



Power Management for Graphics Systems

Dave Brown

Jay Cotton

Sun Microsystems



Power Management

- Moore's law
 - Performance -> power consumption
- Driving Factors
 - Laptops (battery life)
 - Servers (physical plant)
 - EPA's Energy Star
 - Energy: strategic concern for USA

System Design Consequences

- Hardware systems
 - Powered off
 - Reduced power operation
- System Software
 - Power management facilities

Graphics subsystems

- Implications?
 - System Software
 - Graphics Hardware
- Graphics Power Management
 - X.org
 - nVidia, ATI, Intel...

Design for Graphics PM

- Devices of interest
 - Displays -
(DPMS: historical usage)
 - Graphics Controllers -
Frame Buffer Power Management (FBPM)

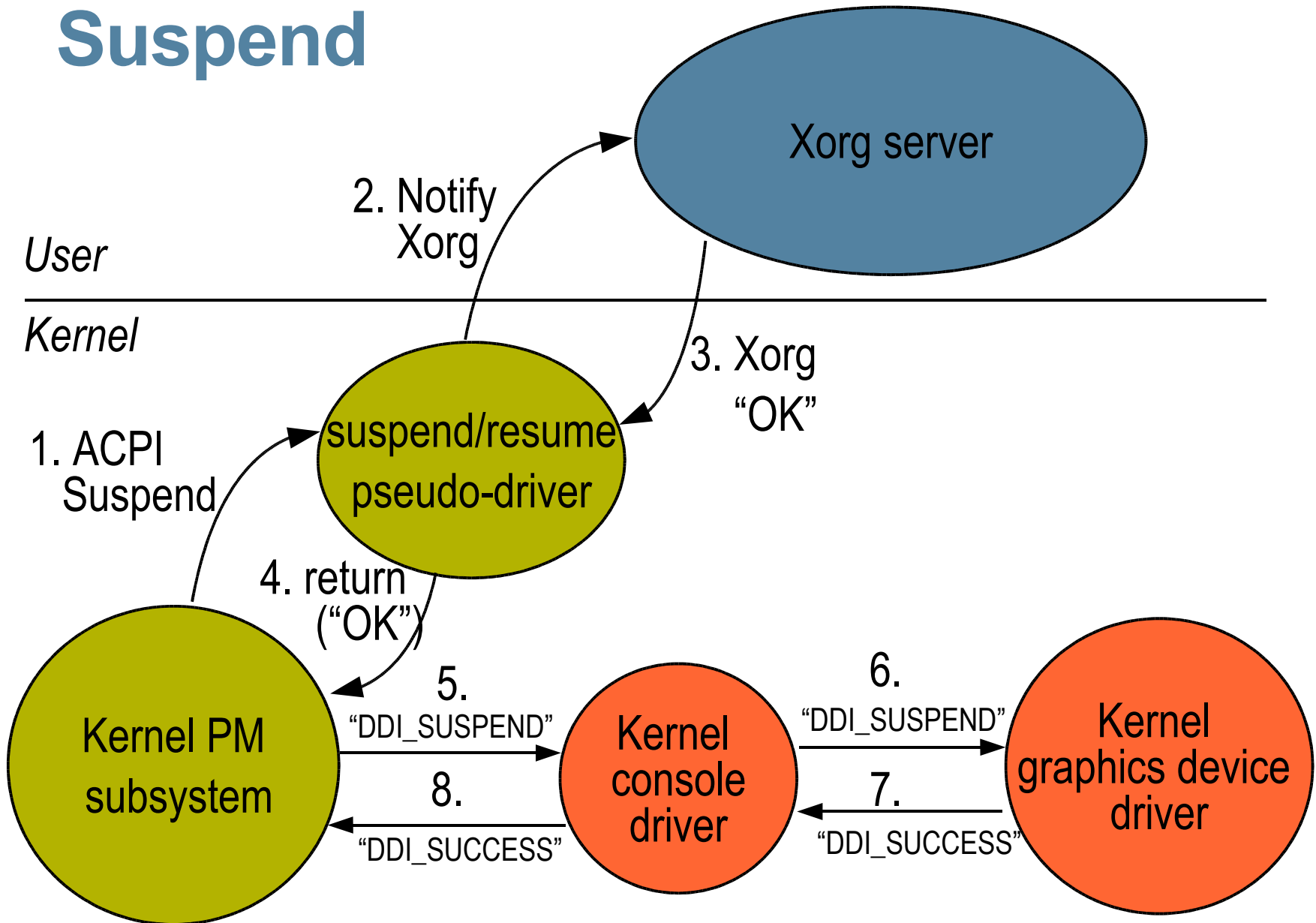
Design for Graphics PM

- Two primary modalities
 - Suspend/Resume (ACPI S3)
Whole graphics subsystem On or Off
 - Reduced Power (“shrink to fit”)
Adjust performance of graphics device

Suspend/Resume

- Mechanism
 - ACPI specification
 - Provides S3, S4 ... power states
- Overview of implementation
 - Xorg's interface is via Kernel PM subsystem
 - `ioctl`s using suspend/resume device driver

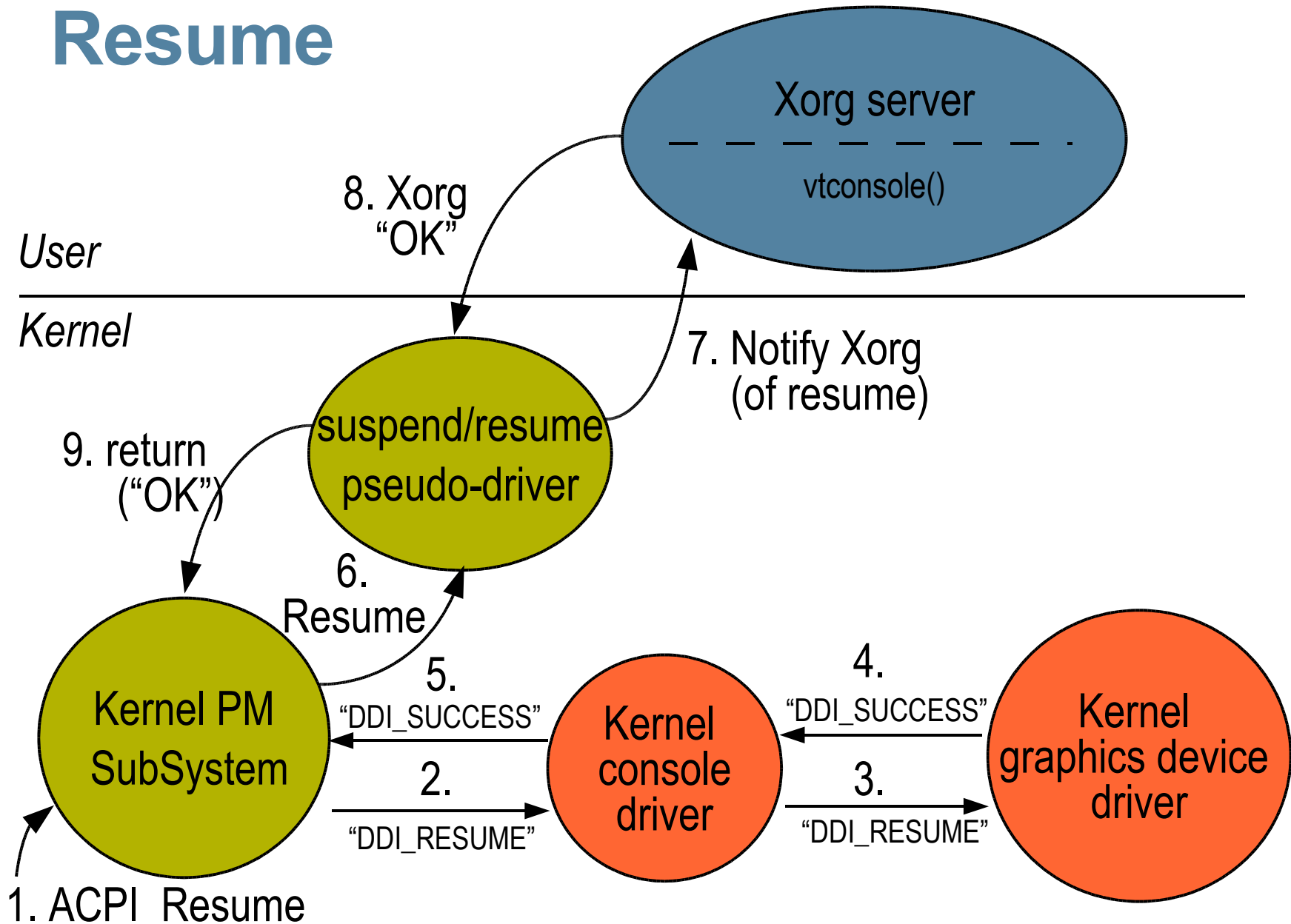
Suspend



Suspend (ACPI S3)

- Kernel's PM subsystem notifies X.org server (using pseudo device driver interface)
- Xserver returns to terminal mode (via vtconsole code) and notifies kernel
- Kernel tells console driver to suspend (DDI_SUSPEND sent to the graphics device that's bound to the console)
- Kernel graphics driver saves its device state
- Kernel PM subsystem powers devices off

Resume



Resume (ACPI S3)

- BIOS sees power request via ACPI, and turns on all hardware; then jumps to kernel's restart entry point
- Kernel's PM subsystem tells console driver to resume in VGA text mode (DDI_RESUME sent to graphics device bound to the console)
- Kernel graphics driver restores saved device state
- Kernel's PM subsystem notifies Xserver to resume graphics operation
- Xserver restores graphics state (via vtconsole code) and notifies kernel once done

Reduced power operation

- Provides ability to:
 - Run at lower speed
 - Disable portions of the graphics controller
- PCI bus specification
 - Provides mechanism (D0..D3 power states)
 - Interface is via Kernel PM subsystem:
(Xorg performs ioctls using /dev/pm)

Framebuffer Power Mgmt (FBPM)

- Xorg tells DDX: “Change power state”
`*FBPMChangeStateNotice(Dn)`
- DDX tells /dev/pm to change power state
`ioctl(pm_dev, PM_CHANGE_POWER)`
- PM subsystem calls Kernel Graphics driver:
`gfx_power(change_power_state, Dn)`
`gfx_power()` returns “OK”
- PM subsystem returns “OK” to Xorg/DDX

X Driver Writers

- If you are a DDX developer
 - Your graphics driver should support
 - Suspend/Resume
 - and Framebuffer Power Mangement
- You can use this design if you want!

Kernel Driver Writers

- If you develop kernel graphics drivers (BSD, Linux)
 - Ensure your drivers support
 - Suspend/Resume and FBPM
 - (Enhance existing drivers)
- You can use the design described if you want!

Graphics Vendors

- If you develop graphics controllers:
 - Continue your involvement
 - Goal is a uniform interface (under Xorg) for Framebuffer Power Management
- We suggest the design (and RI) described as a starting point for discussion

Web sites and email addresses

- David Brown djb@Sun.COM
- Jay Cotton Jay.Cotton@Sun.COM
- Open Solaris Project <http://OpenSolaris.ORG>
- And discussion under:
 - blogs.Sun.COM/roller/page/jaycotton

Power Management for Graphics Systems

• **David J. Brown and Jay Cotton**

• djb@sun.com, jay.cotton@sun.com