OS Virtualization

Presented by: Louis Giokas, Member IEEE

Purpose

- Provide some history of Operating System Virtualization
- Discuss use in mid-range to personal systems
- Review use on handheld devices using Open Kernels Labs' system as an example

Show the evolution of OS Virtualization from the very large systems to the smallest

Agenda

- History: IBM VM
- PC Virtualization: VMware
- Mobile Devices: Open Kernel Labs (OKLabs)

IBM VM

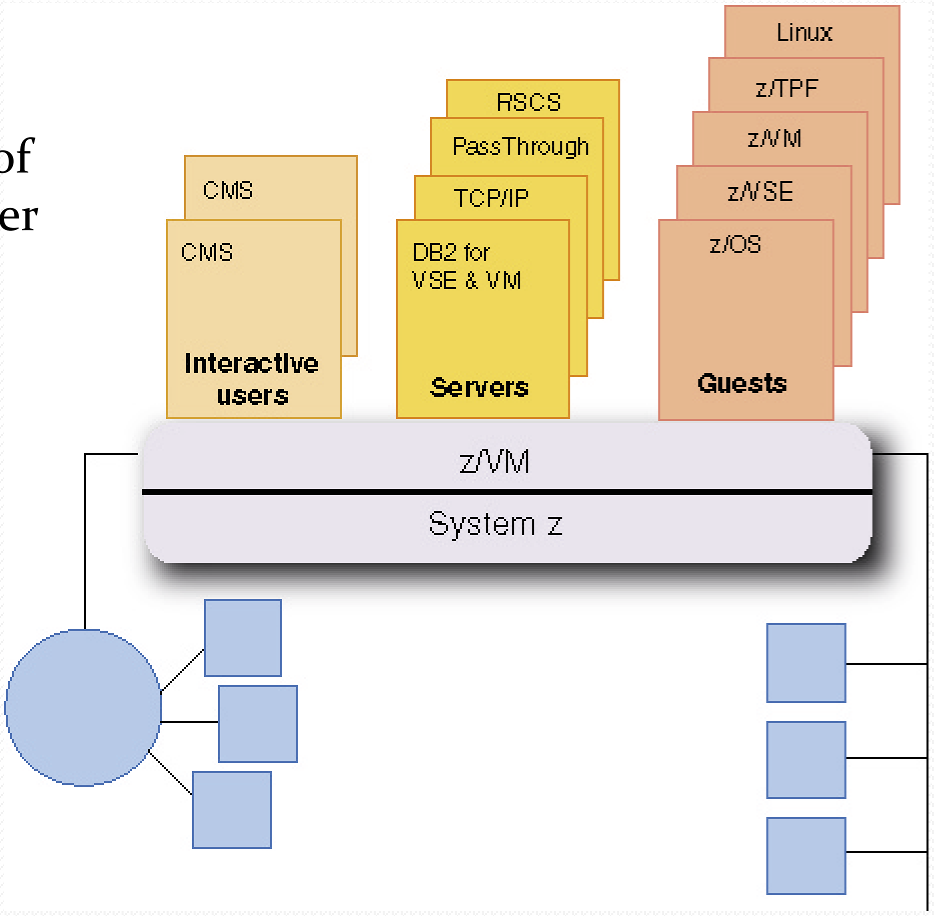
- VM for Virtual Machine
- Introduced with the S/370 which introduced virtual memory support
- Originally “Open Source” (from early 70’s to mid 80s)
- An outgrowth of the CMS (Cambridge Monitor System) – an early interactive timesharing system
- Originally used internally
- IBM was not interested in selling VM as it used hardware more efficiently
- Often referred to as VM/CMS
- Current offering referred to as z/VM

IBM VM - continued

- Advantages of VM
  - Efficient use of hardware resources
    - CPU resources are virtualized
    - External resources (disks, printers, etc.) are virtualized
  - Security – each OS image ran in a dedicated memory space defined by the virtual memory hardware
  - Robustness – one image can “crash” without affecting the rest of the system

IBM VM - continued

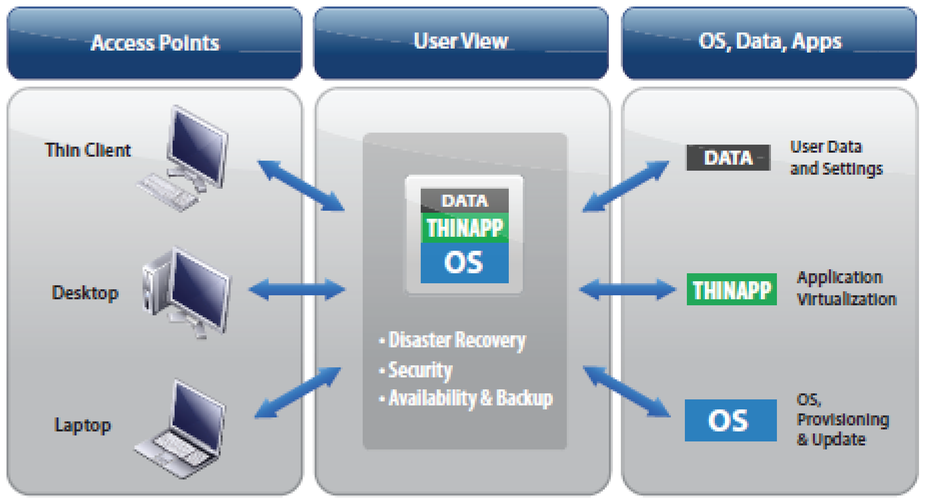
zVM takes controll of The system. All other OSs run as “guests”



Virtualization on the PC - VMWARE

- Bring virtualization to the workstation (PC) market
- Allow users to access multiple compatible OSs on their PC
- Access to software: do not have to install all server software on the host PC, just on the VM
- Easy switching between the host OS and the virtual image
- Depending on resources multiple images can be installed (multiple application environements)

Virtualization on the PC – VMWARE



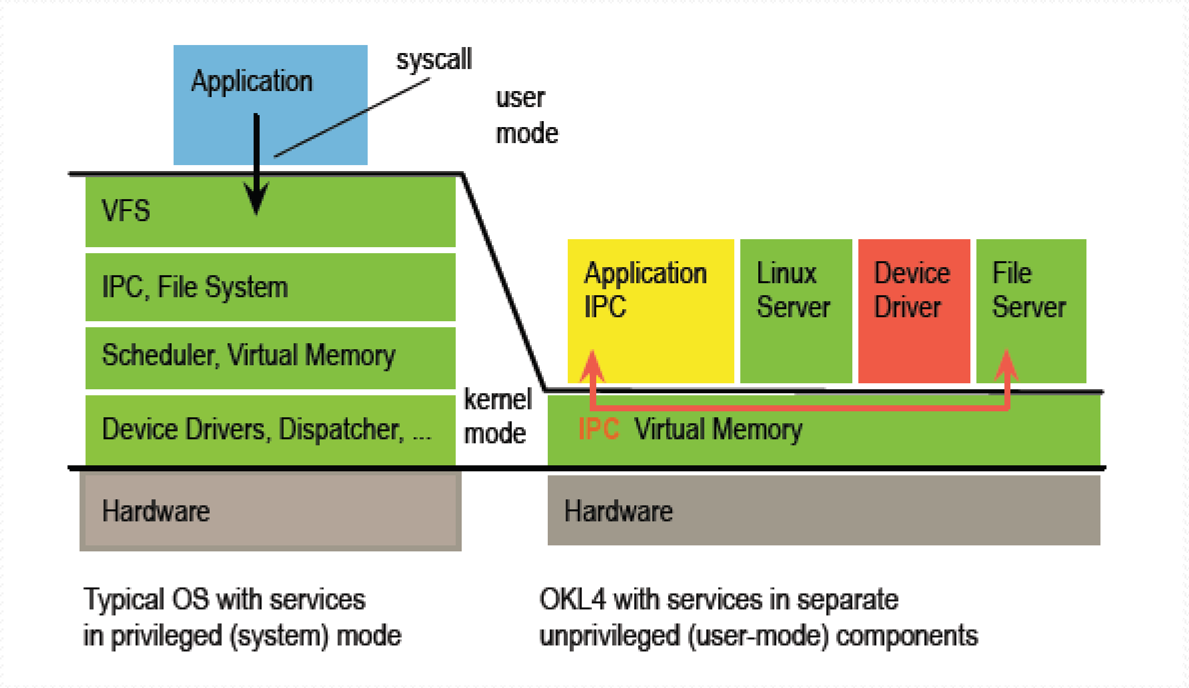
Virtualization on the PC – Other Vendors/Trends

- Microsoft: Hyper-V
  - Will become a major player
- Apple: Built-in virtualization allows users to run Windows and Linux under Mac OS.
- Data Center Virtualization
  - Move virtualization to the data center
  - More efficient use of resources such as blade servers
  - Multiple OS environment

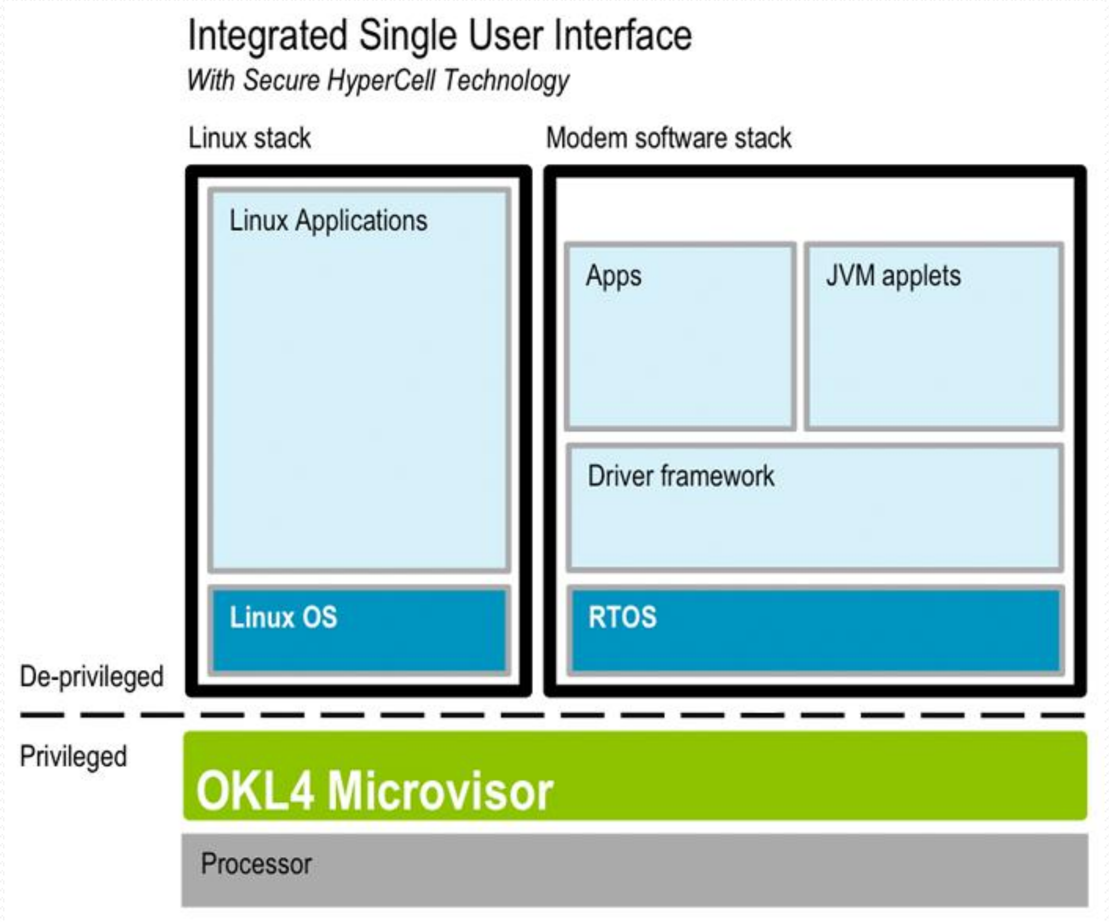
Mobile Device Virtualization

- What drives this application area:
  - Increasingly powerful processors
  - Complex handheld applications
  - Software Defined Radio (SDR)/Cognitive Radio
- Not only cellular handsets

Mobile Device Virtualization – OK Labs


Application
syscall
user mode
VFS
IPC, File System
Scheduler, Virtual Memory
Device Drivers, Dispatcher, ...
kernel mode
Hardware
Application IPC
Linux Server
Device Driver
File Server
IPC Virtual Memory
Hardware
Typical OS with services in privileged (system) mode
OKL4 with services in separate unprivileged (user-mode) components

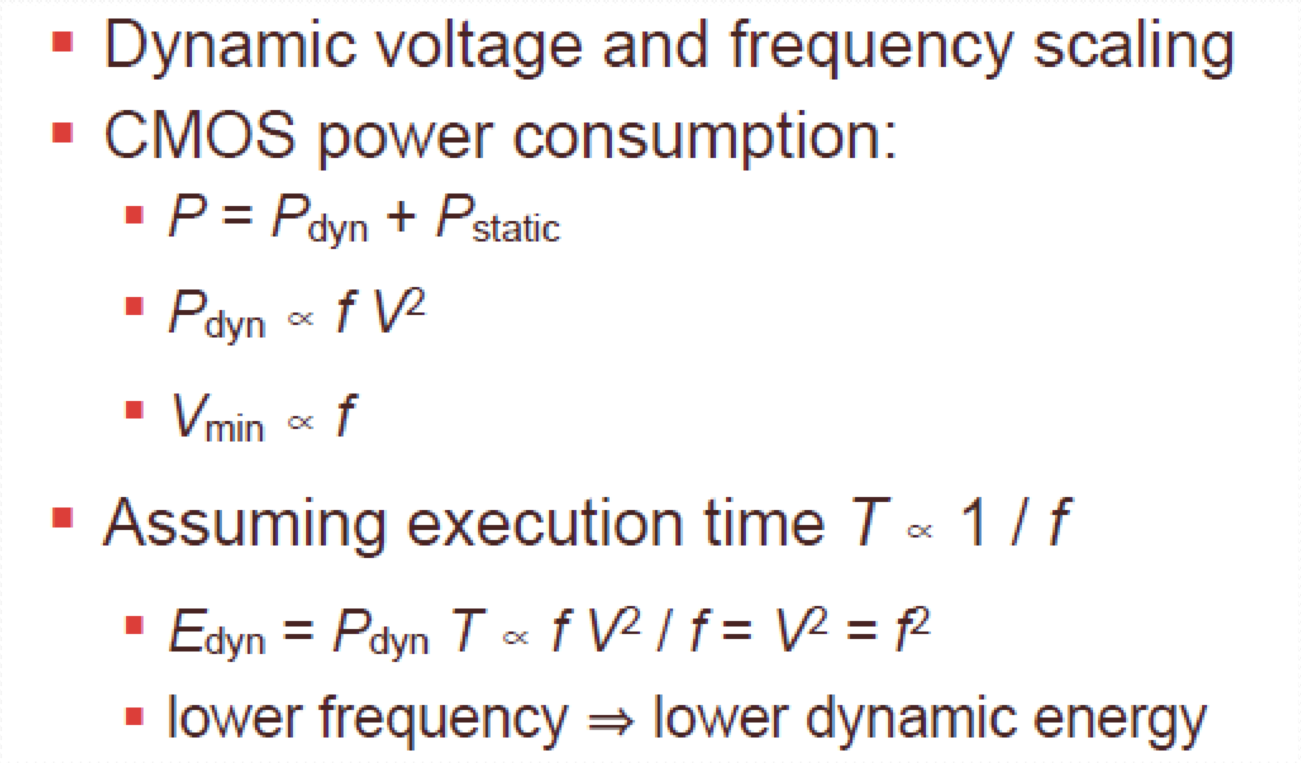
Mobile Device Virtualization – OK Labs



Mobile Device Virtualization – OK Labs

- Motivation and Important Features
  - Efficient use of resources (Power and cycles)
  - Security
  - Fault isolation
  - Ease of implementation
  - Ability to combine applications without having to test together in one image

Energy Management Mechanisms

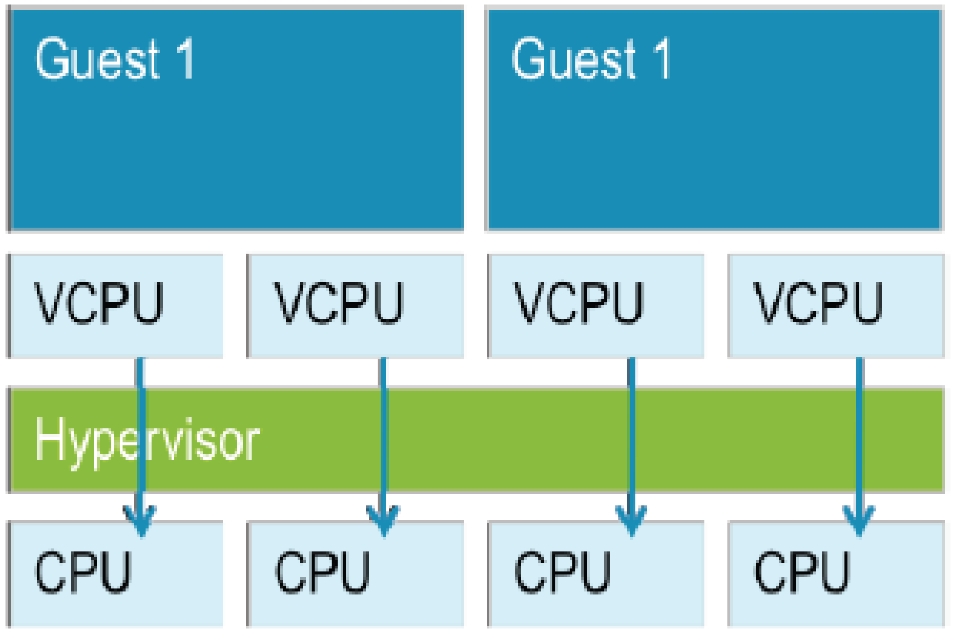


- Dynamic voltage and frequency scaling
- CMOS power consumption:
  - $P = P_{dyn} + P_{static}$
  - $P_{dyn} \propto f V^2$
  - $V_{min} \propto f$
- Assuming execution time $T \propto 1 / f$
  - $E_{dyn} = P_{dyn} T \propto f V^2 / f = V^2 = f^2$
  - lower frequency $\Rightarrow$ lower dynamic energy

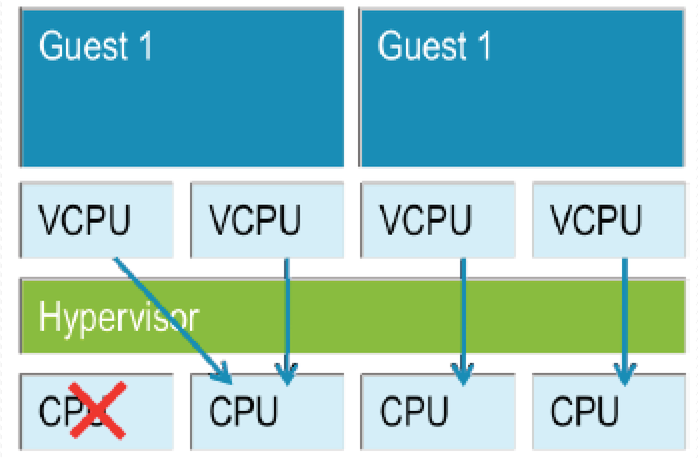
Energy Management

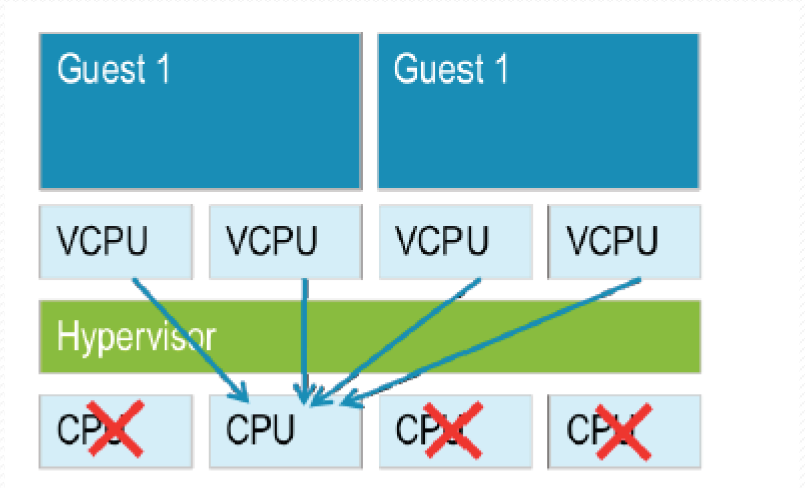
- **Microvisor for global resource management**
  - **energy is a global resource**
  - **how it's done**

Energy Management - Multicore

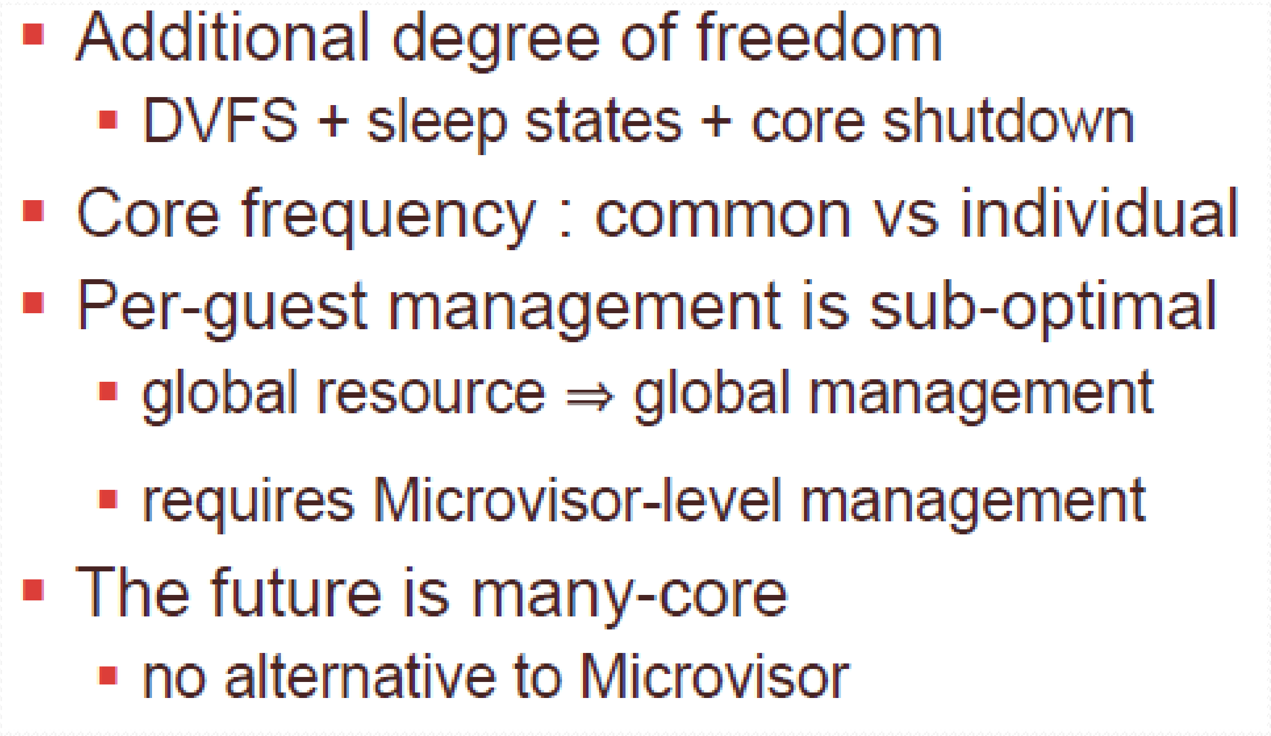


Energy Management - Multicore





Energy Management - Multicore



- Additional degree of freedom
  - DVFS + sleep states + core shutdown
- Core frequency : common vs individual
- Per-guest management is sub-optimal
  - global resource ⇒ global management
  - requires Microvisor-level management
- The future is many-core
  - no alternative to Microvisor

DVFS = Dynamic Voltage and Frequency Scaling

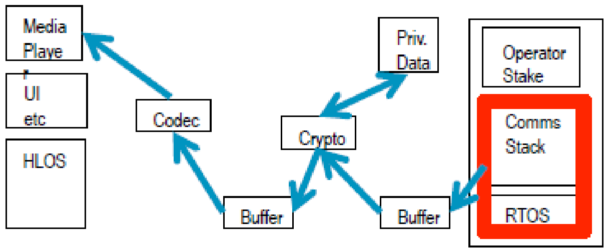
OKL4 Lightweight Execution Environments

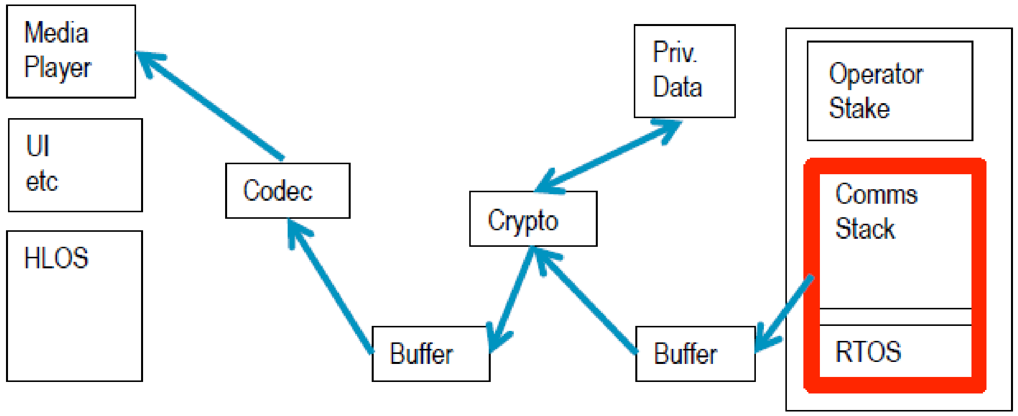
- Conventional virtualization solutions only support running complete operating systems (Linux, WinCE, …) in a virtual machine
- OKL4 supports execution of *lightweight execution environments* (LWEE)
    - A LWEE executes an application or system service directly on OKL4 without inclusion of a complete operating system
    - Critical services software can be hosted directly on top of the OKL4 Microvisor to reduce their *trusted computing base* (code they depend on for correct functionality)
        - Lower cost of cells without a guest OS allow practical application of finer grained isolation (i.e. More cells than what would be possible with vanilla VMs)
        - Separate security-sensitive and proprietary software from an Application OS (e.g. Linux, Android, Symbian) without running a second instance of an Application OS
        - Migrate DSP software components (e.g. multimedia codecs) to host processor without sacrificing real-time guarantees or increasing power consumption

Security

Modern embedded systems are multi-user devices!

- Eg a phone has three *classes* of "users":
- the network operator(s)
    - assets: cellular network

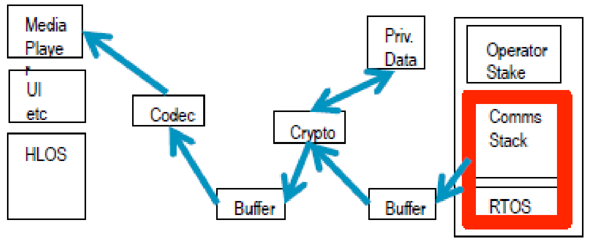


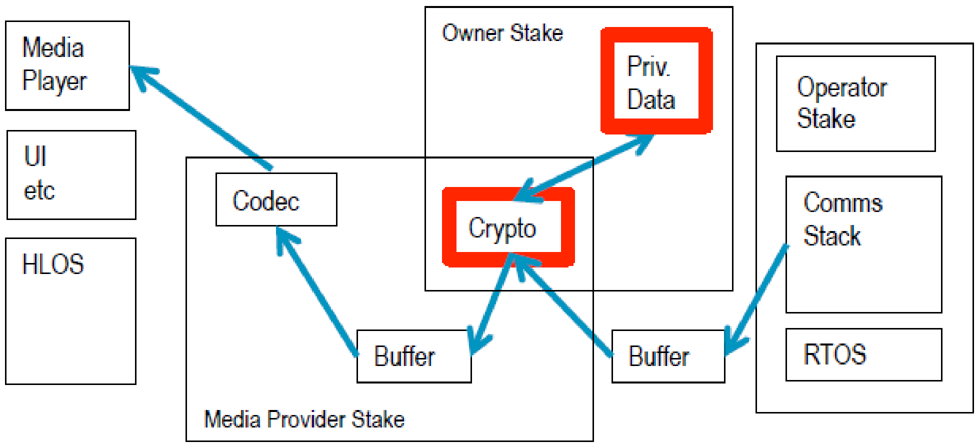


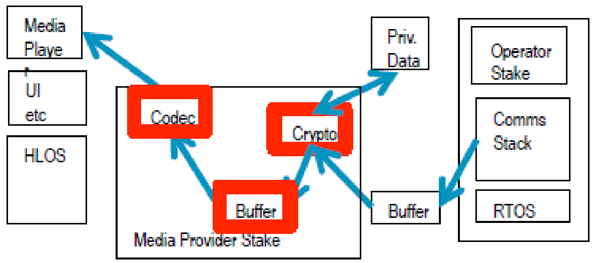
Security - continued

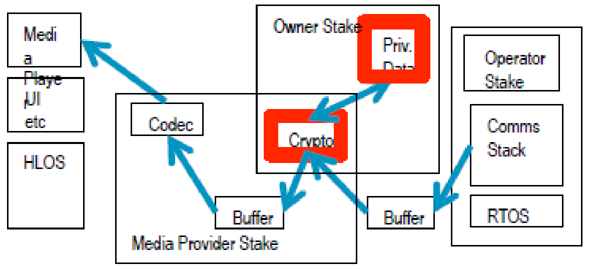
Modern embedded systems are multi-user devices!

- Eg a phone has three *classes* of "users":
- the network operator(s)
  - assets: cellular network
- content providers
  - media content





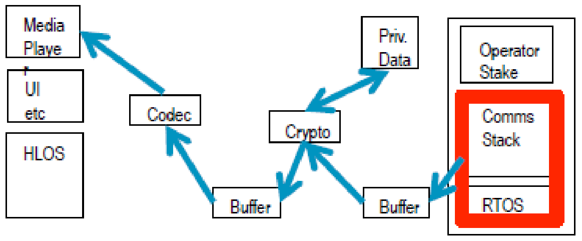


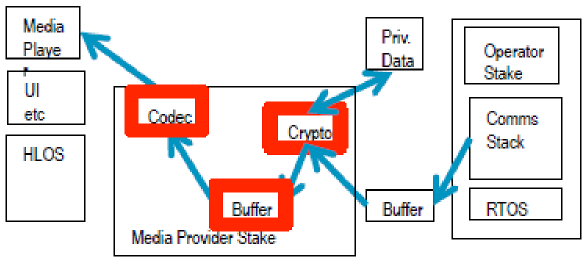


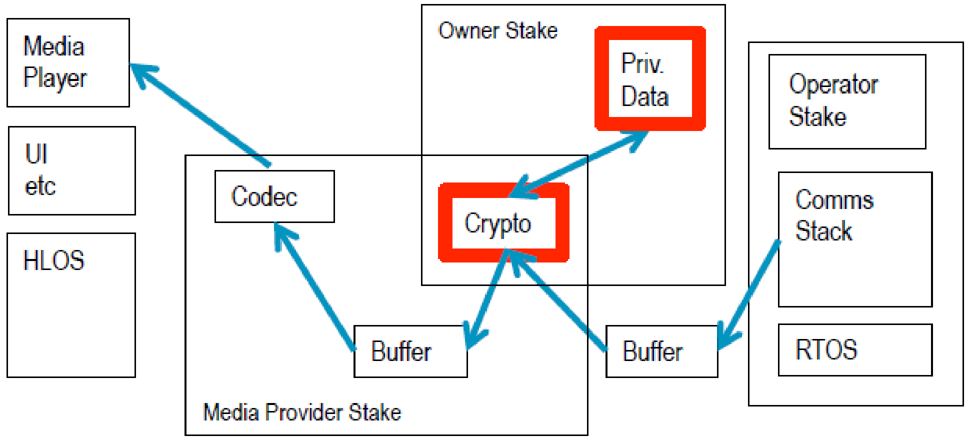
Security - Continued

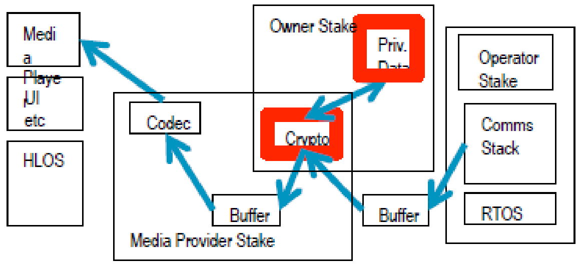
Modern embedded systems are multi-user devices!

- Eg a phone has three *classes* of “users”:
- the network operator(s)
  - assets: cellular network
- content providers
  - media content
- the owner of the physical device
  - assets: private data, access keys



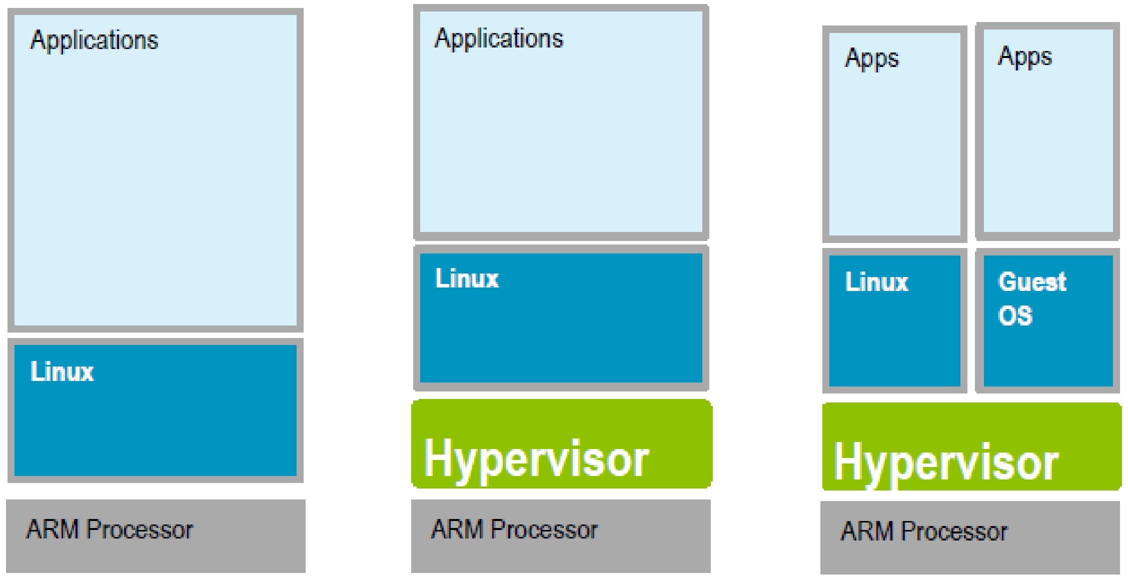






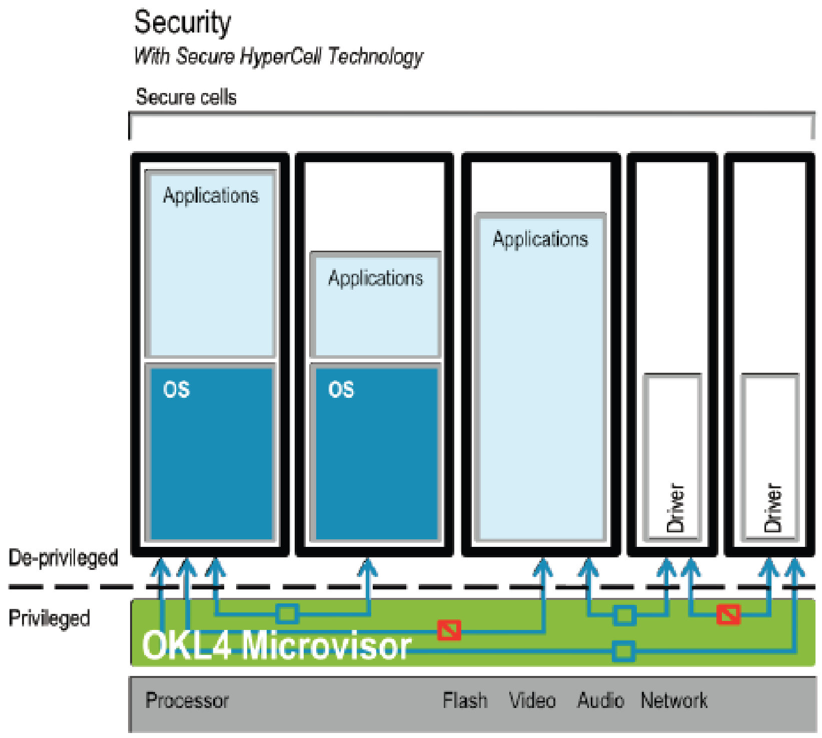
Security - continued

- OS normally runs directly on hardware
- Virtualization inserts software layer between OS and hardware
  - Called hypervisor or virtual-machine monitor
  - Supports several concurrent OSes, called guests



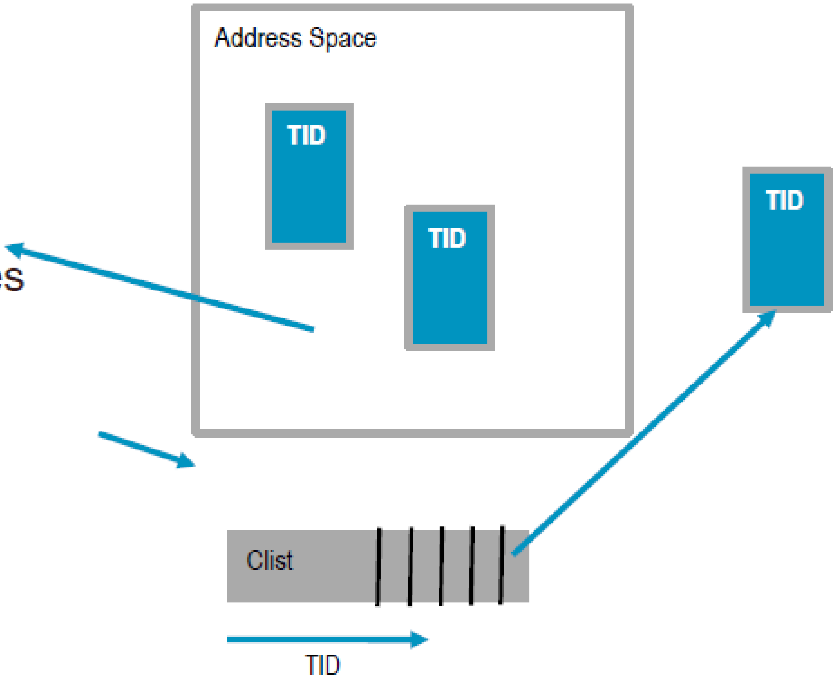
Security - continued

- Types of threats
  - Denial of service
  - Exposure of data
  - Unauthorized access
- Trusted components that are not trustworthy create vulnerability
- Designing for security
  - Isolation between trusted and un-trusted components
  - Principle of least authority/privilege
  - Minimal trusted computing base and complexity for trusted components
  - Protect integrity and confidentiality against *internal* and external exploits
  - Need control over *information flow*
    - who has access to what
    - communication channels



OKL4 Resource & Security Management

- Resources controlled by kernel-protected capabilities
- Capability conveys privilege
- Efficient resource delegation by providing set of caps
- Subject to system-wide security and resource-management policies defined by system designer
- Secure HyperCell Technology provides security management framework



Capabilities vs. ACLs

- Within computer systems, the two fundamental means of enforcing privilege separation are access control lists (ACLs) and capabilities. The semantics of ACLs have been proven to be insecure in many situations (e.g., Confused deputy problem). It has also been shown that ACL's promise of giving access to an object to only one person can never be guaranteed in practice. Both of these problems are resolved by capabilities. This does not mean practical flaws exist in all ACL-based systems, but only that the designers of certain utilities must take responsibility to ensure that they do not introduce flaws. [1]
- Mandatory access control can be used to ensure that privileged access is withdrawn when privileges are revoked. For example, deleting a user account should also stop any processes that are running with that user's privileges. [1]
- Capability and access control list techniques can be used to ensure privilege separation and mandatory access control. [1]

Security – Enterprise vs. Embedded

Mobile/Embedded

- Integrated system
- Cooperation with protection
- Inter-VM communication is critically important
- Performance crucial
- Optimize for ARM HW
- VMs are subsystems accessing shared (but restricted) resources
  - Communication must be controlled

Enterprise

- Independent services
- Emphasis on isolation
- Inter-VM communication is secondary
- Performance secondary
- Optimize for Intel/AMD HW
- VMs connected to Internet

  (and thus to each other)
- Can freely communicate with each

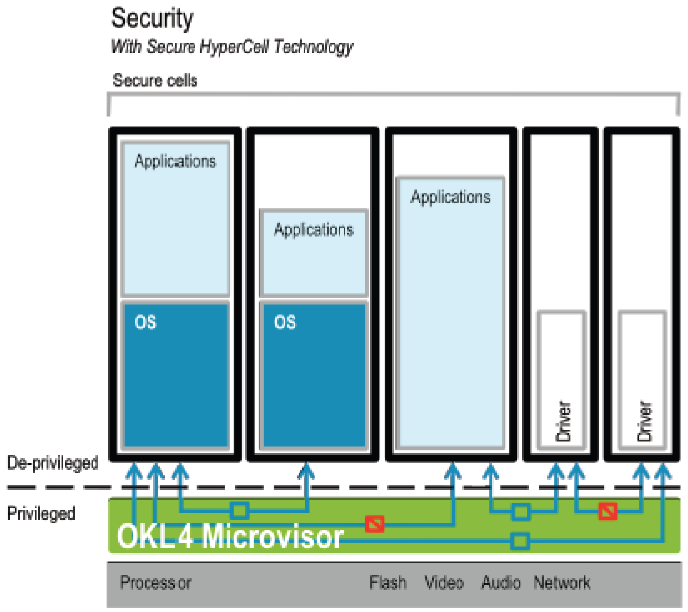
  other and the rest of the world

- Need: Strong protection, yet Highly-efficient (low-latency, high-bandwidth) communication
- This doesn't fit the virtual machine model!
- An embedded hypervisor must provide a more generalized solution to address the specific needs of embedded system applications

Trusted Computing Base

- The **trusted computing base** (TCB) of a computer system is the set of all hardware, firmware, and/or software components that are critical to its security, in the sense that bugs occurring inside the TCB might jeopardize the security properties of the entire system
- The careful design and implementation of a system's trusted computing base is paramount to its overall security

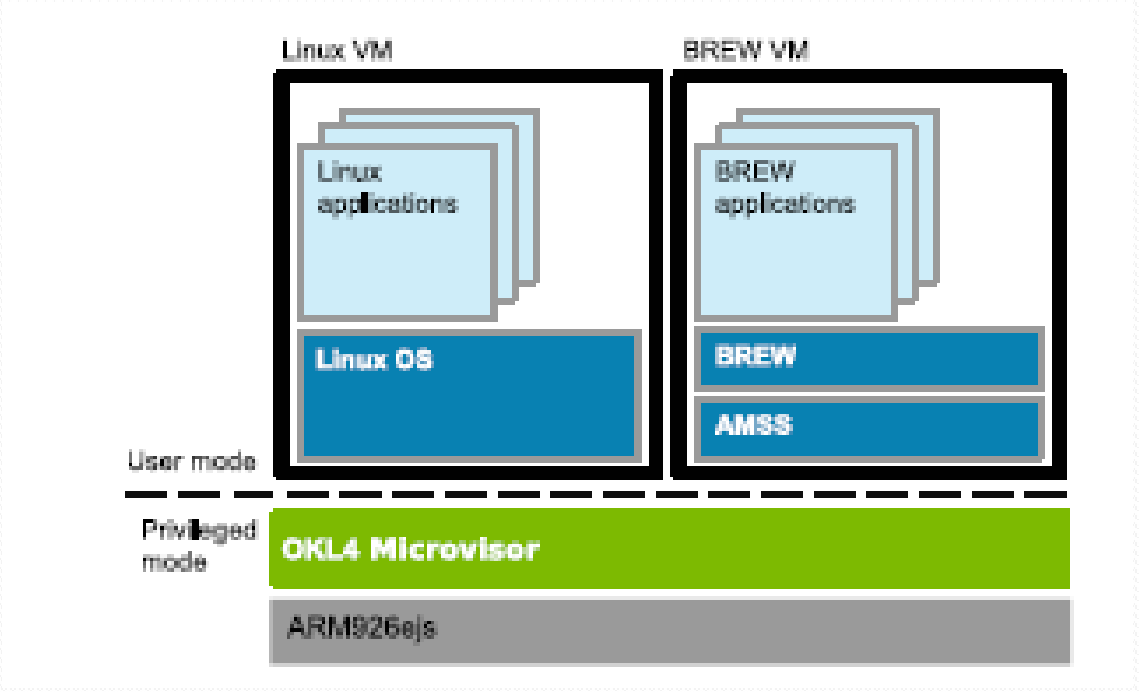
- Identify security-critical apps
- Crypto services
- Run native on OKL4 in their own cell
- Security policy defines allowed communication channels
- TrustZone-like isolation, but more general
- TCB: <15kLOC, much smaller than for a Linux application



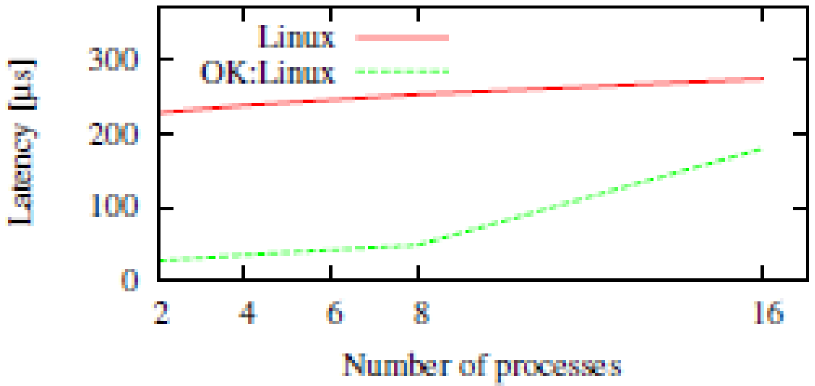
Application – Motorola Evoke QA4

- To meet target price point, single core ARM9 was used
- Linux OS supported the UI
- Baseband stack running outside Linux
- Components from the BREW[QUA] UI framework were reused
    - Porting to Linux was out of the question

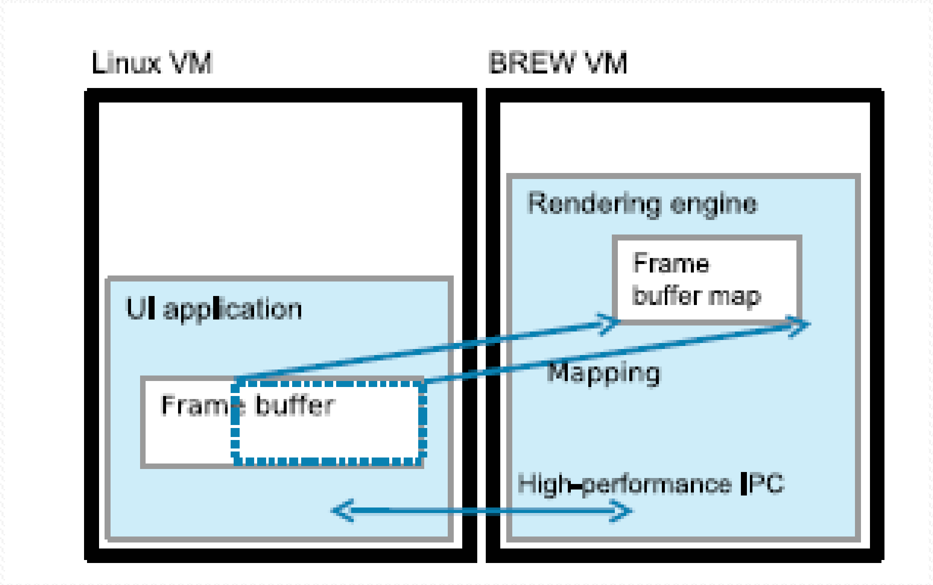
Evoke


Linux VM
BREW VM
Linux applications
BREW applications
Linux OS
BREW
AMSS
User mode
Privileged mode
OKL4 Microvisor
ARM926ejs

Evoke - Context Switching Latencies



Evoke – User I/F Integration



Conclusion

- Virtualization has become a core technology at all levels of computing systems
- Starting out at the “high end” over 35 years ago, VM technology has been applied to many levels of system
- These applications are not a straight “port” of the mainframe technology, but take into account the requirements of each level and type of equipment
- OS Virtualization enables systems of all sizes to run more efficiently and securely
- OS Virtualization also simplifies software development

Resources

- IBM VM
  - http://publib.boulder.ibm.com/infocenter/zvm/v6r1/index.jsp
- VMWARE
  - http://www.vmware.com
- Open Kernel Labs
  - http://www.ok-labs.com