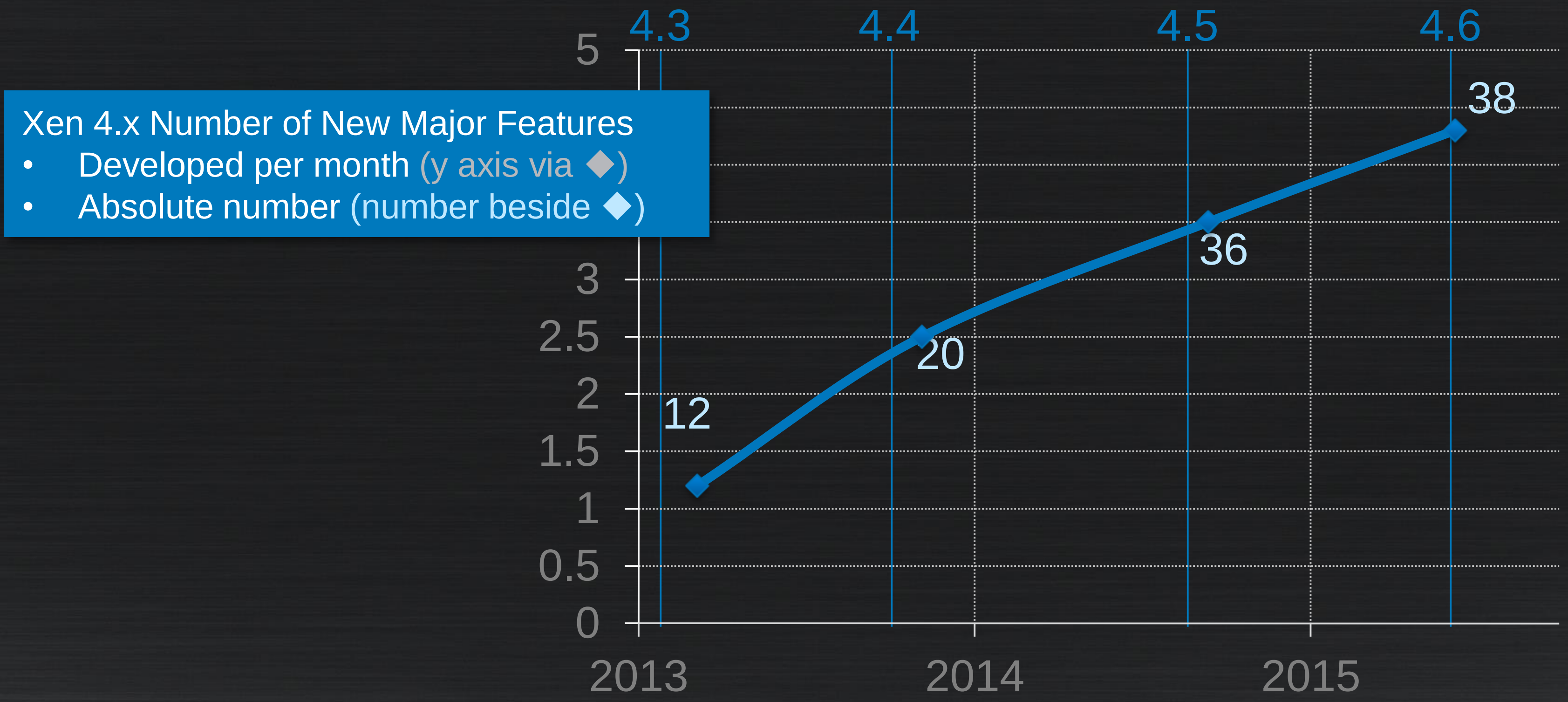
Scale 14x: The Latest From the Xen Project

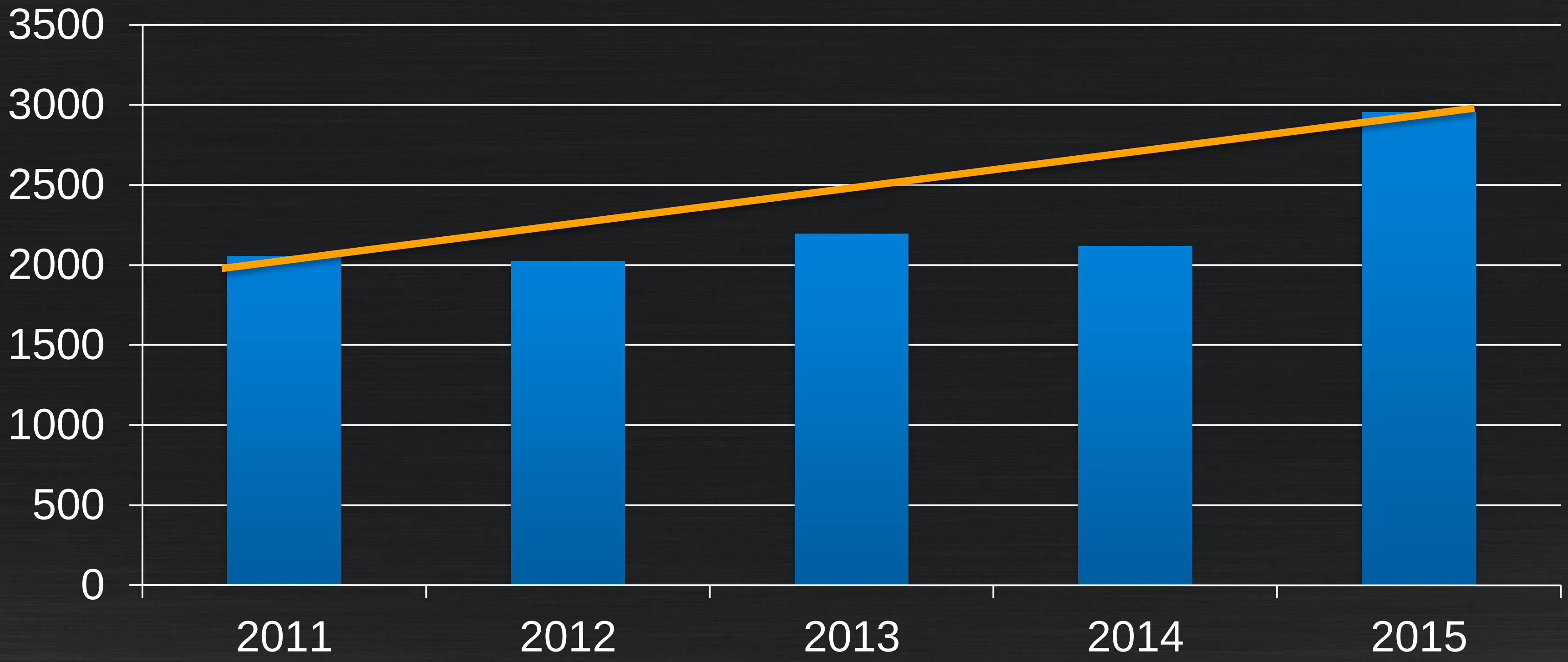
Lars Kurth
Cat Herder, Xen Project
Chairman, Xen Project Advisory Board

IRC lars_kurth

Xen 4.x Hypervisor Release History



Hypervisor Git Commits



Rate of Innovation is Accelerating

While quality and security requirements are increasing simultaneously

A type-1 Hypervisor

with a twist


Console
VM$_0$ (or Dom0)
Toolstack
Dom0 Kernel
Drivers
back
VM$_1$
Applications
Guest OS
PV front
VM$_2$
Applications
Guest OS
VM$_n$
Applications
Guest OS
Config
Scheduler
MMU
Timers
Interrupts
I/O
Memory
CPUs
HW

Architectural Advantages

Density: It's thin
Excellent for supporting many very small workloads (e.g. unikernels)

Scalability: It can support huge numbers of VMs
Terrific for highly dense workloads (e.g. unikernels, disaggregation, …)

Security: Host OS isolated within a VM
This makes it harder to attack the Host OS

Scheduling: Can use dedicated scheduler
Enables specialized workload profiles (mix and match schedulers on one host)

Paravirtualization: Simplified interface
Easy to implement a unikernel base
Enables fast boot times necessary for unikernels

The Interface Between Xen and Unikernels

Virtualisation Modes
The Future of Virtualization Modes
Implications for Unikernel bases

Evolution of Virtualization Modes (x86)

Shortcut	Xen	Mode	With
HVM / Fully Virtualized		HVM	
HVM + PV drivers	3.0	HVM	PV Drivers
PVHVM	4.0	HVM	PVHVM Drivers
PVH	4.4/4.5	PV	pvh=1
PV		PV	

Unikernel Bases:

Primarily depend on PV
E.g. rumprun and Mini-OS

Will work on Xen based clouds and hosting services

Poor Performance
Scope for Improvement
Optimal Performance

PV = Paravirtualized
VS = Software Virtualized
VH = Hardware Virtualized

Shortcut	Mode	With	Disk and Network	Interrupts & Timers	Emulated Motherboard, Legacy Boot	Privileged Instructions, Page Tables
HVM / Fully Virtualized	HVM		VS	VS	VS	VH
HVM + PV drivers	HVM	PV Drivers	PV	VS	VS	VH
PVHVM	HVM	PVHVM Drivers	PV	PV	VS	VH
PVH	PV	pvh=1	PV	PV	PV	VH
PV	PV		PV	PV	PV	P

Making PVH better

The **motivation** behind PVH

- HVM (like) Dom0: performance & Dom0 modification
- PVH as fast or faster than HVM
- PVH runs a PV guest within a HVM container (essentially a mix of PV & HVM)

BUT: PVH inherits all the **PV limitations**, e.g.

- Paging restrictions, lack of access to emulated devices (if needed), …
- Concept designed prior to "additional quality and security requirements"

Solution: HVMlite to eventually replace PVH

- A lot simpler to implement: less code to maintain and thus to keep secure
- Behaves exactly like PVH (but internal implementation different)
- HVM without QEMU

Shortcut	Mode	With	Disk and Netw	Interrupts & Tim	Emulated Motherb Legacy Bo	Privileged Instructions, Page Tabl
HVM / Fully Virtualized	HVM				S	VH
HVM + PV drivers	HVM	PV Dr			S	VH
PVHVM	HVM	PVHVM D rs	PV	PV	VS	VH
~~PVH~~ / HVMlite	HVM	pvh=1 / …	PV	PV	PV	VH
PV	PV		PV	PV	PV	P

Currently:
builder = "hvm"
device_model_version="none"

There will need to be some sort of migration strategy from pvh=1

Status

Xen 4.7 (June 2016)

- HVMlite DomU support in xen.git
- Agree on config file changes and naming (PVH or HVMlite)

Ongoing

- HVMlite Dom0 prototype for FreeBSD
- No Linux implementation yet
- Some clean-up required
- Interfaces not yet declared stable (but almost there)
- Benchmarks already very impressive

Call to Action

PVH / HVMlite not currently used as unikernel base

- Unikernel developers make sure the architecture works for you (before APIs are declared stable)
- MiniOS / Rumprun not yet ported to HVMlite (some different approaches needed compared to pure PV)

Opportunity: Avoid Duplication

- There was a bit of duplication of unikernel bases in the early days of unikernel development (MiniOS clones)

Performance

Focus on significant performance and scalability improvements since Xen 4.5+

Performance Improvements

Examples

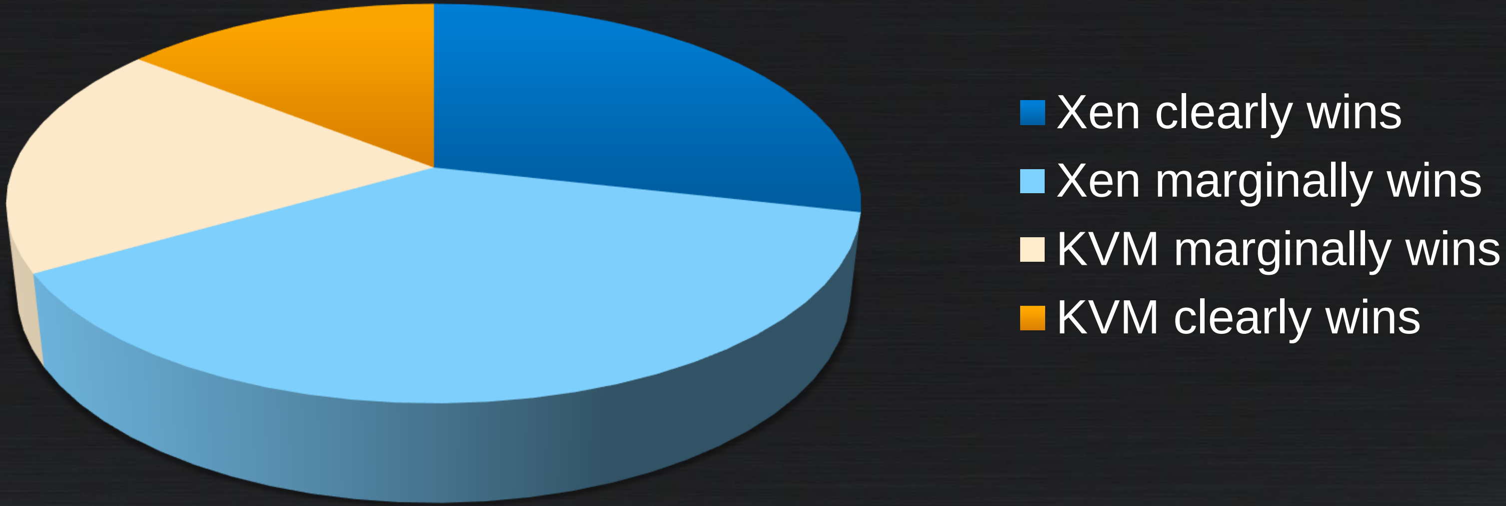
HPET: Better and faster resolution values
Parallel memory scrubbing on boot (large machines)
Lower interrupt latency for PCI passthrough (machines > 2 sockets)
Soft affinity for non-NUMA machines
Multiple IO-REQ services for guests
(remove bottlenecks for HVM guests by allowing multiple QEMU back-ends)
SandyBridge: VT-d posted interrupts for HVM
(I/O intensive workloads)

Grant table scalability by using finer grained locks
Ticket locks for improved fairness and scalability
…

Benchmarks: Xen & KVM

(Kernel 4.2 with Xen 4.5 & QEMU 2.3)

Xen 4.6 should be even better



Source: http://www.phoronix.com/scan.php?page=article&item=ubuntu-1510-virt

Schedulers

Overview
Possibilities

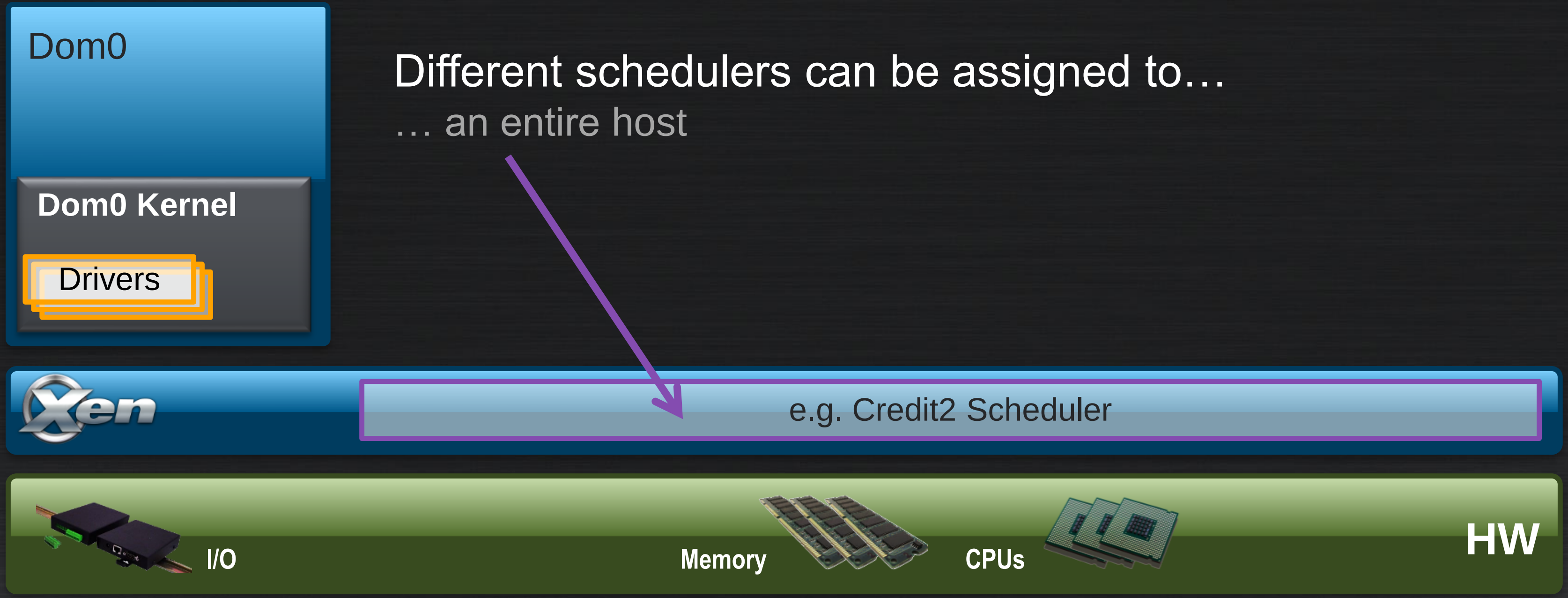
Resources:
Docs: bit.do/xen-schedulers

Overview: Xen Project Schedulers

The Xen Project Hypervisor supports **several different** schedulers with different properties.

Different schedulers can be assigned to…

… an entire host



Overview: Xen Project Schedulers

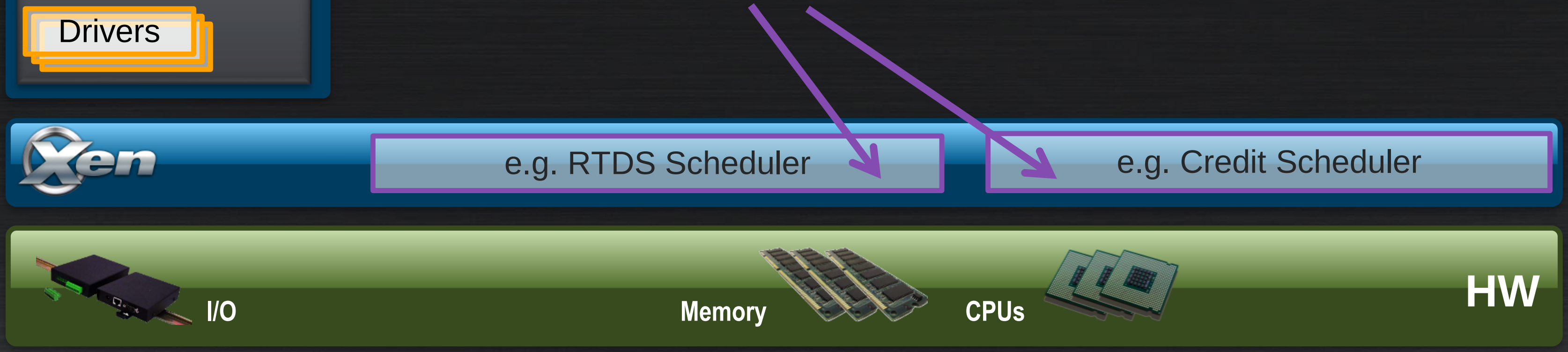
The Xen Project Hypervisor supports **several different** schedulers with different properties.

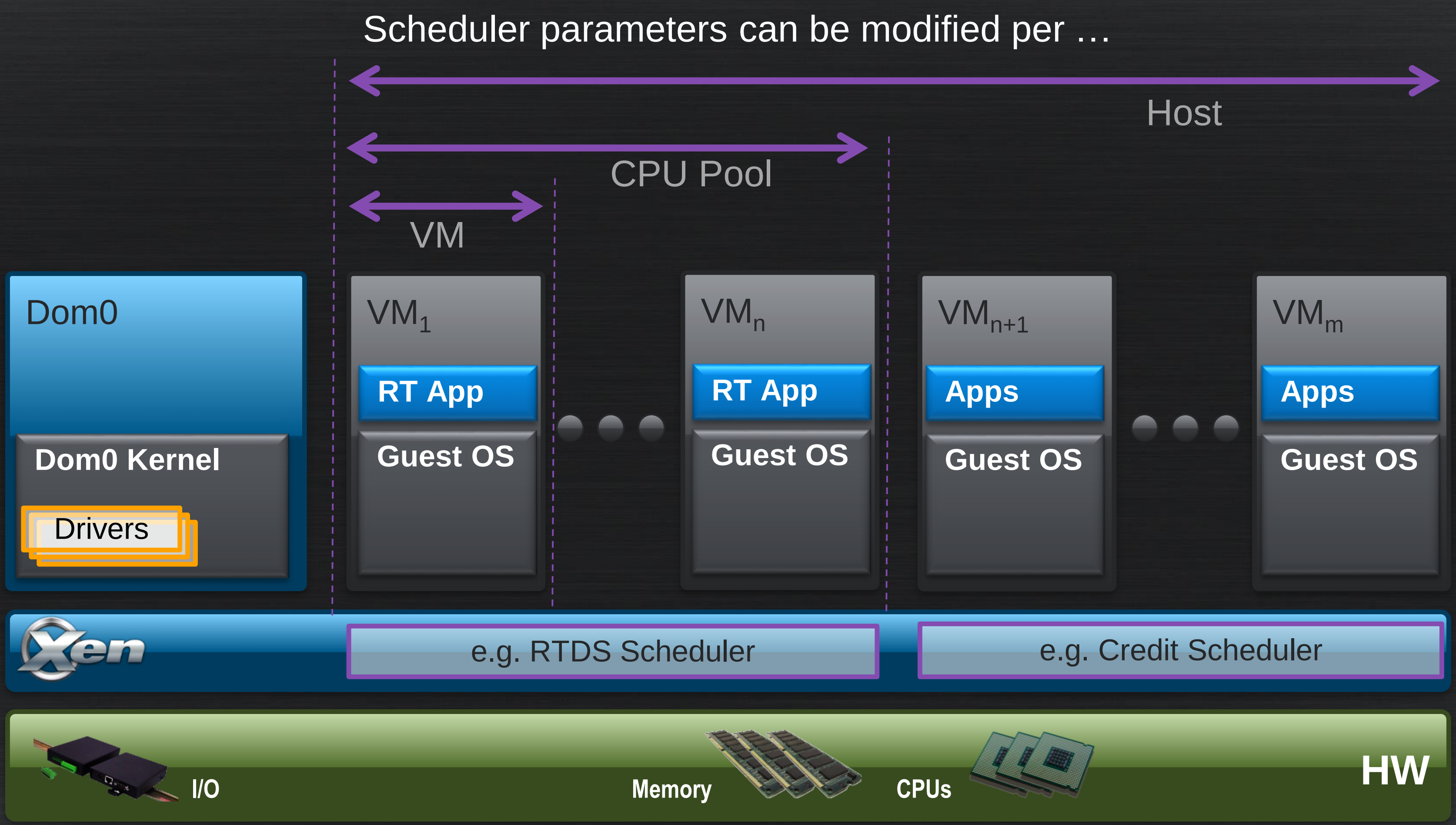
Different schedulers can be assigned to…

… an entire host

… a pool of physical CPU's (=CPU Pool) on a host

(VMs need to be assigned to a pool or pinned to a CPU)



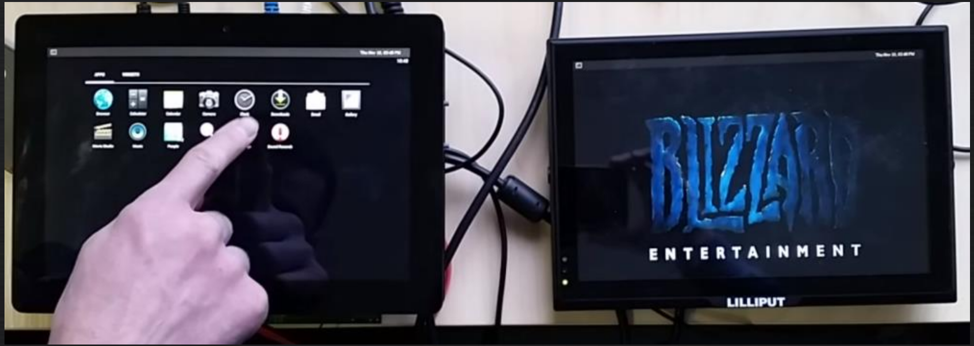

Scheduler parameters can be modified per …
Host
CPU Pool
VM
Dom0
Dom0 Kernel
Drivers
VM1
RT App
Guest OS
VMn
RT App
Guest OS
VMn+1
Apps
Guest OS
VMm
Apps
Guest OS
Xen
e.g. RTDS Scheduler
e.g. Credit Scheduler
I/O
Memory
CPUs
HW

Schedulers Overview

Scheduler	Use-cases	Xen 4.6	Plans for 4.7+
Credit	General Purpose	Supported **Default**	Supported **Default**
Credit 2	General Purpose Optimized for lower latency, higher VM density	Experimental	Supported
RTDS	Soft & Firm Real-time **Multicore** Embedded, Automotive, Graphics & Gaming in the Cloud, Low Latency Workloads	Hardening Optimizations Better XL support Experimental	Adaptive granularity Supported
ARINC 653	Hard Real-time **Single core** Avionics, Drones, Medical	Supported Compile time	No change

Legend:
likely in 4.7
possible in 4.7

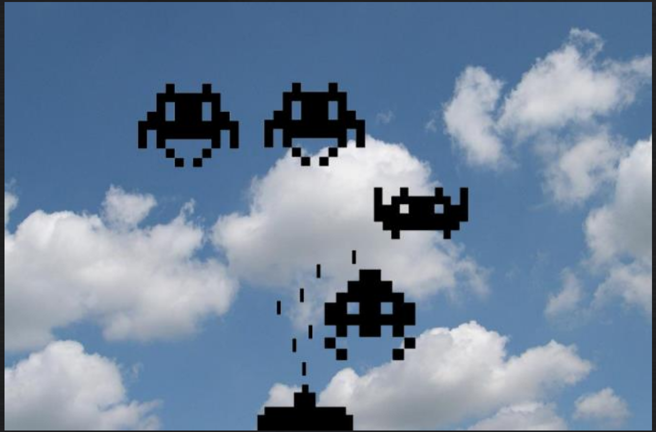
RTDS Scheduler: Use-cases



Embedded & automotive

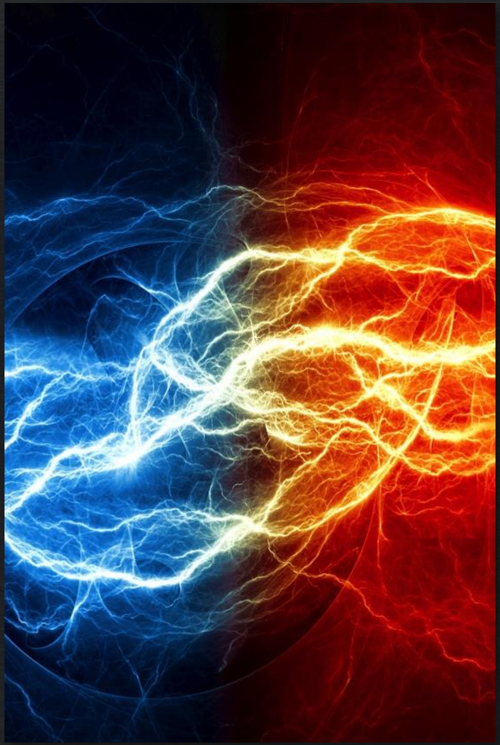
Latency sensitive workloads

Guaranteed QoS



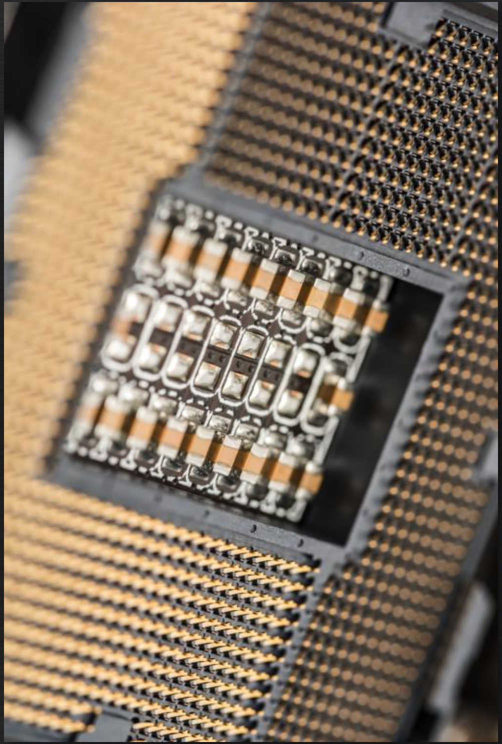
Cloud based gaming, video, TV delivery, …

Guaranteed QoS
(Price ➔ SLAs ➔ QoS)



Other major and on-going Innovations in Xen

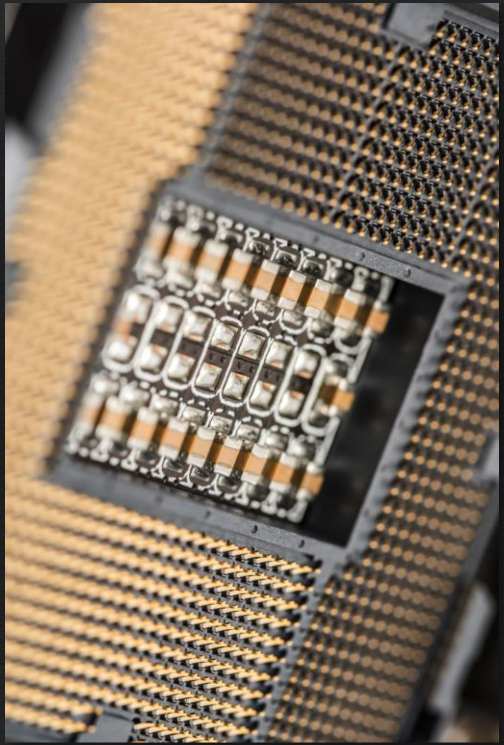
Hardware Support
Graphics
Security: VMI
Security: QEMU and Isolation
Security: xSplice
Security: Configurability
…



X86 Hardware Support

Intel Platform QoS technologies (CMT, CAT, MBM, …)
Virtual Performance Monitoring Unit
vTPM v2.0

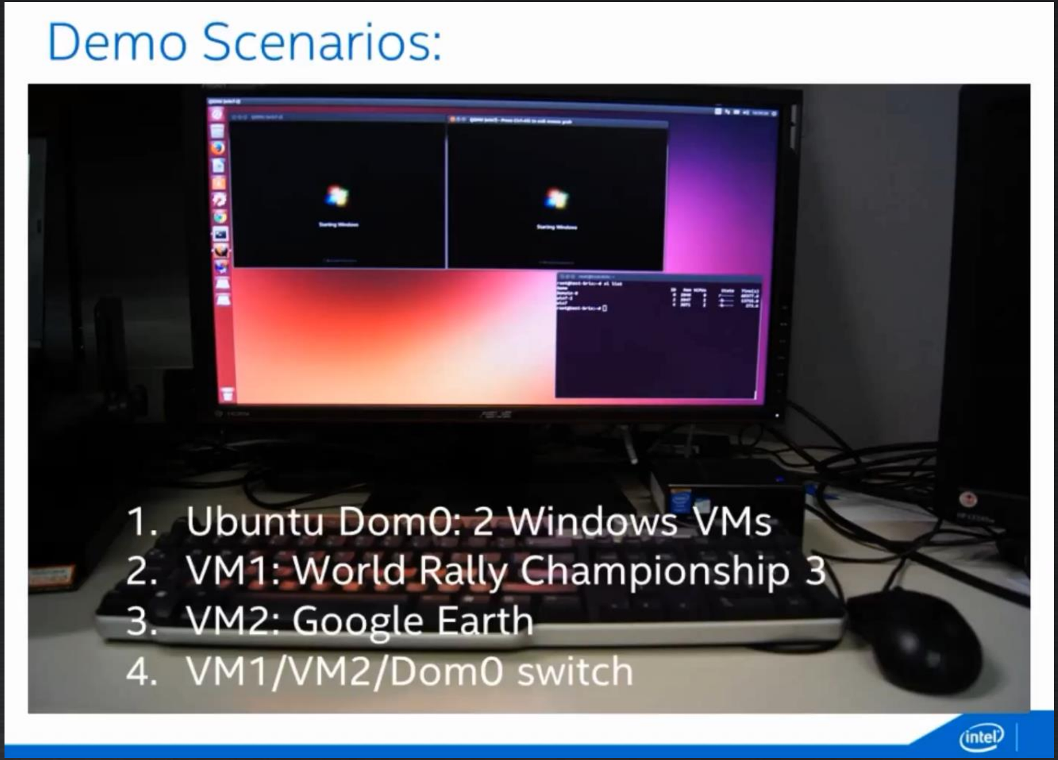
Code/Data Prioritization
Memory Protection Keys
VMX TSC scaling
Intel PState Drivers
Posted Interrupts
…



ARM Hardware Support

Tracking ARM/ARM-partner server roadmap
Hardening (more 64 bit servers in Test Lab)
Live Migration

What is Intel GVT-g (XenGT)?



Watch the demo at

https://www.youtube.com/watch?v=V2i8HCcAnY8

Virtual GPU per VM

Performance critical resources directly assigned to VM

Intel GVT-g (XenGT) – What's next?

GVT-g support is partly out-of-tree

In use by XenClient 5.5 and XenServer Dundee

Most Xen patches are part of xen.git

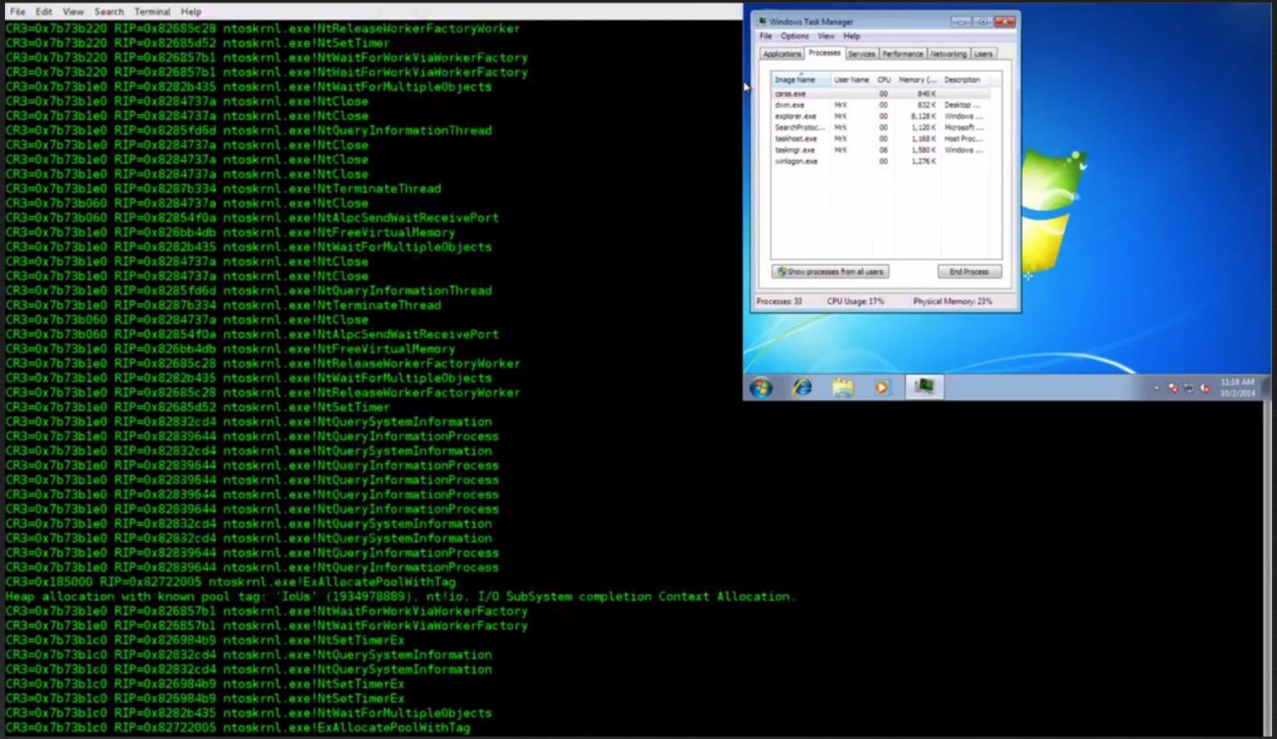
BUT: some Linux and QEMU patches that are still in progress

Motivation: create a common code base for Xen & KVM

Similar approach for embedded developed by GlobalLogic

(for ARM based architectures)

Demo: Virtual Machine Introspection



Watch the demo at

https://www.youtube.com/watch?v=ZJPHfpDiN4o

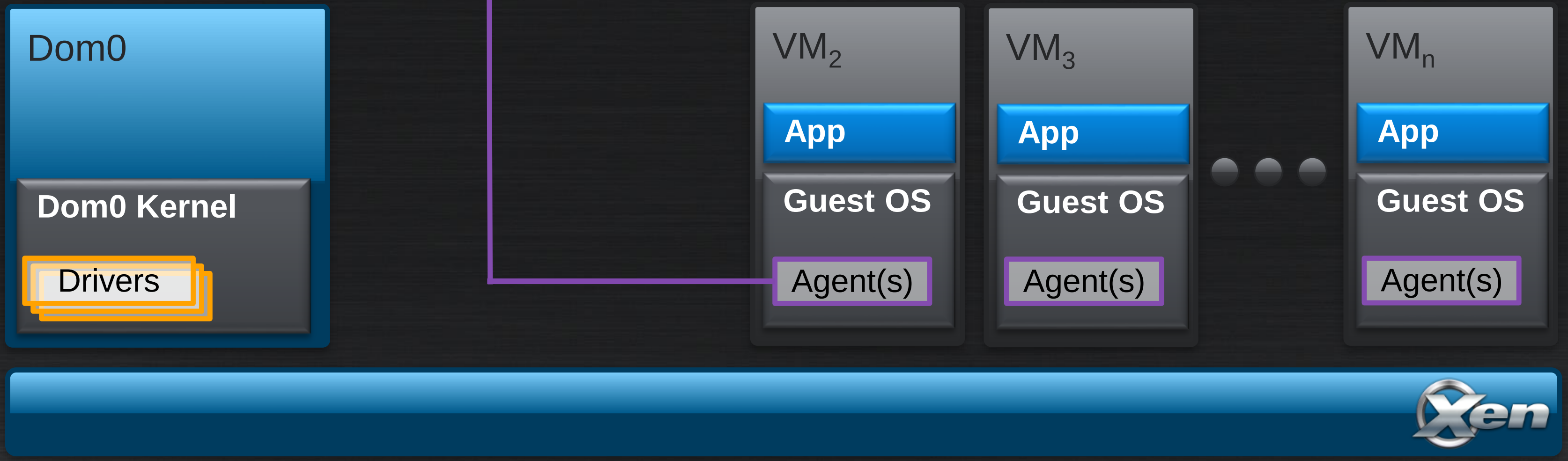
Credit: Tamas K Lengyel

Cloud Security Today

Today

Installed in-guest agents, e.g. anti-virus software, VM disk & memory scanner, network monitor, etc.

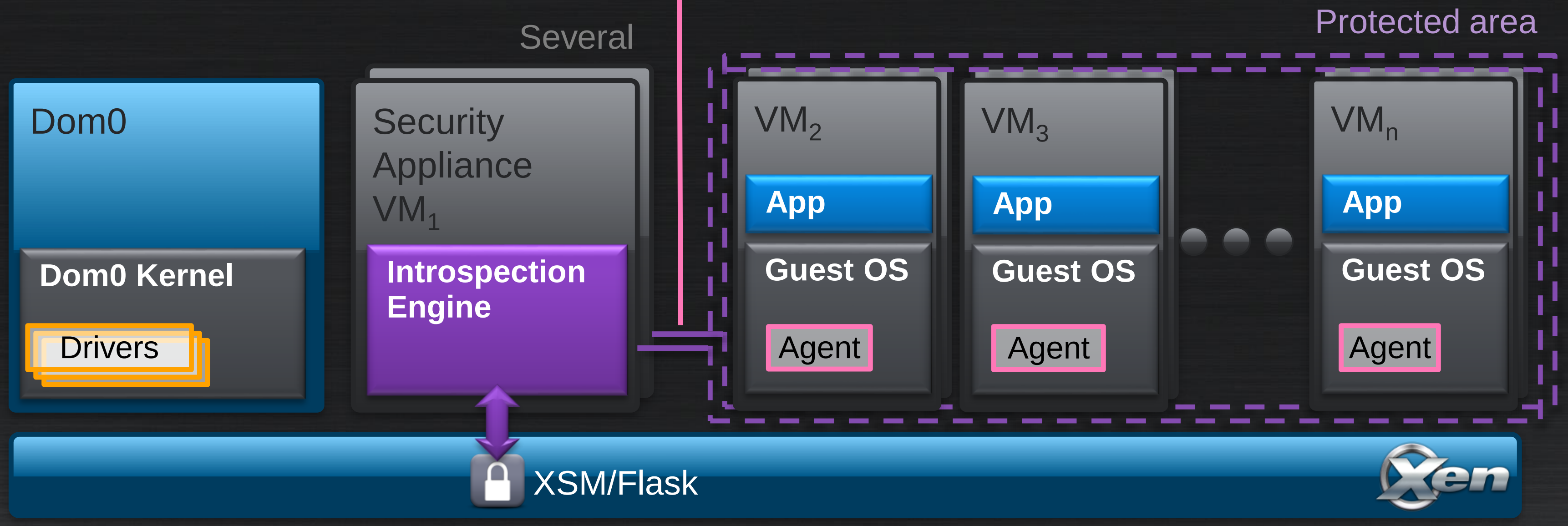
Anti virus storm, deployment/maintenance, …



A new model for Cloud Security?

VMI Approach

Hybrid approach: no need to move everything outside (chose best trade-off)



Other Security Features being developed

QEMU and Emulation for Xen secure by default (4.7)

- Response to Venom and other QEMU bugs as an alternative to Stub Domains
- Defense in depth mechanisms to secure the execution of QEMU + Inbuilt Emulation

Hot Patching or x-splice (4.7+)

- Response to “Cloud Reboots” of 2014 and 2015
- Hypervisor and Workload generation Tooling
- Start with some use-cases and successively add less common ones

Better Configurability (4.7+)

- Response to criticism from Invisible Things Labs
- Use KCONFIG to disable Hypervisor functionality
- A more wholesome approach to disable and remove undesired functionality

Conclusion

The project has a history of proactively innovating

The rate of innovation is increasing
(e.g. more features, more quickly)

The demands on the project are shifting
(e.g. quality and security, conflicting requirements)

The project has a track record of adapting
(e.g. to criticism, challenges, …)

Best Platform for Unikernels in the Cloud
(e.g. reach, innovation, unique features)