



# Comparing Dual OS Technologies

## One OS at a time versus simultaneous OS use

### Executive Summary

This document explores different techniques to run two or more operating systems (OS) on a single device and uses Windows and Android to provide the specific context for the discussion. These techniques are generally applicable for running Windows with any other Linux variant such as WebOS or Chromium, or different combinations of Linux variants together. The approaches compared include: dual boot, hibernate and restore, OS API virtualization, Type-1 virtualization, hosted virtualization, embedded virtualization and emulation.

Some techniques such as OS level virtualization can be used for running combinations of Linux based OS's, e.g., BlackBerry Playbook and Android. However, they cannot be used when Windows is in the mix.

While it is possible to compare the different techniques based on performance metrics for throughput, disk performance, context switch times, etc., the comparison provided here is more qualitative and focuses on what is required to give the end-user an integrated experience. If the comparison seems subjective, the rationale and use cases for a Dual OS system is very different from that for running a large number of virtual machines in the data center- the two or three OS should co-exist on the same device and appear seamless to the end-user.

### Overview

A brief description of the different techniques appears below:

#### Dual Boot

With this technique, the end-user chooses which OS to boot up with when the device boots up. In order to switch between OS's the end-user has to shutdown the OS that is currently running and boot up the next OS. Usually this requires a long period of time, from tens of seconds up to a minute. Dual boot systems have been popular for a long time. SplashTop from DeviceVM is an example of commercially available dual boot systems.

#### Hibernate and Restore

Phoenix introduced this concept in the HyperSpace product, which was purchased by HP. This technique is similar to dual boot but offers faster switch time. One OS is hibernated by the BIOS and the other one is restored. Hibernation is faster than boot, and the speed of Hibernate and Restore depends on the memory on the system and the speed of the media to which the state is being written and read from.

Also for some devices such as GPU, the graphics card remembers the system state. The Hibernate and Restore solution should be able to back up each graphics card and restore it, and



## Comparing Dual OS Technologies

the drivers in both the OS's must support this requirement. Persisting the state across a Hibernate and Restore can be a very complex problem and is impossible to do generically for any graphics card.

### OS API Virtualization

If both the operating systems are based on Linux, e.g., Meego and Android or QNX, or Blackberry and Android, then one can take advantage of OS API level virtualization. Different application stacks run on top of the same base kernel but in different virtual containers or *chrooted* environments. Always innovating offers a solution based on this approach. Such approaches were pioneered for virtual hosting by Ensim and Plesk virtual servers (now part of Parallels).

### Type-I Virtualization

Citrix XenClient was an example of the Type-I hypervisor, which has direct control of the hardware and each operating system runs as a guest above it. It was offered on a couple of devices: one from Dell and another from HP. Since most Type-I hypervisors and the dom-0 or root partitions are based on variants of Linux, drivers for underlying hardware are hard to find, limiting their commercial viability. It seems that both VMware and Citrix have discontinued their Type-I efforts.

### Type-II or Hosted Virtualization

Parallels, VMware Fusion and Virtual Box are the three commercially available hosted virtualization products that run an operating system on top of a base or host OS. The advantage of hosted virtualization is that these products don't need to do anything from a driver perspective and are wildly popular.

### Emulation

Some of the development environments like the Google Android Emulator emulate ARM instructions on x86 platforms. Most of these are based or derived from open source QEMU work. They are very slow and not a practical commercial solution.

### Embedded Virtualization

We use this term to refer to the concept that virtualization is embedded into the system and completely transparent to the user. The user should not be aware that virtualization is in use to permit simultaneous operation of Dual OS's on the same hardware. The solution should require zero configuration, all the devices should be automatically shared, there should be a single, common control point for all devices, switching from one OS-specific application to an application for the other OS should be instantaneous and seamless.



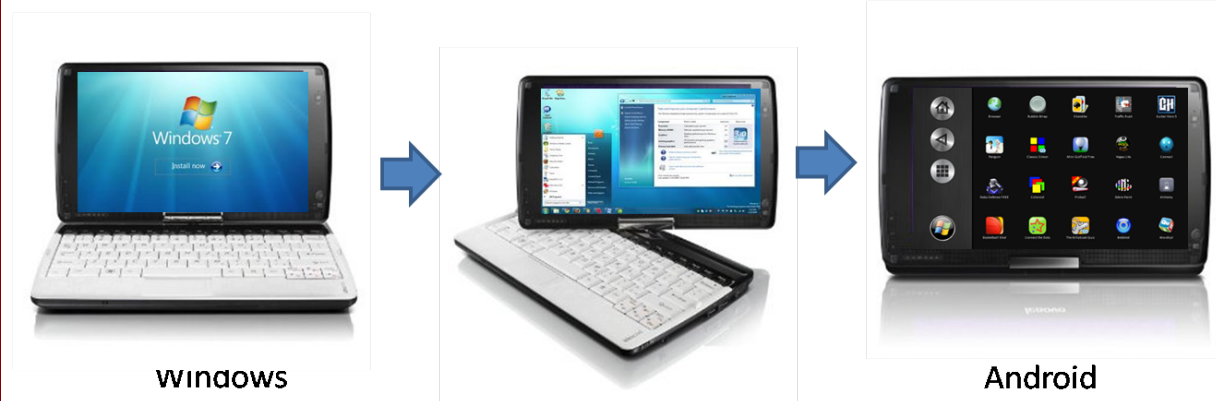
## Comparing Dual OS Technologies

### Comparison Criteria

The primary focus of this note for the dual OS use case is Android together with Windows. The end-users of the dual OS are assumed to be average consumers who have no technology savvy. Ease of use, intuitiveness, and integrated experience are more important than raw speeds and feeds. For example, traditional benchmarks of how many virtual machines can be run by the dual OS are neither meaningful nor relevant.

#### Switch time

How quickly can the user switch from one operating system to another? The use case of this is demonstrated in the picture below where a user flips from one orientation to another. Can the OS switch appear instantaneous to the user, in less than a couple of seconds?

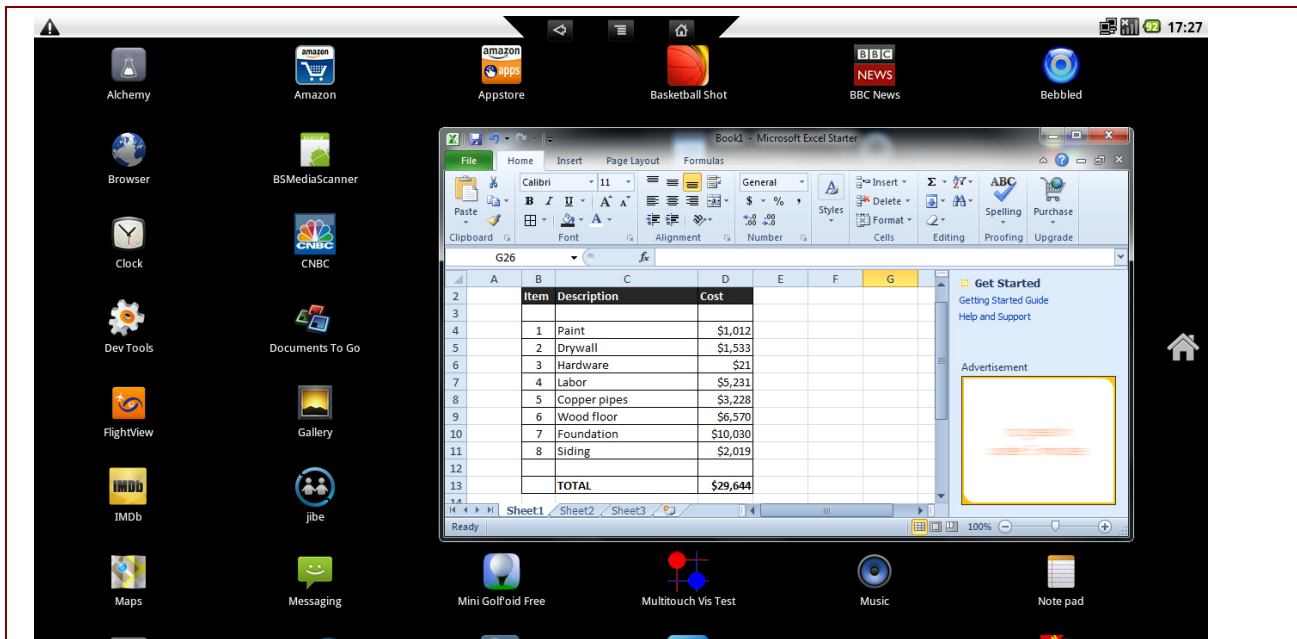


#### Simultaneously active apps

Does a user care whether an app is a Windows or Android app? The user just sees cons on the Windows desktop or on the Android grid. Can the user run these apps simultaneously? For example, when the device is docked and connected to a large display, can the user interact with Windows on the large screen using a keyboard and the device runs Android? Or can two applications, e.g., Microsoft Excel running alongside Android apps, run simultaneously with overlapping windows?



## Comparing Dual OS Technologies



### Data Sharing (Encrypted/Un-Encrypted)

Can the user data be shared seamlessly between Android and Windows? For example can the Pictures folder in Windows be viewed in the Android Gallery App and vice-versa? Does the user have to do this manually or is it automatically setup?

Also if Windows uses an encrypted file system can that data be shared with Android?

There are other subtleties with Data Sharing. For example in Hibernation & Restore, if one OS is hibernated, the file system data structures on disk may not be consistent and thus reading them from another OS may result in intermittent failure.

### Additional Codecs Required

A device manufacturer has to license codecs for each device that is shipped. Audio/Video codecs can add up to \$ 4-5 per device. In a dual OS situation do you need codecs for each OS or can one OS re-use the codecs from the other?

### Need Hardware Support for Virtualization

Type-1 virtualization solutions often require hardware support in terms of VT-X or AMD-V in the processor. While this requirement is ok for servers, on the client side several chipsets like Intel Atom or AMD Brazos don't have this capability. Both of these are most commonly deployed in tablet or netbook computers on the lower end of the market.

### Linux Device Drivers



## Comparing Dual OS Technologies

Does the solution require Linux device drivers for the hardware? This is true for both Dual Boot solutions and also Type-I hypervisors like Xen, where the root partition is based on Linux.

For client devices this is a major challenge and effectively reduces the commercial viability of the solution, as such drivers are not available for the multitude of hardware components in the market.

### Tech Savvy User

Virtualization still remains the domain of the technology aware person. Running two or more operating systems on the same machines introduces more complexity than an average user can handle. The level of knowledge and sophistication the solution requires greatly restricts what it can be used for. For consumer devices the sophistication can't be more than point and click.

### Battery Life

This is a difficult one to define. What is the comparison basis? If the device is running both Windows and Android, should the battery life of the device for running Windows alone be compared with the battery life for Android alone? What are the perception and/or expectation of the end user buying the device? If a device manufacturer sells a Windows device with added capability to run Android is the expectation different than if the device is sold as an Android device with the ability to run Windows.

Also note that battery life is applicable mostly for mobile devices, and not really meaningful for desktops or all in one (AIO), which are mostly plugged in.

### Overall Performance

If Android is running on top of Windows using a Type-II technology, a question you will get asked is, "What is the performance compared to running Android natively on the device?" How much overhead does adding a virtualization layer impose? While this is a fair question, if the user just wanted an Android only device they could have bought something cheaper and different. So the fundamental assumption is that the user wants to run both Windows and Android and values the ability to switch between them seamlessly. With that if you can get acceptable performance, it is great.

The table below summarizes the pros and cons of using different techniques for running a dual OS solution.



## Comparing Dual OS Technologies

	Dual Boot	Hibernate & Restore	OS API	Type-I	Hosted	Emulation	hyperDroid
Switch Time	30-60s	10-20s depends on machine	< 2 s	< 5s	< 5s	> 20 s	< 5 s
Simultaneous Use	No	No	Yes	Yes	Yes	Yes	Yes
Data Sharing	Limited requires NTFS codec, no encryption support		Yes	Limited, same as Dual Boot	Yes	Yes	Yes
Encrypted Data Sharing	No	No	Yes	No	Yes	Yes	Yes
Commercially Viable				Only for reference hardware		No. Its painfully slow	Yes
Windows can be one of the OS	Yes	Yes	No	Yes	Yes	Yes	Yes
Need H/W Virt. Support	No	No	No	Yes	No	No	No
Tech Savvy User	Yes	Yes	No	Yes	Yes	Yes	No
Require Linux Driver Support	Yes	Yes	No	Yes	No	No	No
Require codec for each OS	Yes	Yes	Yes	Yes	Yes	Yes	No
Battery Life	Same as native OS	Same as native OS	Same as native OS	--	Same as host OS	--	Same as host OS
Performance	Native	Native	Native	15-25% below native	25% below native with guest tools	Painfully Slow	Close to Native



## Comparing Dual OS Technologies

The red boxes are showstoppers for any of these approaches.

For Dual Boot systems, switching between the two operating systems take so much time that it is not practical for the user to use it.

The Hibernate and Restore approach improves on the switching time, but has a significant drawback in that it does not offer the ability to run apps simultaneously. An even bigger drawback is the need to have device drivers for each of the operating systems. While Microsoft is very rich in providing driver support, Linux based OS tend to be lacking. Also it introduces large amounts of work to develop, test and QA such a solution. So while one can make dedicated devices, using this broadly across different SKUs can be very challenging.

While the OS API virtualization is great on all counts for Linux-only solutions, it is not applicable for running Android & Windows or WebOS & Windows together.

Type-I hypervisor requires hardware virtualization support and is therefore not applicable for the tablet and netbook space. On the high end it may be a viable option, but suffers from the same draw back of requiring drivers in a Linux based root partition.

Type-II or hosted virtualization is a viable option for a dual OS solution. The only drawback is that it may run the virtualized OS slower than if run natively. This can be greatly improved with guest tools as demonstrate by use of windows using VMware Fusion or Parallels on Macs.

Emulation is not really a commercially viable solution because it is painfully slow. Its use is constrained mostly to dev. environments.

hyperDroid, the custom hypervisor from BlueStacks, has all the benefits of a hosted virtualization solution, without the performance drawbacks.

**Bluestack Systems, Inc.**  
**2105 South Bascom Avenue, Suite 380**  
**Campbell, CA 95008-3278**