cares?

do you trust your kernel? are you sure it:

- doesn’t crash, overrun buffers, write to random memory
- doesn’t leak information to untrusted processes
- enforces full isolation between processes
- ensures the highest-priority process is the one that’s running
seL4

a modern microkernel, 9k lines of formally verified C:

▶ proven not to overrun buffers or invoke undefined behavior
▶ proven to enforce *isolation* between processes
▶ proven to not access the wrong memory
cost of formal verification

optimistic cost estimate:

- over $1 trillion to write a formally verified Linux kernel
cost of formal verification

optimistic cost estimate:

- over $1 trillion to write a formally verified Linux kernel
- that’s 10% of the US GDP
cost of formal verification

optimistic cost estimate:

- over $1 trillion to write a formally verified Linux kernel
- that’s 10% of the US GDP
- (but it’s only 3x the SLOCCount estimate)
NOVA “microhypervisor”

Figure 2: NOVA architecture
Genode: portable userspace for microkernels
microkernel ⇒ no drivers in kernel

where do you get device drivers from?

▶ every research microkernel writes their own drivers
microkernel ⇒ no drivers in kernel

where do you get device drivers from?

▶ every research microkernel writes their own drivers

“Genode” project:

▶ drivers portable to 8+ microkernels
  ▶ x86 and ARM
  ▶ basic framebuffer and input drivers
  ▶ sound, block, network, usb, uart
  ▶ filesystems: FAT32, ext2, etc.

▶ POSIX-ish libc, Qt, and other porting aids for userspace
▶ VMs (in VirtualBox or Seoul) alongside native components
ready for prime-time?

some brave souls now run Genode with a Linux VM as their primary desktop (!)
microkernel-friendly graphics architecture
current Linux graphics architecture
straw-man microkernel graphics architecture
straw-man microkernel graphics architecture

Application

API: OpenGL

libGL

DRI-1.0-style Device Driver

libDRM

Mesa 3D

API State Tracker

GPU-specific Device Driver

OS WinSys

Kernel

DRM

CPU & registers & L1 & L2 & L3 & L4 & main memory

GPU & registers & L1 & L2 (& graphic memory)
original work by Norman Feske of Genode Labs in 2010
“proof of concept”: wrap Linux i915 driver in compatibility glue shove i915 in the GL client’s address space give GL client direct hardware access not quite what anyone wants, but proves the concept no Mesa changes needed clear path to a real graphics architecture
demo!
questions?

- http://galois.com
- jamey@galois.com
- @jamey_sharp
- http://genode.org