

# Linuxfest Northwest 2015

## The Bare-Metal Hypervisor as a Platform for Innovation

By Russell Pavlicek

Xen Project Evangelist

[Russell.Pavlicek@XenProject.org](mailto:Russell.Pavlicek@XenProject.org)

@RCPavlicek



# About the Old, Fat Geek Up Front

- Linux user since 1995; became a Linux advocate immediately
- Delivered many early talks on Open Source Advocacy
- Former Open Source columnist for Infoworld, Processor magazines
- Former weekly panelist on “The Linux Show”
- Wrote one of the first books on Open Source: Embracing Insanity: Open Source Software Development
- 30 years in the industry; 20+ years in software services consulting
- Currently Evangelist for the Xen Project (employed by Citrix)
- Over 75 FOSS talks delivered; over 150 FOSS pieces published



# About Innovation...

- A favorite buzzword for marketing purposes
- Many things in our industry labeled “Innovation” are nothing more than hackneyed placid tripe
- Innovation calls for thinking of the world in a different way and seeing it come to life
- Simply changing the shade of lipstick on a pig does not qualify



# About Innovation...

- Real innovation can borrow from the known to create the unknown
- Many innovations are reassemblies of known objects in a new way
  - Example: many cloud concepts resemble similar concepts in mainframes, but they've been reapplied to a multi-server environment
  - But the net result needs to be something significantly different than what existed before



# Some of the More Interesting Advances

- Xen Automotive: the effort to craft an embedded automotive infotainment system
- Realtime virtualization: work to facilitate applications which need realtime processing
- ARM-based hypervisor: enabling a new breed of applications, from servers to cell phones, on the ARM architecture
- Mirage OS and other unikernel systems: creating highly-dense farms of ultra-small and secure cloud appliances



# But First...

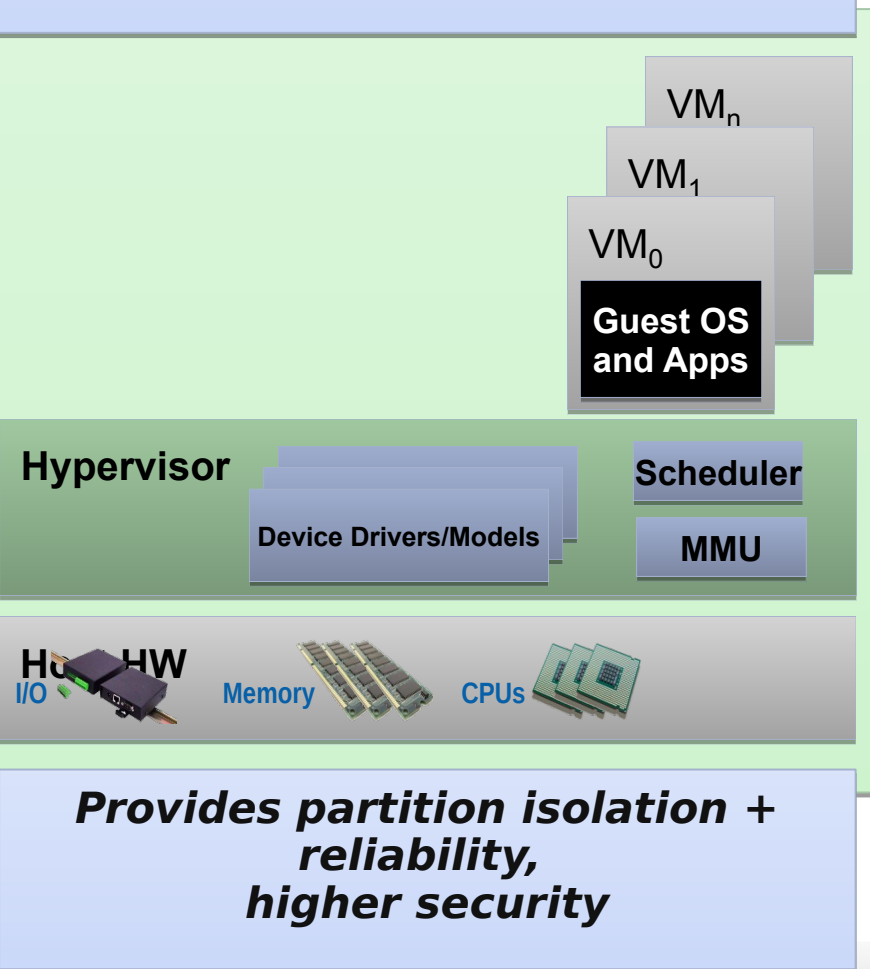
What exactly is a “Bare-Metal Hypervisor”?



# Hypervisor Architectures

## Type 1: Bare metal Hypervisor

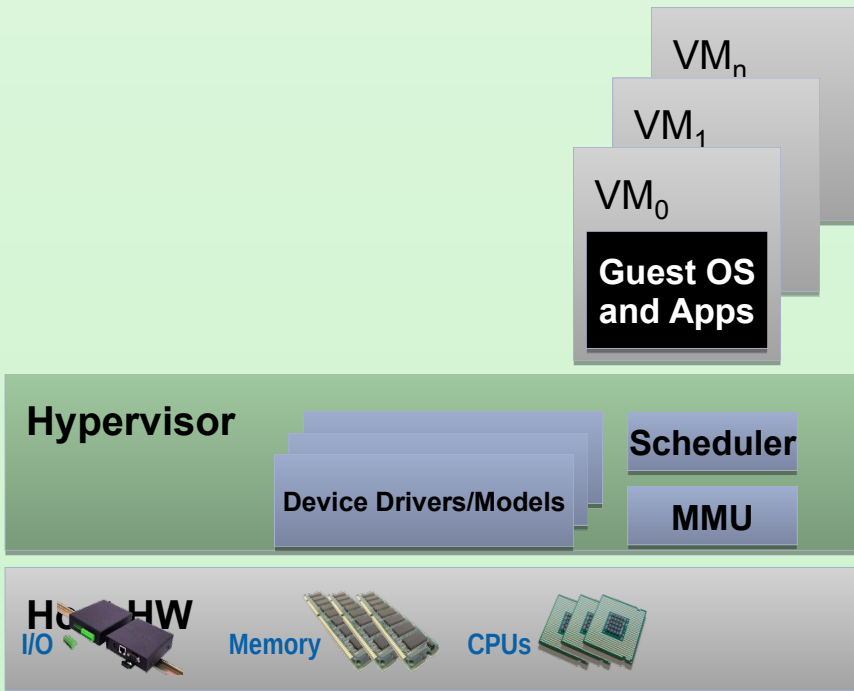
A pure Hypervisor that runs directly on the hardware and hosts Guest OS's.



# Hypervisor Architectures

## Type 1: Bare metal Hypervisor

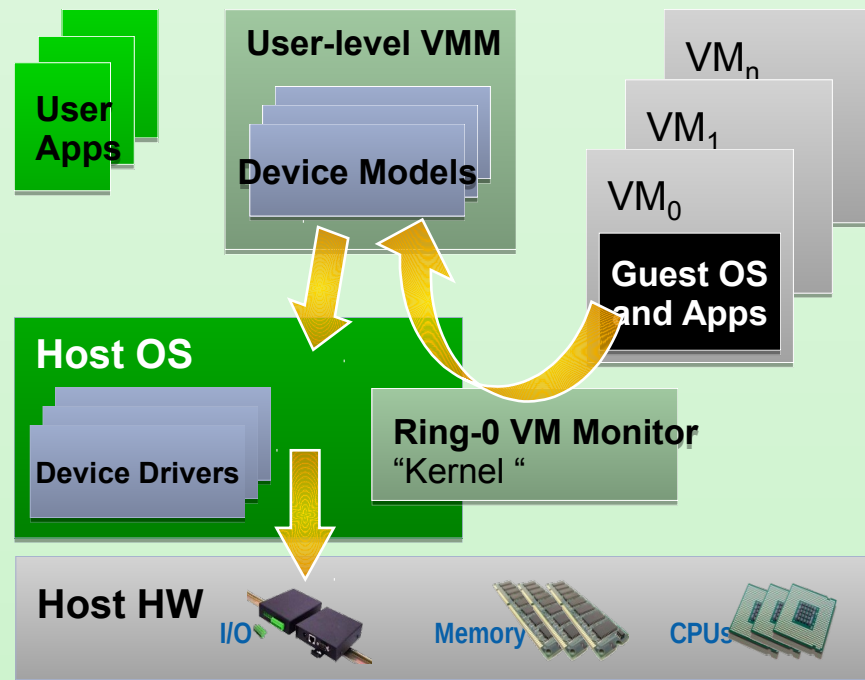
A pure Hypervisor that runs directly on the hardware and hosts Guest OS's.



***Provides partition isolation + reliability, higher security***

## Type 2: OS 'Hosted'

A Hypervisor that runs within a Host OS and hosts Guest OS's inside of it, using the host OS services to provide the virtual environment.

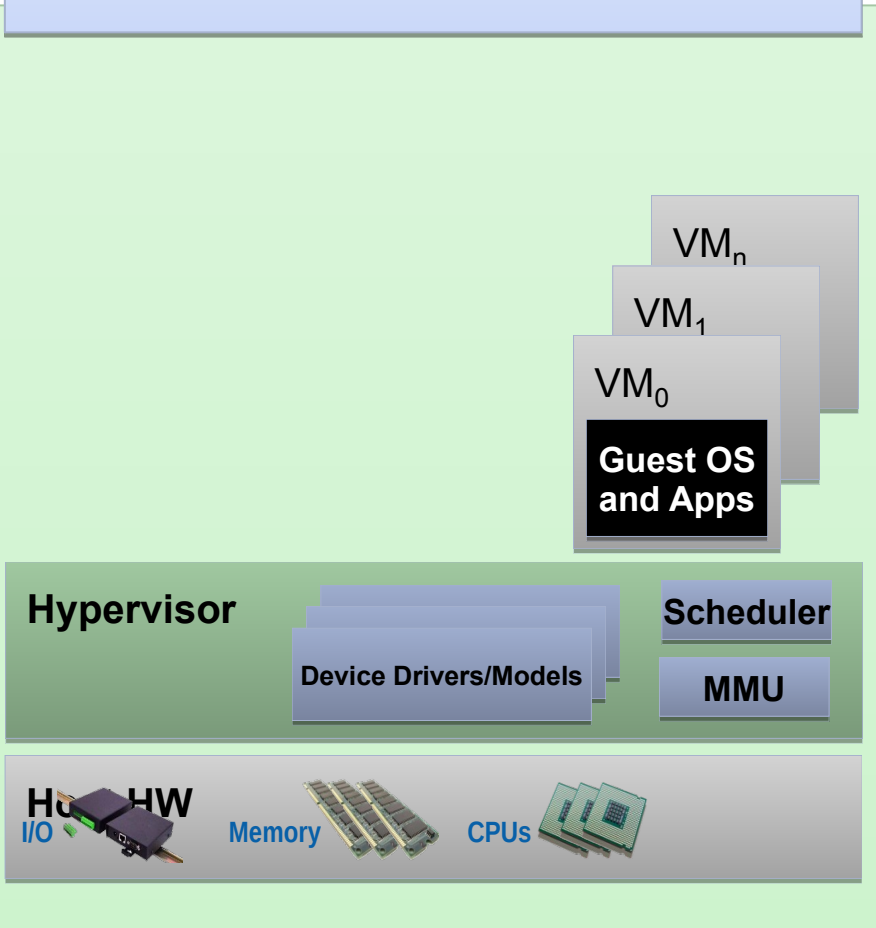


***Low cost, no additional drivers  
Ease of use & installation***



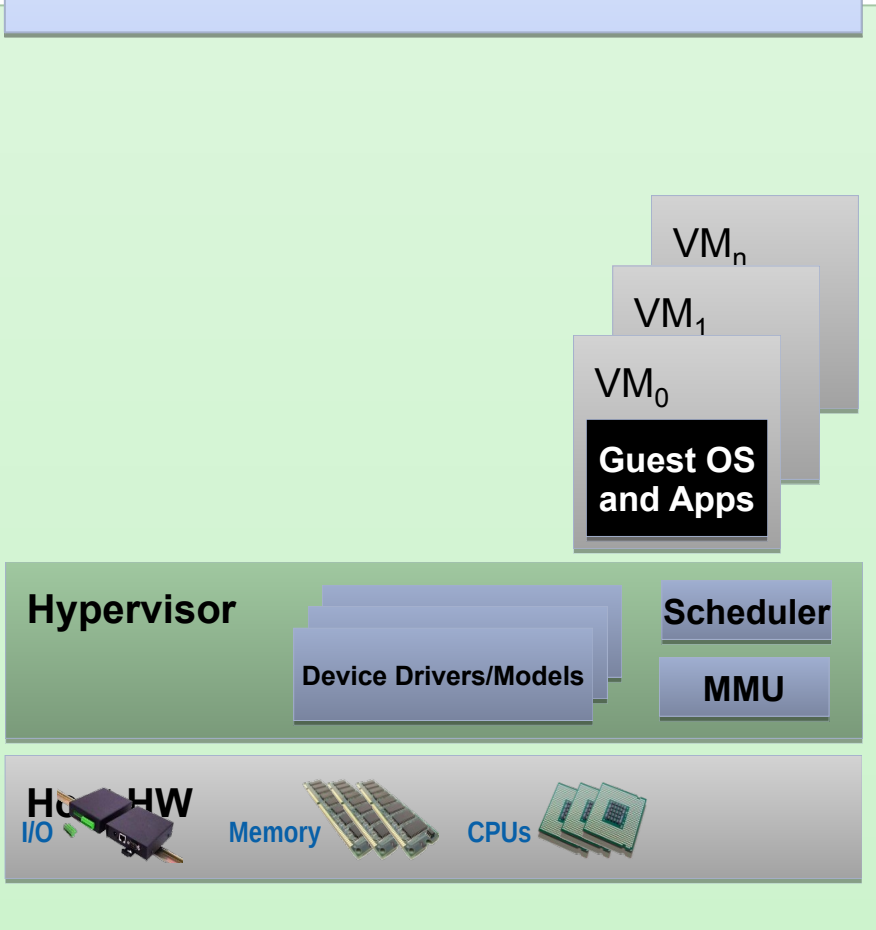
# Xen Project: Type 1 with a Twist

## Type 1: Bare metal Hypervisor

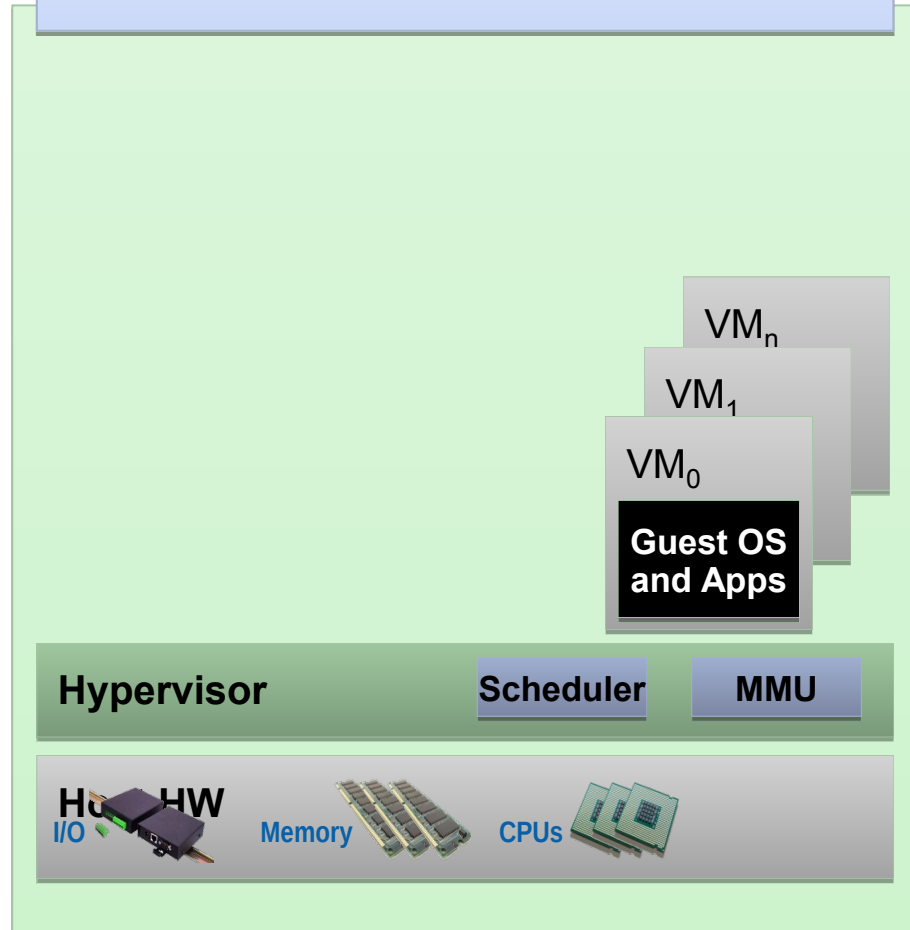


# Xen Project: Type 1 with a Twist

## Type 1: Bare metal Hypervisor

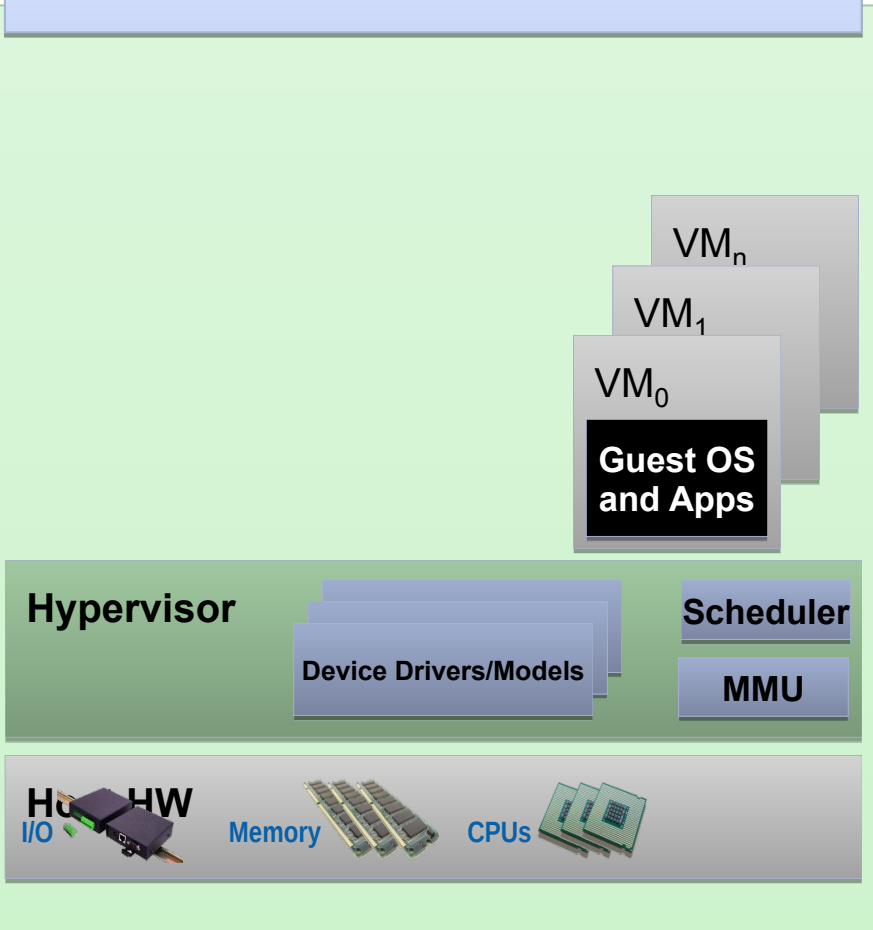


## Xen Project Architecture

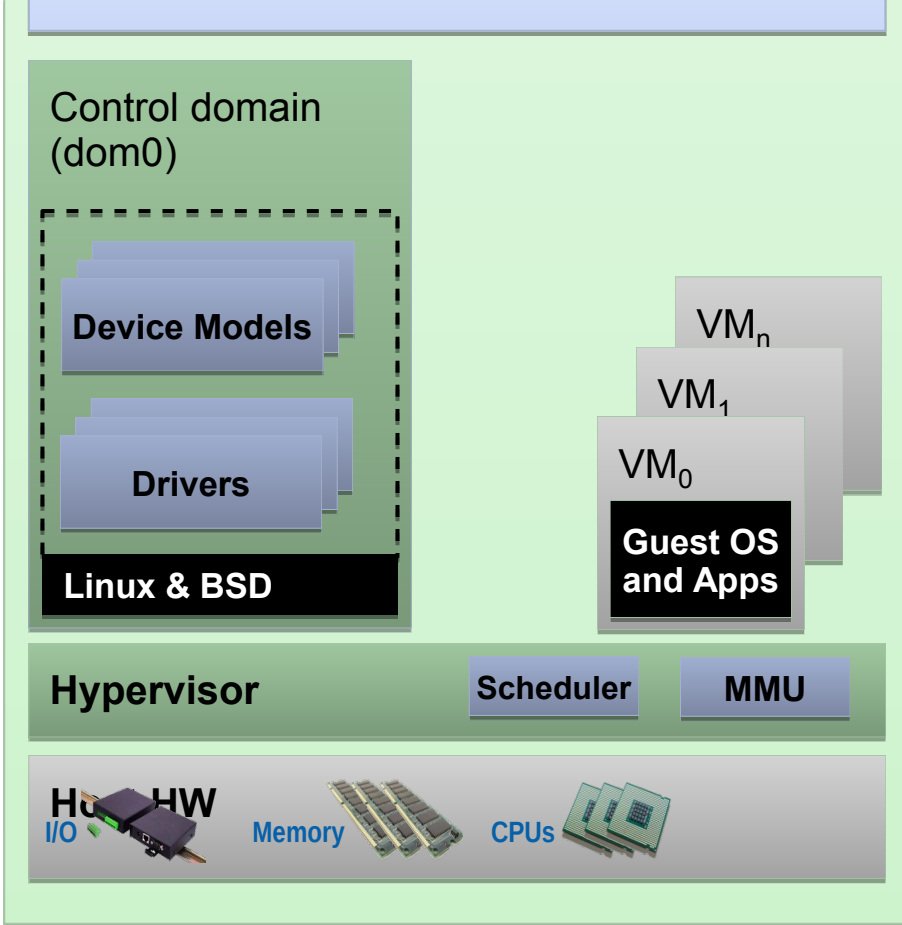


# Xen Project: Type 1 with a Twist

## Type 1: Bare metal Hypervisor



## Xen Project Architecture



# Some Bare-Metal Advantages

- What are the advantages of a Bare-Metal Hypervisor?
  - Density: It's thin
    - Excellent for supporting very small workloads
  - Scalability: It can support huge numbers of VMs
    - Terrific for highly dense workloads
  - Security: No host OS
    - It has no host OS layer to attack
  - Scheduling: Can use dedicated scheduler
    - Needed for specialized workload profiles where a host OS scheduler just won't do
  - Paravirtualization: Simplified interface
    - Easier to code to when no OS is present
- And now some of the innovations they enable...



# #1: Xen Automotive

- A subproject of the Xen Project
- Proposed by community member GlobalLogic
- Support for infotainment systems (for now...)
- Spares multiple discreet systems needing sourcing, installation, and testing
- ARM-based



# Automotive Challenges

- Soft-Real-time support
- Hard-Real-time support
- GPU virtualization
- Other co-processor (DSP, IPU, etc.)
- Certification
- Driver support for Android, e.g. Backend ION memory allocator and Linux User Space Device Drivers for Graphics, Sound, USB, Giros, GPS, etc.
- Driver support for operating systems such as QNX and other guest operating systems that are relevant for these use-cases



# A Focused Hypervisor

- Automotive requires extreme focus
- Simply repurposing a server-based hypervisor won't cut it
- A Bare-Metal hypervisor can add and modify pieces as needed
  - There is no legacy Host Operating System to be accommodated
  - Bare-Metal can do what the situation requires



# #2: Realtime Virtualization

- Support for Xen Automotive and beyond
- RT-Xen
- Streaming video, etc. cannot wait for next time slice
- Leverages a custom scheduler





# Custom Schedulers

- Type 2 (Hosted) Hypervisors use the scheduler of the host (e.g., Linux)
  - That scheduler is designed for the host operating system, not for special needs
- Type 1 (Bare Metal) Hypervisors use schedulers designed for the needs of the hypervisor itself
  - It is possible to change the scheduler to meet the needs of the hypervisor
  - That's the way to handle Realtime Scheduling



# A Scheduler for Every Need

- Current schedulers in Xen Project:
  - Credit
    - General Purpose
    - Default scheduler in 4.5
  - Credit2
    - Optimized for low latency & high VM density
    - Currently Experimental
    - Expected to become supported and default in future



# A Scheduler for Every Need

- Current schedulers in Xen Project (continued):
  - RTDS
    - Soft & Firm Realtime scheduler
    - Multicore
    - Currently Experimental
    - Embedded, Automotive, Graphics, Gaming in the Cloud
  - ARINC 653
    - Hard Realtime
    - Single Core
    - Currently Experimental
    - Avionics, Drones, Medical



# A Scheduler for Every Need

- Past schedulers in Xen Project:
  - Borrowed Virtual Time
  - Atropos
  - Round Robin
  - SEDF
- For more information:
  - [http://wiki.xenproject.org/wiki/Xen\\_Project\\_Schedulers](http://wiki.xenproject.org/wiki/Xen_Project_Schedulers)



# #3: ARM-based Hypervisor

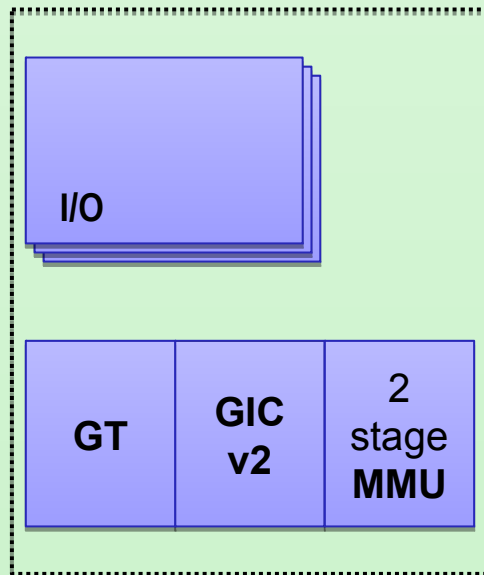
- ARM expanding from handhelds to servers
- Virtualization extensions added to ARM V7
- Architecture is hand-in-glove fit for Bare-Metal hypervisor
- No mode changes means greater speed and security



# Xen + ARM = a perfect Match

## ARM SOC

Device Tree describes ...



## ARM Architecture Features for Virtualization

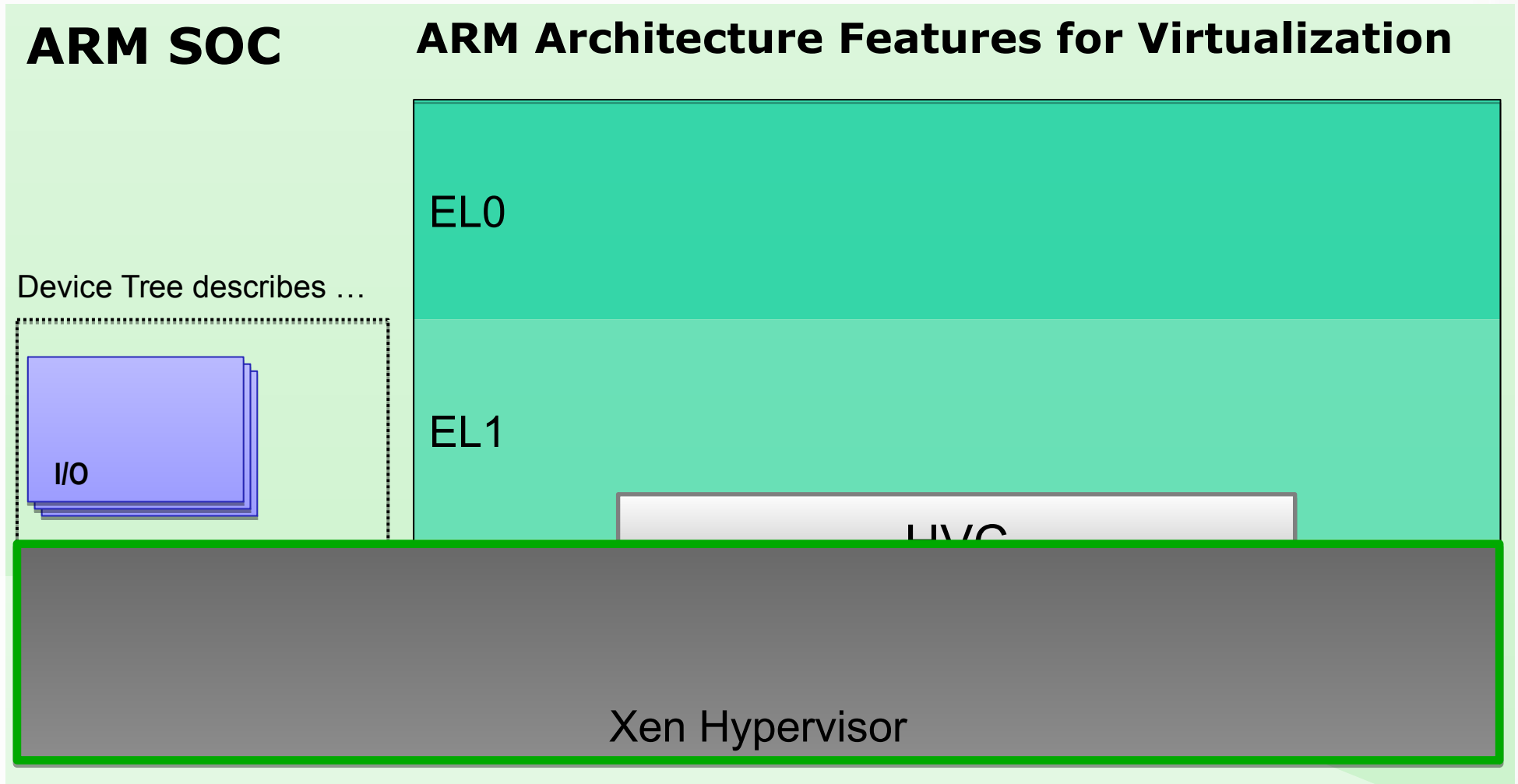
User mode : EL0

Kernel mode : EL1

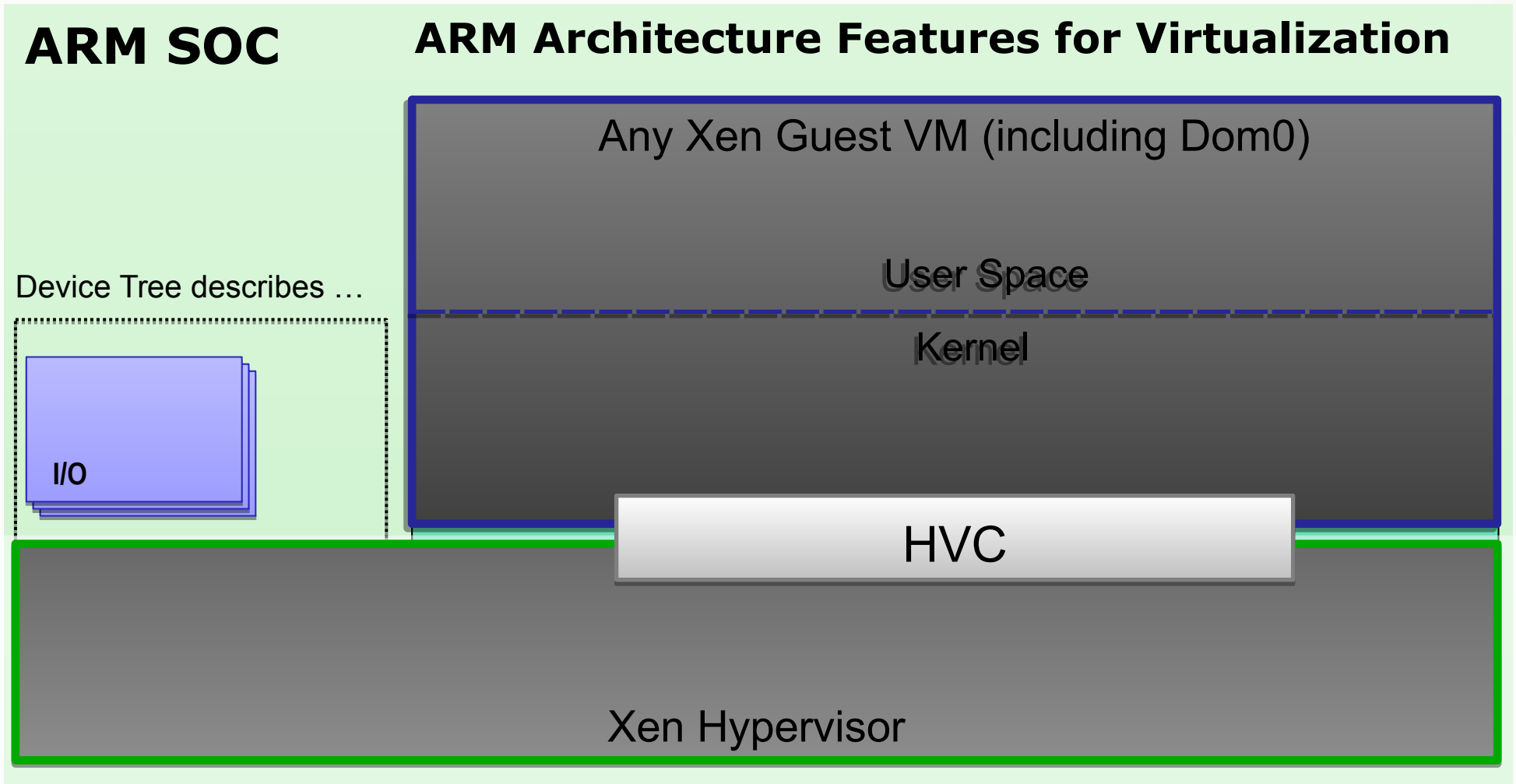
Hypercall Interface HVC

Hypervisor mode : EL2

# Xen + ARM = a perfect Match

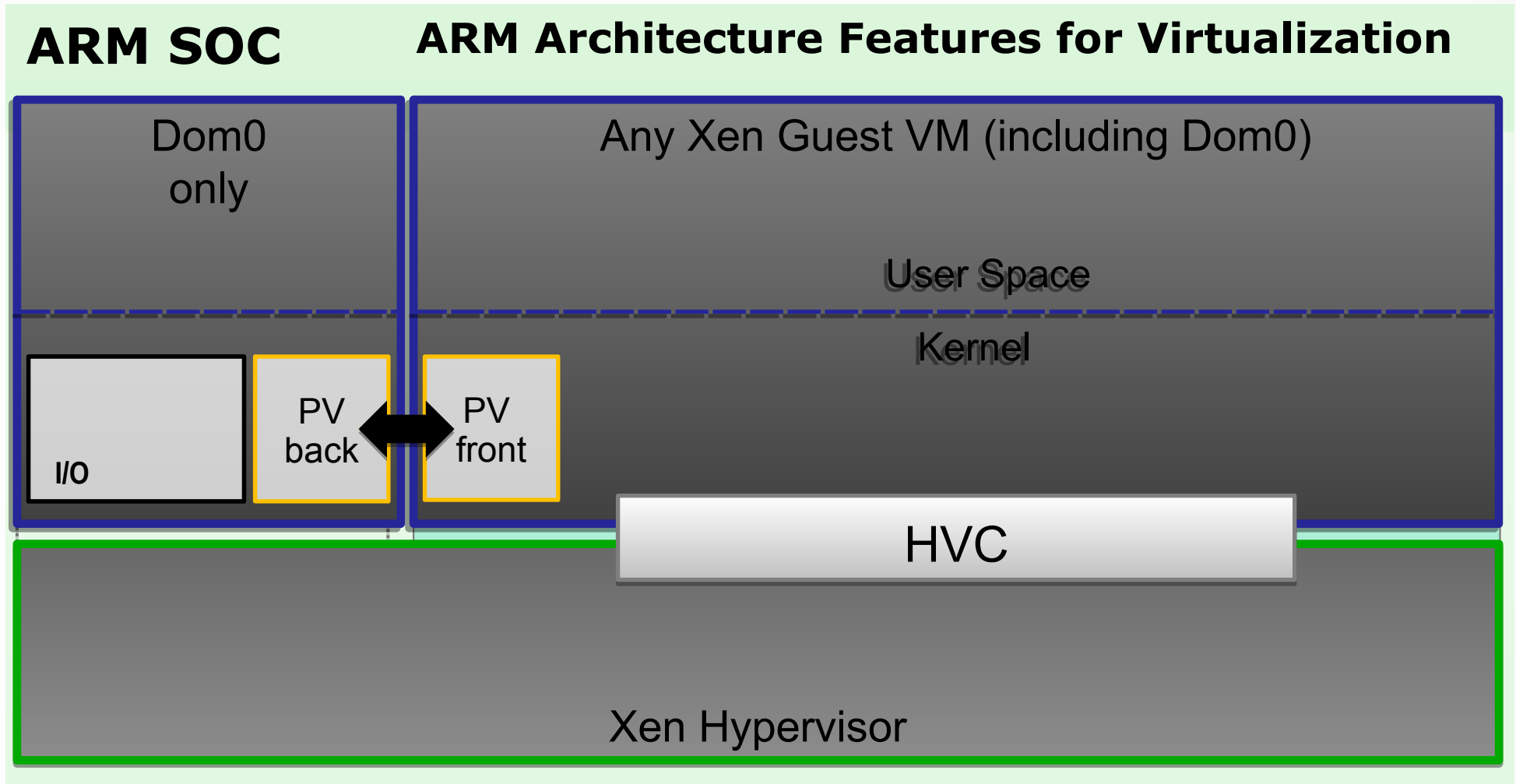


# Xen + ARM = a perfect Match





# Xen + ARM = a perfect Match



# Where Will an ARM Hypervisor Play?

- You name it...
  - Cell phones
    - Multiple personalities are possible
  - Embedded systems
    - Automotive is just the beginning
  - Internet of Things (IoT)
    - Lots of little things means lots of responses needed
  - Servers
    - Lower power footprint
    - Real green technology



# #4: The Unikernel

- Super-small VMs
- Quick booting
- Enhanced security
- Easy deployment
- Enables transient services
  - Services that appear when needed and disappear when done



# The Cloud We Know

- Field of innovation is in the orchestration
  - The Cloud Engine is paramount (OpenStack, CloudStack, etc.)
  - Workloads adapted to the cloud strongly resemble their non-cloud predecessors
    - Some basic adaptations to facilitate life in the cloud, but basically the same stuff that was used before the cloud
    - Applications with full stacks (operating system, utilities, languages, and apps) which could basically run on hardware, but are run on VMs instead.
    - VMs are beefy; large memory footprint, slow to start up
    - It all works, but its not overly efficient
    - 10s of VMs per physical host



# The Next Generation Cloud

- Turning the scrutiny to the workloads
  - Should be easier to deploy and manage
  - Smaller footprint, removing unnecessary duplication
  - Faster startup
  - Transient microservices
  - Higher levels of security
  - 1000s of VMs per host



# The New Stuff: Docker & Containers

- Makes deployment easier
- Smaller footprint by leveraging kernel of host
- Less memory needed to replicate shared kernel space
- Less disk needed to replicate shared executables
- Really fast startup times
- Higher number of VMs per host



# Docker Downsides

- Improvements, yes; but not without issues
  - Can't run any payload that can't use host kernel
  - Potential limits to scalability
    - Linux not really optimized for 1000s of processes
  - Security
    - Security is a HUGE issue in clouds
    - Still working on real security; someday...
    - At LinuxCon North America 2014, Docker CEO doesn't even identify security as one of the top priorities
    - Google & others run Docker in VMs when they need security



# The Unikernel: A Real Cloud Concept

- Very small
- Very efficient
- Very quick to boot
- And very, VERY secure!
- It's a Green (energy) technology which saves you green (cash); extremely important to foster adoption
- Many unikernels already exist, including Mini-OS and Mirage OS, a Xen Project Incubator Project





# What is a Unikernel? From Mirage OS

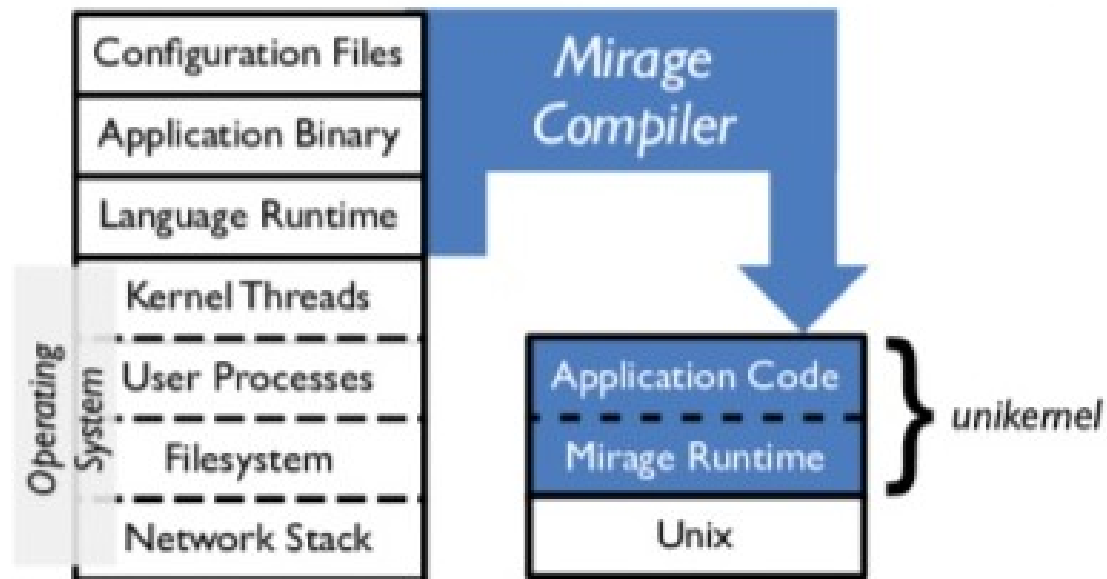


## THE UNIKERNEL APPROACH

***Unikernels** are specialised virtual machine images compiled from the modular stack of application code, system libraries and configuration*

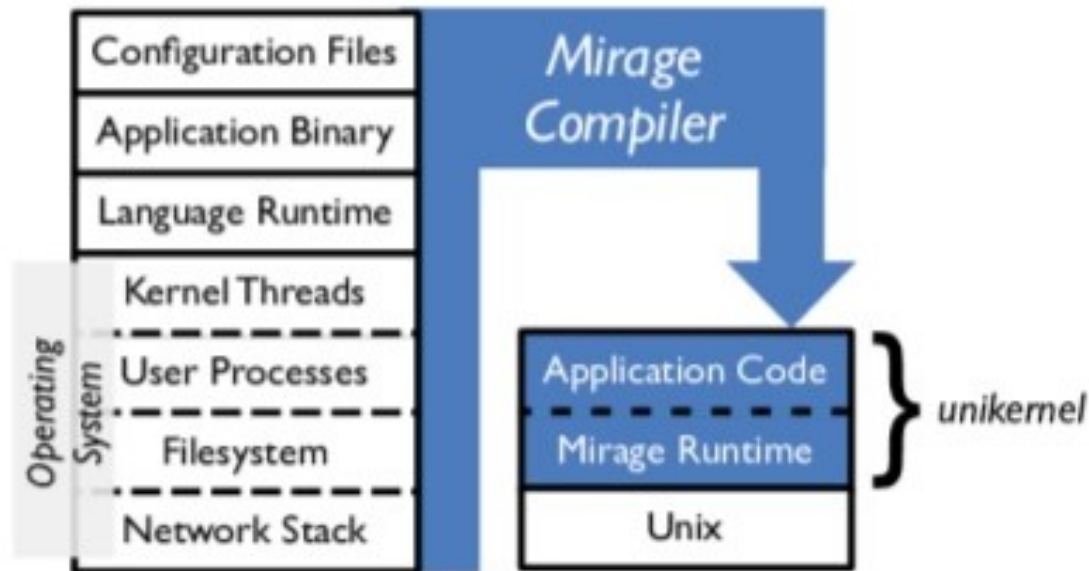
# Unikernel Approach: Mirage OS

Swap system libraries to target different platforms:  
**develop application logic using native Unix.**



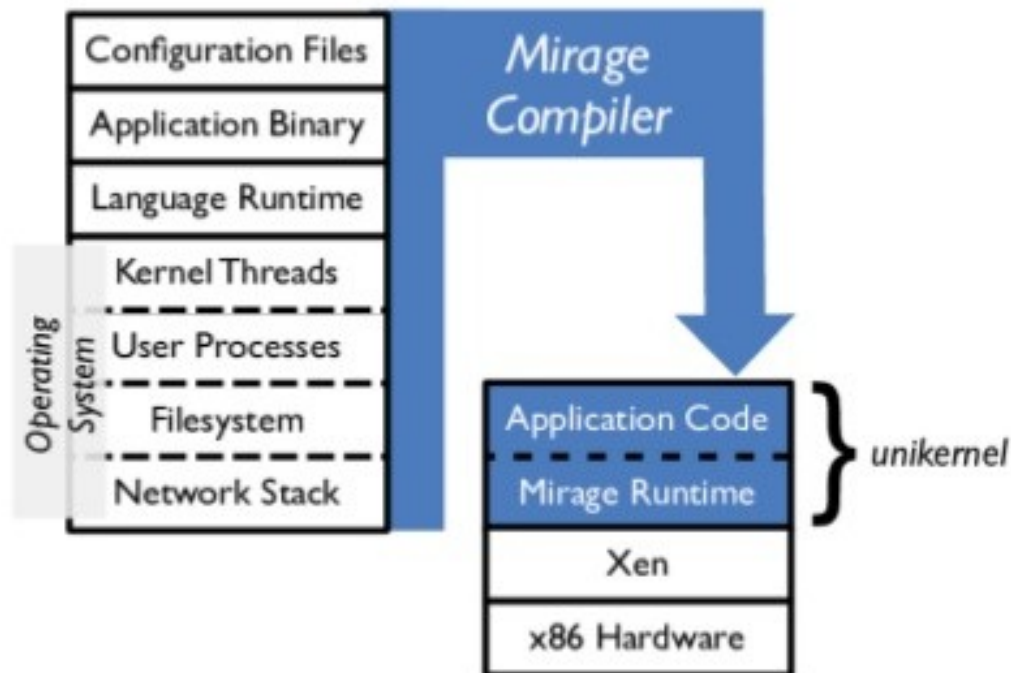
# Unikernel Approach: Mirage OS

Swap system libraries to target different platforms:  
**test unikernel using Mirage system libraries.**



# Unikernel Approach: Mirage OS

Swap system libraries to target different platforms:  
**deploy by specialising unikernel to Xen.**



# Unikernel Concepts

- Use just enough to do the job
  - No need for multiple users; one VM per user
  - No need for a general purpose operating system
  - No need for utilities
  - No need for a full set of operating system functions
- Lean and mean
  - Minimal waste
  - Tiny size



# Unikernel Concepts

- Similar to an embedded application development environment
  - Limited debugging available for deployed production system
  - Instead, system failures are reproduced and analyzed on a full operating system stack and then encapsulated into a new image to deploy
  - Tradeoff is required for ultralight images



# What Do the Results Look Like?

- Mirage OS examples:
  - DNS Server: 449 KB
  - Web Server: 674 KB
  - OpenFlow Learning Switch: 393 KB
- LING metrics:
  - Boot time to shell in under 100ms
  - Erlangonxen.org memory usage: 8.7 MB
- ClickOS:
  - Network devices processing >5 million pkt/sec
  - 6 MB memory with 30 ms boot time



# What About Security?

- Type-Safe Solution Stack
  - Can be certified
  - Certification is crucial for certain highly critical tasks, like airplane fly-by-wire control systems
- Image footprints are unique to the image
  - Intruders cannot rely on always finding certain libraries
  - No utilities to exploit, no shell to manipulate





# What's Out There Right Now?

- Mirage OS, from the Xen Project Incubator
- HaLVM, from Galois
- LING, from Erlang-on-Xen
- ClickOS, from NEC Europe Labs
- OSv, from Cloudeus Systems
- Rumprun, from the Rump Kernel Project
- And that's just the beginning...



# How Does Xen Project Enable Unikernels?

- No Host OS means it's lean and mean
  - A tiny VM can sit on a thin hypervisor layer on the hardware
  - Attack surface is small
  - Scale out support
    - Can currently support about 600 concurrent VMs per host without losing performance
    - Current target: 2000-3000 concurrent VMs per host
  - Enhanced scheduler (Credit2)
  - ARM as an option



# Innovation: Is This All?

- By no means!
- The list of other subprojects & capabilities continues to grow:
  - Virtualized GPUs
  - Enhanced NUMA
  - COLO: Coarse-grained lockstepping of VMs
  - Native VMware VMDK support
  - And so on...
- <http://xenproject.org/users/innovations.html>



# In Review...

- Some advantages of a Bare-Metal Hypervisor
  - Density: It's thin
    - Excellent for supporting very small workloads
  - Scalability: It can support huge numbers of VMs
    - Terrific for highly dense workloads
  - Security: No host OS
    - It has no host OS layer to attack
  - Scheduling: Can use dedicated scheduler
    - Needed for specialized workload profiles where a host OS scheduler just won't do
  - Paravirtualization: Simplified interface
    - Easier to code to when no OS is present



# The Xen Project Difference

- Tomorrow's workloads are not yesterday's workloads
  - If your hypervisor is just focused on yesterday's payloads, it is suffering from planned obsolescence
  - Select a hypervisor which is innovating – and Open Source
- Xen Project is busy enabling the next generation in virtualization



# Questions?

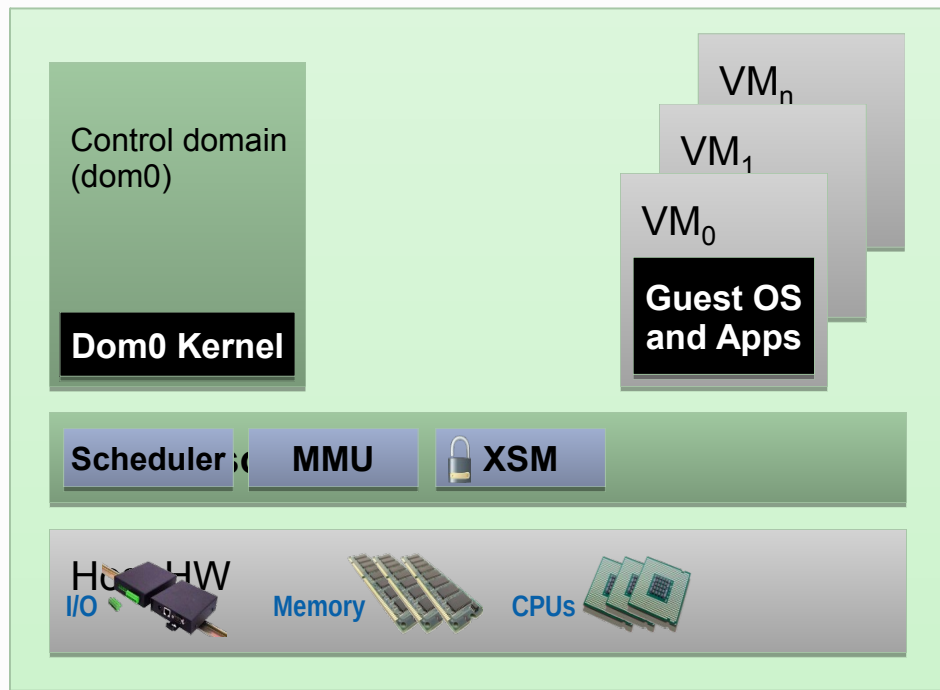
[Russell.Pavlicek@XenProject.org](mailto:Russell.Pavlicek@XenProject.org)

Twitter: @RCPavlicek

This presentation will be available in the Presentations Section of  
[XenProject.org](http://XenProject.org)



# Basic Xen Project Concepts



■ Trusted Computing Base

## Console

Interface to the outside world

## Control Domain aka Dom0

- Dom0 kernel with drivers
- Xen Management Toolstack

## Guest Domains

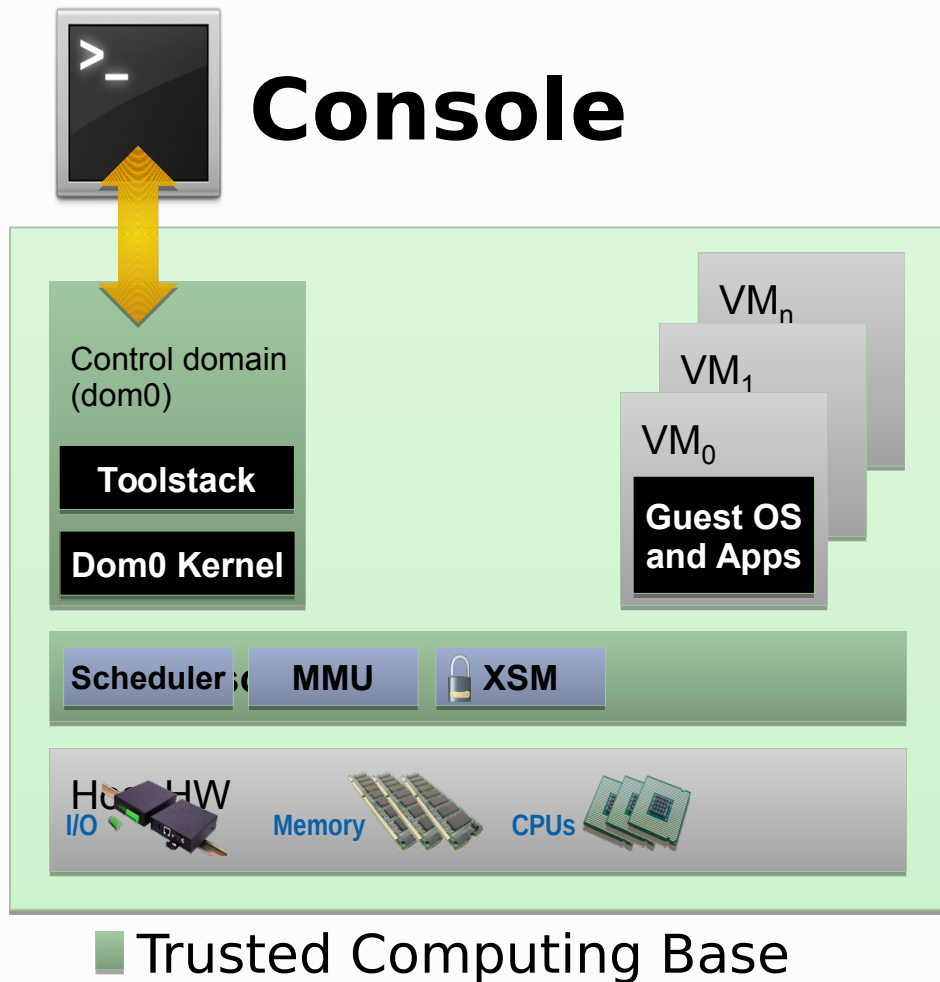
- Your apps

## Driver/Stub/Service Domain(s)

A "driver, device model or control service in a box"  
De-privileged and isolated  
Lifetime: start, stop, kill



# Basic Xen Project Concepts: Toolstack+



## Console

- Interface to the outside world

## Control Domain aka Dom0

- Dom0 kernel with drivers
- Xen Management Toolstack

## Guest Domains

- Your apps

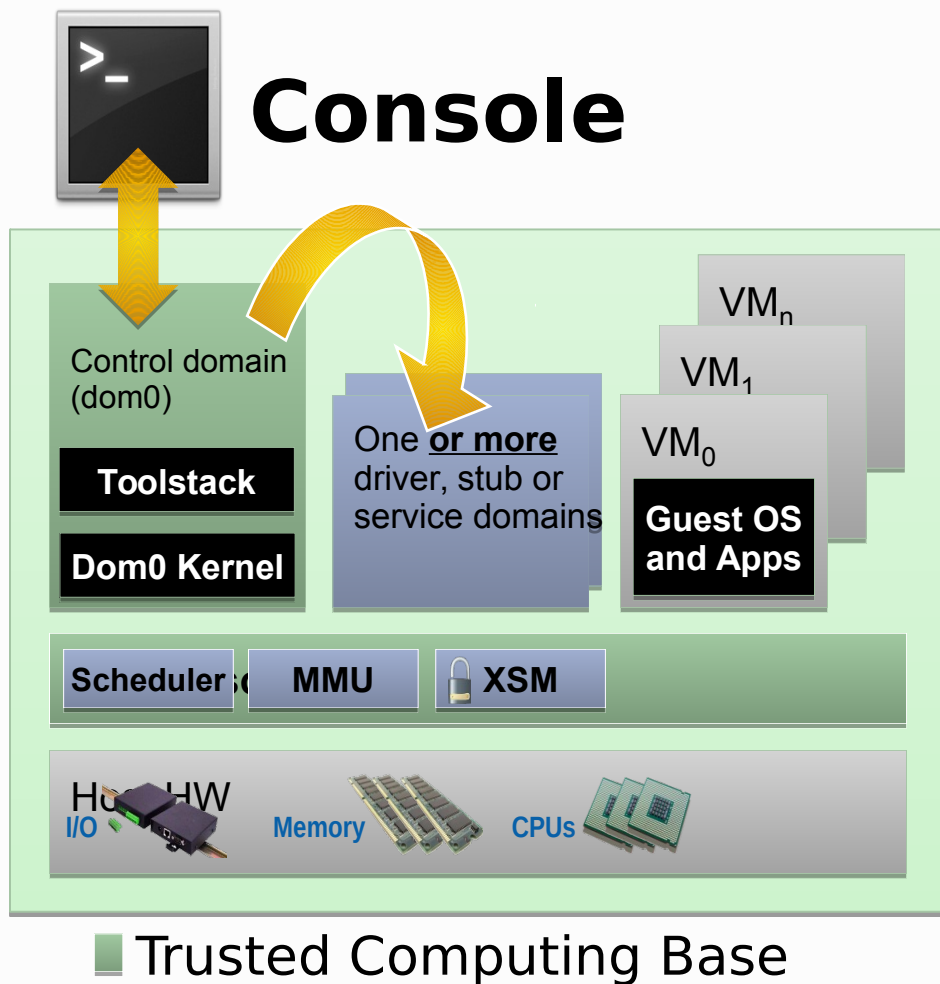
## Driver/Stub/Service Domain(s)

A "driver, device model or control service in a box"  
De-privileged and isolated  
Lifetime: start, stop, kill





# Basic Xen Project Concepts: Disaggregation



## Console

- Interface to the outside world

## Control Domain aka Dom0

- Dom0 kernel with drivers
- Xen Management Toolstack

## Guest Domains

- Your apps

## Driver/Stub/Service Domain(s)

- A “driver, device model or control service in a box”
- De-privileged and isolated
- Lifetime: start, stop, kill

