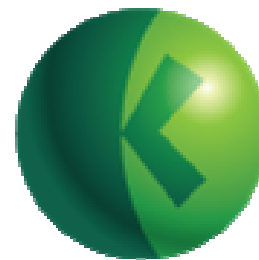


# *Android Migration at the Speed of Light*

**Guest:**

**Rob McCammon**  
VP Product Management  
Open Kernel Labs



**Open  
Kernel  
Labs™**

**Moderator:** Don Dingee, OpenSystems Media

# *Agenda*



**Quick tour of viewer tools**  
**A few topical thoughts**  
**Our guest presentation**  
**Your questions answered**

## *The phone of this Tuesday*

- **App'd up**
- **Open for biz**
- **Safer**
- **Developed**



## *Smarter layers now*

**Inside today's phone ...**

- **Strong app processor**
- **System Software**
- **App execution environment**



**System Software**

App processor and hardware

## Android Migration at the Speed of Light

Migrating to Android on a low cost,  
single core processor

Whitepaper available for download



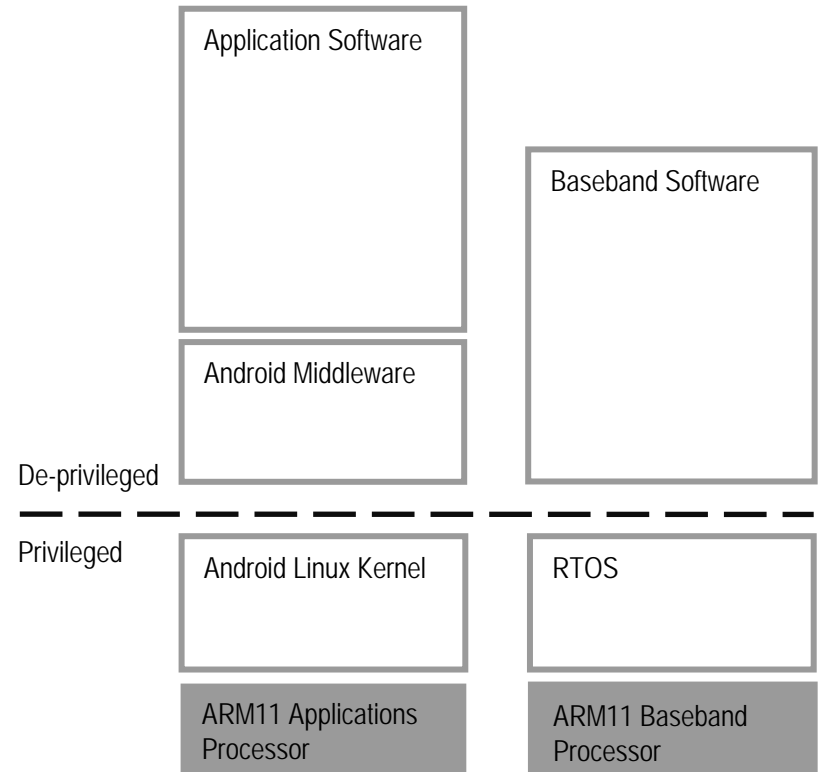
# The Android Revolution

- > **ANDROID**: a truly game-changing platform for mobile devices
  - Complete set of software: operating system, middleware, key applications
  - Completely open for end users and developers
  - Feature-rich software development kit and open App Market
- > Significant opportunity! But significant challenges
  - How to integrate such an all-encompassing platform?
  - How to maintain existing development and technology investment?
  - How to differentiate?



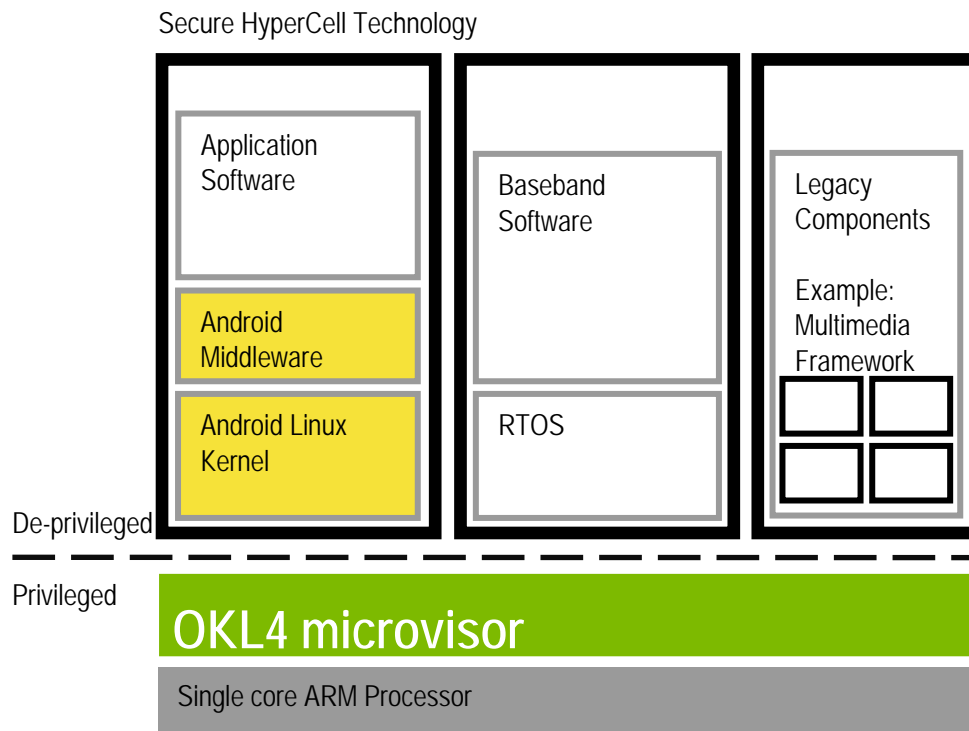
# Android's Architecture

- > Existing Android solutions are deployed on dual-core processors
  - Dedicated communications processor
  - Heavy-weight ARM11 application processor just for Android
- > Example: HTC Dream (G1) – 528 MHz ARM11, 192 MiB RAM



# Android on a Single Core ARM Processor?

- > What if we could achieve the same level of features and performance on a much lower cost single core ARM?
- > And reuse as much of your existing stack investment as possible?



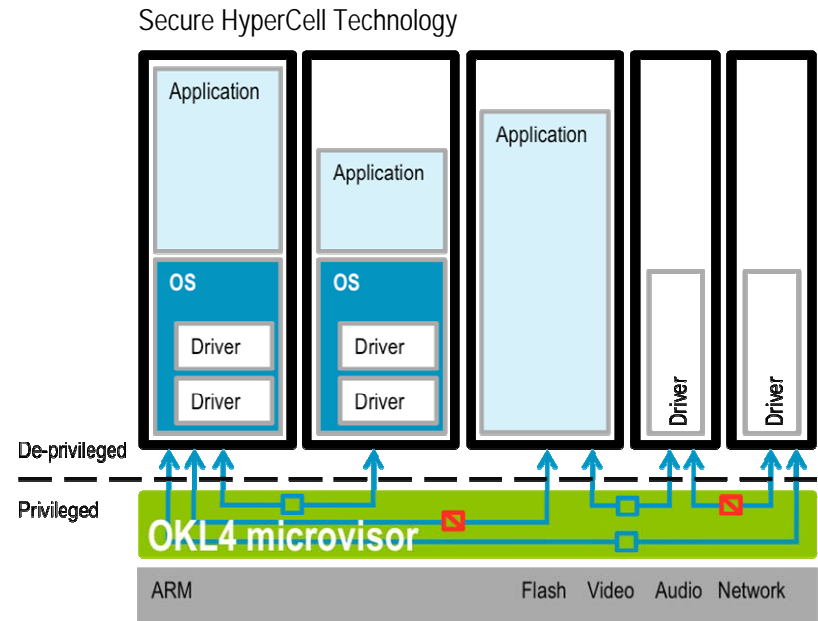
# The Business Case

- > Two key business benefits
- > Lower BOM cost
  - By consolidating application and communications processors
  - And deploying the entire solution on a single core, low cost ARM
- > Dramatically decrease time to market
  - Improve legacy software re-use, maintain investment and expertise
  - Much lower impact on development process – keep the pieces that work for you
- > We've done this already!
  - Commercial success: Motorola Evoke QA4
  - “Smart” phone with two OSeS (Linux and RTOS) on a single ARM core, enabled by OKL4

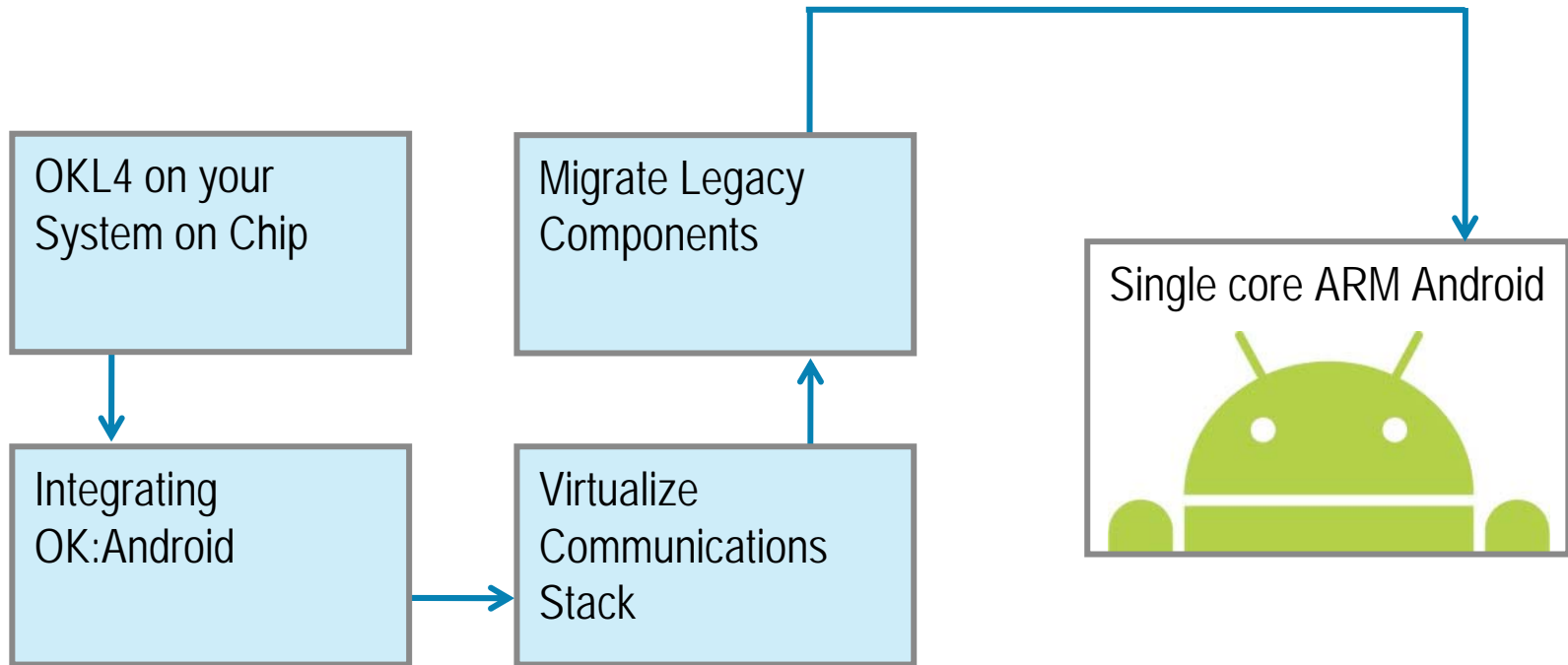


# OKL4 with Secure HyperCell Technology

- > Goes well beyond the classical hypervisor model
- > Enables virtualization and componentization
  - VM = OS plus its applications in a cell
  - Lightweight execution environments
  - Drivers
  - HW enforced isolation between cells
- > Control over communication between cells
  - Required for mandatory access control
- > Fast context switching and high performance inter-cell communication
- > Highly trustworthy privileged code
  - Small, clean, open source

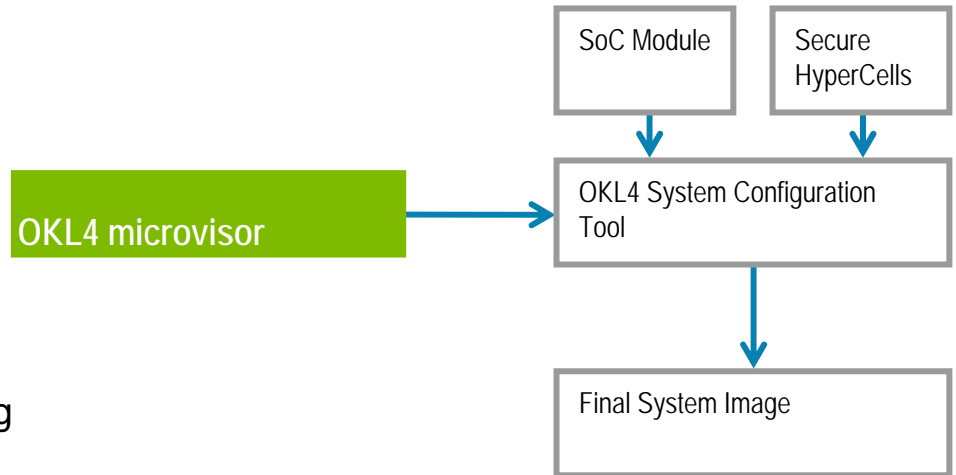


# The Four Steps to Freedom



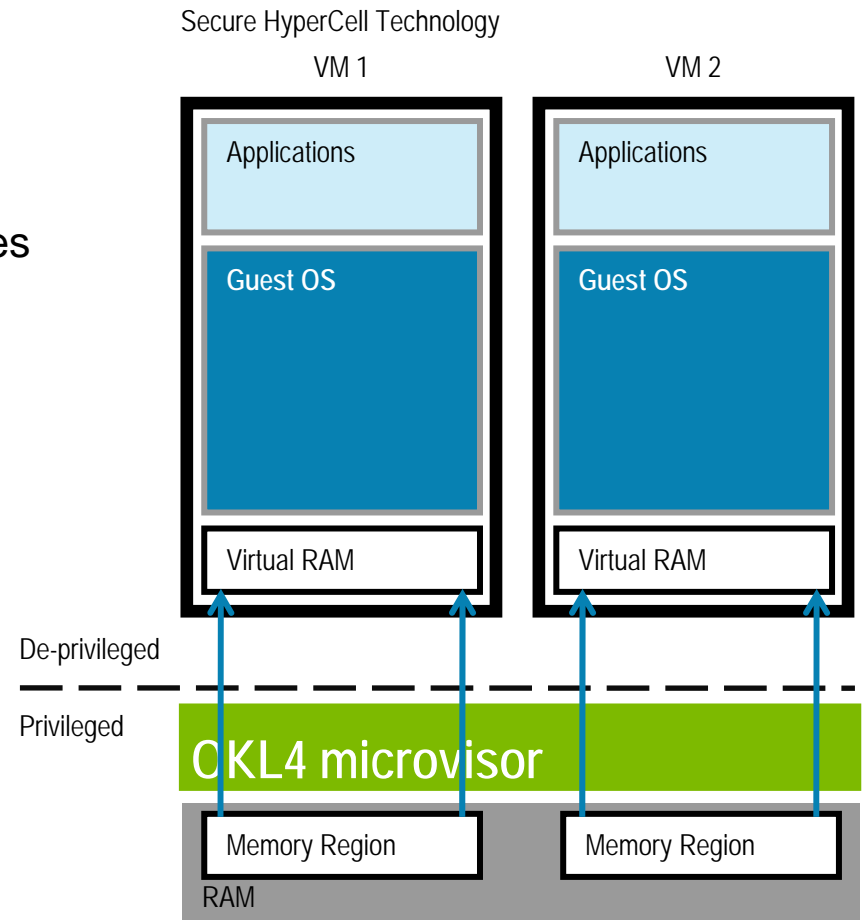
# OKL4 on Your SoC

- > OKL4 System on Chip Software Development Kit (SoC SDK)
  - Completely abstracts SoC implementation from OKL4 kernel
  - Guides SoC developer through process of supporting OKL4 on your hardware
- > SoC Developer implements
  - SoC module startup
  - Interrupt configuration and control
  - Cache operations
  - Timer operations
  - SoC-specific debugging and error handling
- > OKL4 System Configuration Tool
  - Takes kernel object file, implemented SoC module, and any Secure HyperCells and “weaves” into file system image



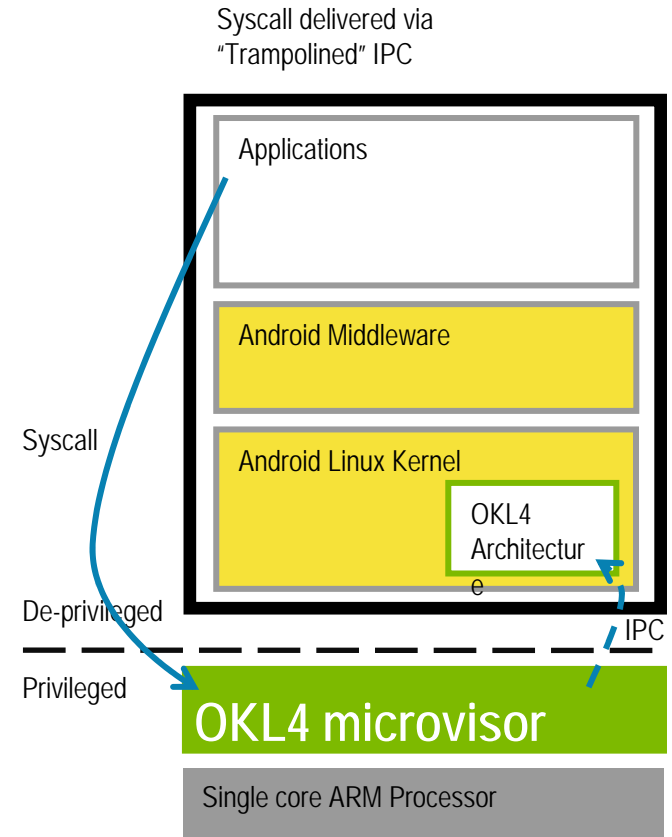
# Virtualizing Android: The role of the microvisor

- > Microvisor partitions and multiplexes hardware between guests
- > Microvisor is in complete control of all resources
  - Completely abstracts SoC implementation from OKL4 kernel
- > Virtual machines access virtual resources
  - Mapped to physical resources by microvisor
- > Guest OS executes at lesser privilege
  - Only microvisor runs in most privileged mode
  - Essential to ensure microvisor has control over resources
  - Guest OS should not run in the most privileged mode
- > Microvisor schedules virtual machines
  - Performs world switch between VMs
  - Each guest OS schedules its apps



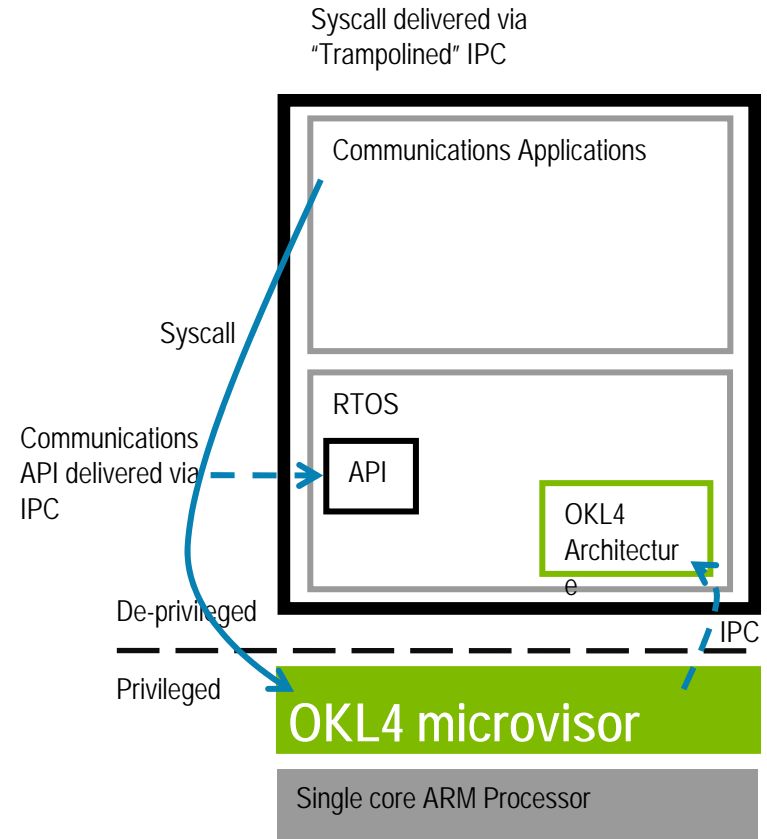
# Virtualizing Android: The paravirtualization approach

- > Minimally-invasive paravirtualization
  - Introduces the OKL4 architecture to the Linux/arch directory (roughly 8 kLOC)
  - No modifications to architecture-independent Linux
- > OKL4 is the only software in privileged mode
  - Receives interrupts, syscalls, and exceptions
  - “Trampolines” to Linux using IPC
  - OKL4 architecture in Linux just decodes IPC
  - Past that point, it’s just standard Linux
- > All Linux applications (including Android middleware) are fully binary compatible
- > OK Linux is distributed as a simple patch against a kernel.org kernel tree
  - Simple to virtualize any Linux distribution



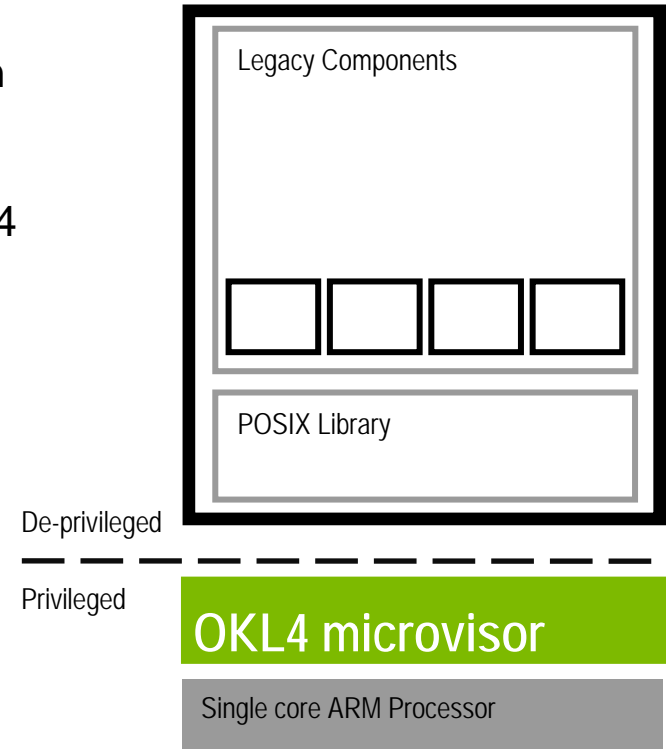
# Virtualizing the Communications Stack

- > Similar paravirtualization approach for communications RTOS
- > OKL4 is fully real-time capable
  - Will maintain all RTOS RT guarantees
  - Example: < 3us interrupt delivery on all platforms
- > Maintain the same API for application stack to use
  - Simply implement a small translation layer using IPC
- > Typical effort: one engineer-week
- > Several RTOSes already exist

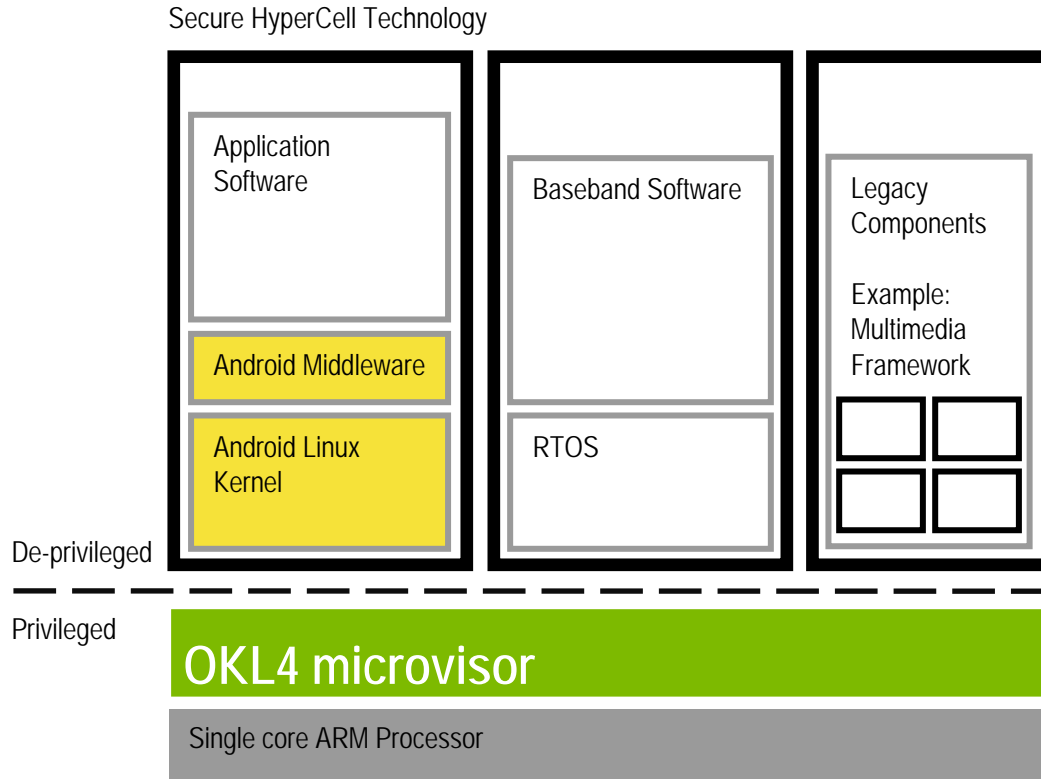


# Componentization and Legacy Migration

- > Secure HyperCell can contain any component
  - Not just a full operating system/virtual machine
  - Unique to OKL4, true fine-grained componentization
- > How?
  - POSIX compatibility library implemented using OKL4 primitives
  - Simply port your application or component
- > Several benefits
  - Support for legacy applications
  - True portability and future-proofness
  - Improved security (e.g. payment apps)
  - Improved reliability
- > Ex: gstreamer port to OKL4 component
  - 1.5 engineer-weeks



# Final Architectural Result



But what about performance?



# OKL4 Performance Benefits

- > The goal: zero-overhead virtualization!
  - All the benefits of virtualization
  - No drawbacks in performance
  - No compromises on security
- > Aggressive optimizations on ARM
  - Fast Address Space Switching (FASS)
  - Compressed Page Tables
  - Super Pages and TLB Sharing
  - IPC Fastpaths
  - Efficient and Secure Shared Memory



# GtkPerf Performance

- > GtkPerf latencies (milliseconds) – smaller is better
- > Setup: arm926ejs, 240MHz, 128MiB RAM, Linux 2.6.24, OKL4 3.0.1

Benchmark	Native (mS)	Virtualized (mS)	% Overhead
GtkEntry	0.04	0.04	0
GtkComboBox	18.03	18.6	3%
GtkComboBoxEntry	14.21	14.57	2%
GtkSpinButton	2.57	2.64	3%
GtkProgressBar	1.15	1.16	1%
GtkToggleButton	4.08	4.23	4%
GtkCheckButton	4.19	4.26	2%
GtkRadioButton	7.74	7.85	1%
GtkTextView - Add text	11.69	11.71	0
GtkTextView - Scroll	6.63	6.35	-4%
GtkDrawingArea - Lines	12.25	11.86	-3%
GtkDrawingArea - Circles	25.64	25.04	-2%
GtkDrawingArea - Text	28.12	27.97	-1%
GtkDrawingArea - Pixbufs	4.24	4.2	-1%
TOTAL	140.58	140.48	0



# Android Migration at the Speed of Light

- > Android is a game-changer
- > Challenge is to go to market as quickly and cheaply as possible
- > OKL4 enables the fastest possible migration to Android
  - While maintaining your existing engineering investments
  - On a differentiating, low-cost, single core ARM platform
  - With absolutely no impact on performance
  - And with significant advantages to your future development uses

Whitepaper available for download



[ok-labs.com](http://ok-labs.com)

© 2009 Open Kernel Labs, Inc. All rights reserved.



# Open Kernel Labs™

Rob McCammon

Vice President, Product Management

robm@ok-labs.com

+1 312 924 1445



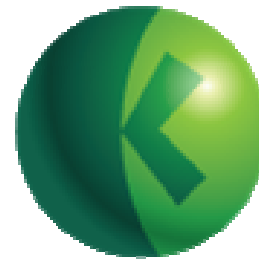
LEARN MORE at [ok-labs.com](http://ok-labs.com)

© 2009 Open Kernel Labs, Inc. All rights reserved.

# *Questions & Answers*

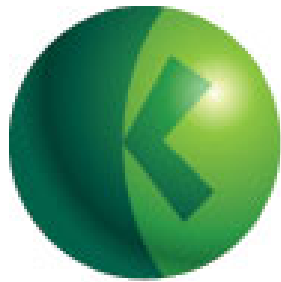
***Guest:***

**Rob McCammon**  
VP Product Management  
Open Kernel Labs



**Open  
Kernel  
Labs™**

*Thanks for joining us*



**Open Kernel Labs™**

*Be open. Be safe.*

Event archive available at:

<http://ecast.opensystemsmedia.com/>

E-mail us at: [clong@opensystemsmedia.com](mailto:clong@opensystemsmedia.com)