

# Streamlining Android Migration with Virtualization

By Rob McCammon, VP Product Management, Open Kernel Labs

In a scant 18 months, the Google Android platform has progressed from "new kid" status to increasing importance in the global mobile marketplace, and beyond mobile too--in other intelligent device categories. Already deployed on a growing fleet of handsets, Android is likely to power two dozen or more shipping phone designs by the end of 2010, and also enjoys a growing audience for applications beyond mobile.

The appeal of the Android platform is twofold - (i) Android offers OEMs a functionally rich, open source mobile OS that powers devices out of the box and hosts third party applications; (ii) The growing portfolio of such applications in the Android Market app store is creating pull for Android-based devices and engendering new business opportunities across the ecosystem. That appeal notwithstanding, targeting device hardware with Android presents developers with a range of challenges, from kernel support for CPUs and SoCs to driver development to performance optimization to integration with other system software.

This article will examine those challenges and highlight how mobile/embedded virtualization can streamline Android support and migration for OEMs and other developers. In particular, it will use the popular Beagle Board together with the OKL4 Microvisor. It will also illustrate how hypervisor technology helps developers build and deploy more robust Android devices and software and services on them, using secure VoIP as an example application.

## Supporting Android on BeagleBoard and Other Hardware

The Beagle Board (<http://beagleboard.org>) enjoys the support of a dynamic developer community for the Beagle Board hardware and software that runs on it. The underlying ARM® Cortex-A8 architecture CPU, the TI OMAP3530 also benefits from strong open source community and commercial support, making the platform an ideal prototyping vehicle for both professional and hobbyist purposes.

Android and Beagle Board represent the intersection of ubiquitous software and hardware technologies, and Android has been ported to Beagle Board on several occasions, with varying results. In particular, the team at Open Kernel Labs (OK Labs) found versions with issues ranging from limited driver support to moderate instability to frequent panics to complete failure to boot. These faults are not necessarily representative of the software and hardware

platforms per se, but are instead endemic to the rapid evolution of and diffuse development efforts around the Android project.

**Virtualization as a Migration Tool** In the enterprise data center and on the desktop, virtualization has become a *jack-of-all-trades*, supporting myriad use cases and applications, including hardware consolidation, load balancing, security and sandboxing, provisioning, multi-OS support, cross development and legacy software migration. In embedded and especially mobile applications, virtualization is also enjoying increasing use, for both development and deployment purposes. My company, OK Labs, provides mobile handset manufacturers (OEMs) with OKL4, our “microvisor”, for applications that in some cases eerily mirror server and desktop virtualization: CPU consolidation, multicore support, secure partitioning and firmware updates, support for multiple OSes – RTOSes and applications OSes like Android, Linux and Symbian, and embedded/mobile platform migration. The introduction of a microvisor (microkernel-based embedded hypervisor) into an existing design must not perturb the particulars of that design. A microvisor should confer flexibility to system designers by isolating new or legacy subsystems but also allowing them to work with existing software components (e.g., Android) just as they would in a native environment. In keeping with this approach, a well-structured microvisor environment should support a simple and straightforward migration process:

### 1. Integration of the Microvisor with Target Hardware

In the case of the Beagle Board with OKL4, the OKL4 Microvisor already supports most common embedded/mobile CPUs, including the TI OMAP3530, easing integration.

### 2. Android Integration with the Microvisor

For enterprise and desktop systems with integrated virtualization support (e.g., Intel VT and AMD-V), restriction of access to critical resources (e.g., interrupt control, MMU and cache configuration) is accomplished through hardware mechanisms. Absent those constructs, Android (or other “guest” OSes) must be “para-virtualized” – modified to let the microvisor intercept and re-implement those operations. To streamline Android support, OK Labs provides an off-the-shelf paravirtualized implementation of Android called OK:Android.

### 3. Device Driver Migration

Applications platforms must, of course, interface with device hardware. In its original mobile handset implementations, Android must drive an LCD display, a touchscreen or keyboard, audio input and output, USB, WiFi and 3G network interfaces, digital camera, an accelerometer, etc. For applications beyond mobile, in digital video, home entertainment, and other areas, drivers must also target HDTV, infrared and radio remote controls, Ethernet, and home automation buses (e.g., X10).

Migrating drivers to a virtualized environment is a question of ROI – there are several options available, each with benefits in line with the level of investments:

- *In situ*: the legacy driver “stays put”, resident in its original

setting inside its host OS (Linux, RTOS, etc). Migration in this case is mostly integration and configuration of the microvisor to optimally support the guest OS environment.

- *In situ virtualized*: existing drivers still remain in their original OS context, with additional configuration to allow other guests to access them via the OKL4 Microvisor virtual device framework.
- *Stand-alone*: legacy drivers are re-hosted in dedicated lightweight virtual machines (secure cells) called device servers and run directly on the microvisor. All guests use virtual devices, governed by microvisor-enforced policies. Stand-alone drivers are thereby the most secure and portable among current and future guest OSes.

### 4. Communications Stacks

Communications stacks (3G, WiFi, etc), most commonly running on a legacy RTOS, can be re-hosted unchanged in the microvisor environment, along with their original host OS. In mobile applications, the capability confers a major time-to-market advantage, by avoiding the need to recertify the code for the network operator or for other regulatory compliance.

### 5. Other Legacy Components

Other legacy components (middleware, utilities, native applications) can migrate to the microvisor in a straightforward manner using OKL4 compatibility libraries, and integration into the final system as a cell. One obvious candidate here would be a custom GUI/application stack, which can be ported unchanged, saving time and also preserving product differentiation features.

## Virtualized Android on Beagle Board – Software Architecture

Following the above procedures, we arrived at an architecture for hosting Android in a virtualized environment on the Beagle Board along with a secure VoIP capability as illustrated in the figure 1 on the next page.

The architecture in figure 1 includes the OKL4 Microvisor running “bare metal” on the Beagle Board OMAP CPU. At start-up, the system boots into OKL4, which then loads and mediates execution of Android and other guest software.

In dedicated secure cells (virtual machines), the microvisor hosts:

- Android, fully isolated from other applications-critical components to ensure that software added to this open OS platform cannot impact stability or security of the whole system. Indeed, should Android crash or lock up, OKL4 can reload/restart it without impact on software running in other secure cells (e.g., Linphone).
- Linphone (or another VoIP stack and application). In our reference design, we ported Linphone to run directly on the OKL4 Microvisor, but it could, resources permitting, reside on an instance of Linux in a secure cell. Our design specifically isolates Linphone from Android to guarantee its stability and security, and hosts on the OKL4 Microvisor to conserve memory and enhance performance.

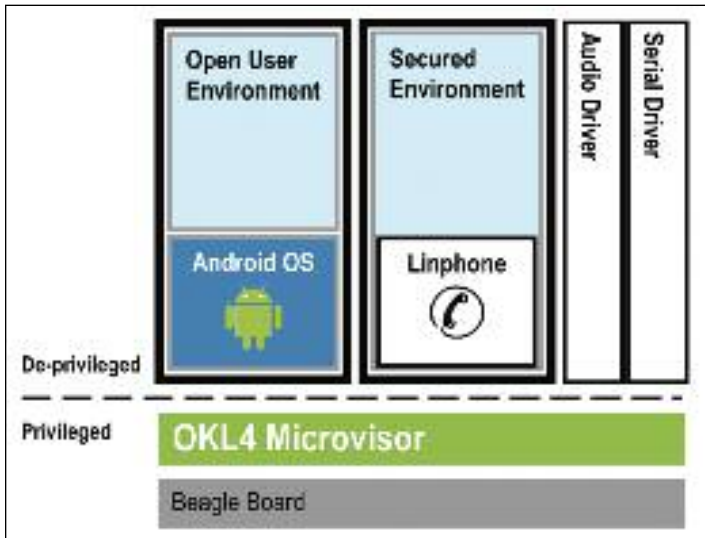


Figure 1: Virtualized Android Architecture on the Beagle Board

- Audio and serial drivers. The audio driver is shared by Android and Linphone, and provides audio input/output for VoIP calls even if Android is disabled. The serial driver is isolated and shared to support system console and debugging.

### The Running System

The system functions as follows:

- Android appears to the user as the primary OS with the standard Android user interface presented via OMAP display hardware resident on the Beagle Board.
- The user initiates a secure VoIP call using an Android-based VoIP “dialer” application or through a Linphone console server running under Android.
- Once the call is initiated, the user can run other Android applications (games, web browsing, etc.) without impacting the quality or security of the VoIP call. Android could sleep or even hang and the call would continue without interruption.

### Virtualization and Performance Optimization

Embedded developers have always distrusted software that “gets in the way”. Early in the history of intelligent devices, hardware-oriented developers would side-step high-level languages in favor of assembly code, hoping to squeeze more performance out of “bare metal”. Later, the perceived bottleneck was the RTOS. And more recently, application OSes like Linux and Android inspire the displeasure of performance-hungry engineers, product managers and end-users.

Mobile/embedded virtualization presents yet another layer of abstraction. It enhances productivity but also pushes developers and their code another step away from underlying hardware and presumed performance advantages of bare metal execution.

Perhaps surprisingly, in many applications, virtualization not only induces minimal performance overhead, it can also enhance throughput and responsiveness, principally through better memory utilization and by leveraging microvisor IPCs. The

following table illustrates results for the GTKPerf benchmark suite for the popular GTK+ UI framework. GTKPerf provides a common testing platform to run predefined GTK+ widgets (opening comboboxes, toggling buttons, scrolling text yms.) and measure the speed of a device or platform. Learn more at <http://gtkperf.sourceforge.net/>

We built the following table by running GTKPerf to show actual timings and overhead for both native and virtualized environments.

Benchmark - GTKPerf (milliseconds)	Native	Virtualized	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%
GtkCheckBox	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%
Average Overhead			0 %

Table 1: GTK Performance for Native and Virtualized Execution <sup>1</sup>

You should expect comparable results for Android graphics and other subsystems in the platform as well.

Let’s examine some further examples of microvisor implementation that actually enhance the performance of guest OSes like Android, especially on lower-cost chipsets in the ARM9 family.

### Fast Address Space Switching

A well-engineered microvisor like OKL4, independently of the guest OSes it hosts, can deliver fast, low-latency context switches among different guests and guest processes. Android software architecture builds on isolated components communicating via custom IPC mechanisms. This architecture generates more inter-process communication than in traditional Linux-based environments, where a full-cache flush is required on each inter-process context switch. If context switches in regular Linux induce latency and memory bus usage, imagine the situation with Android.

To improve Android performance, OKL4 maintains cache state across these context switches, leading to both improved context

<sup>1</sup> ARM926ejs @ 240 MHz with 128MB RAM running Linux 2.6.24 and OKL4 3.0.1

switch latency and better overall performance through improved cache utilization.

**Minimizing Memory Footprint**

The OKL4 Microvisor is optimized for performance and memory usage. It is difficult to directly compare the memory size of full stacks like Android and Linux, and microkernel-based environments like the OKL4 Microvisor. Android and Linux obviously provide a different range of services.

However, it is worth noting that the OKL4 Microkernel itself occupies a few tens of KBytes, while the Android and Linux kernel are measured in megabytes. The inherently small size of OKL4 does not by itself minimize total system footprint, but when a small base is combined with a selection of additional components, the total integration results in a lighter-weight paravirtualized OK:Android. When compared with enterprise and desktop hypervisors like Xen and KVM, which include entire Linux kernels, OKL4 is tiny indeed.

**Extending the Architecture and Its Applications**

The system architecture described in this article is intended to illustrate the simplicity and utility of running Android and accompanying software in a virtualized environment. The same structure could equally support other guest OSes in place of, or in addition to, Android. To facilitate OS integration, OK Labs offers an

off-the-shelf para-virtualized version of popular mobile/embedded OSes, including OK:Linux and OK:Symbian, and tools and services to assist developers in para-virtualizing almost any OS with available source code.


Moreover, this type of design is not limited mobile applications or to the Beagle Board. To readily enable a wide range of mobile devices, OKL4 supports the ARMv5, ARMv6, and ARMv7 architectures. This broad support streamlines migration of Android and other supported OSes to other CPUs and SoCs and to device types beyond the Beagle Board development environment and beyond mobile applications where OKL4 today ships on over 500 million devices.

Using the same virtualization architecture, OKL4 can easily support high-level OSes like Android, RTOSes and purpose-built platforms on netbooks, webpads, set-top boxes and DVRs, HDTVs, in-car systems, medical equipment – indeed any ARM-based device.

**Conclusion**

By designing with virtualization, OEMs and developers both streamline migration and integration, and benefit from the stability and security of strong isolation among software components. On the Beagle Board, the OKL4 Microvisor provides a short path to stable and robust hosting of the popular Android platform. Imagine what it can do for your next design.


**END**




What does our 35 years of experience bring to YOU?

- Out of the box solutions
- Extensive product line
- Reliable, proven code
- Free evaluation kits
- No royalties
- Full source code

SMX <sup>®</sup> RTOS	WiFi	USB Device
BSPs	TCP/IP	USB Host
Device Drivers	FAT File System	USB OTG
Kernel Awareness	Flash File System	USB Class Drivers
Simulator	GUI	Floating Point



800.368.2491 sales@smxrto.com



BOOTH # R36

ARM7 • ARM9 • ARM11 • Cortex

### ARMs Supported

<p><b>Atmel</b></p> <ul style="list-style-type: none"> <li>• AT91x406xx</li> <li>• AT91M55800</li> <li>• AT91CAP9</li> <li>• AT91RM9200</li> <li>• AT91SAM3U</li> <li>• AT91SAM7A1/2/3</li> <li>• AT91SAM7S</li> <li>• AT91SAM7SE</li> <li>• AT91SAM7X</li> <li>• AT91SAM9260/1/3</li> <li>• AT91SAM9G20</li> <li>• AT91SAM9M10/G45</li> <li>• AT91SAM9RL64</li> <li>• AT91SAM9XE</li> </ul> <p><b>Cirrus Logic</b></p> <ul style="list-style-type: none"> <li>• EP93xx</li> </ul>	<p><b>Freescale</b></p> <ul style="list-style-type: none"> <li>• i.MX1/i.MXL</li> <li>• i.MX31</li> <li>• MAC71xx</li> </ul> <p><b>NXP</b></p> <ul style="list-style-type: none"> <li>• LH754xx</li> <li>• LH79520/4/5</li> <li>• LH7A400/4</li> <li>• LPC17xx</li> <li>• LPC21xx</li> <li>• LPC22xx</li> <li>• LPC23xx</li> <li>• LPC24xx</li> <li>• LPC2858</li> <li>• LPC29xx</li> <li>• LPC313x/4x/5x</li> </ul>	<ul style="list-style-type: none"> <li>• LPC3180</li> <li>• LPC32xx</li> </ul> <p><b>Samsung</b></p> <ul style="list-style-type: none"> <li>• S3C2410/40</li> <li>• S3C2443</li> </ul> <p><b>STMicro</b></p> <ul style="list-style-type: none"> <li>• STM32F101/2/3</li> <li>• STM32F105/7</li> <li>• STR71x</li> <li>• STR75x</li> <li>• STR91x</li> </ul> <p><b>TI</b></p> <ul style="list-style-type: none"> <li>• LM3S3xxx/5xxx</li> <li>• LM3S8xxx</li> <li>• LM3S9xxx</li> <li>• TM5470</li> </ul>
--	--	--

Free Evaluation Kits: [www.smxrtos.com/eval](http://www.smxrtos.com/eval)  
Free Demos: [www.smxrtos.com/demo](http://www.smxrtos.com/demo)

www.smxrtos.com/processors