Unikernelized Preempt-RT Linux & IoT
-- UniLinux for IoT

Tiejun Chen <tiejunc@vmware.com>
VMware China R&D Advanced Technology Center

vmware

Warning ☺

This is our own exploration of unikernels.

This is not a roadmap or commitment from VMware.

vmware

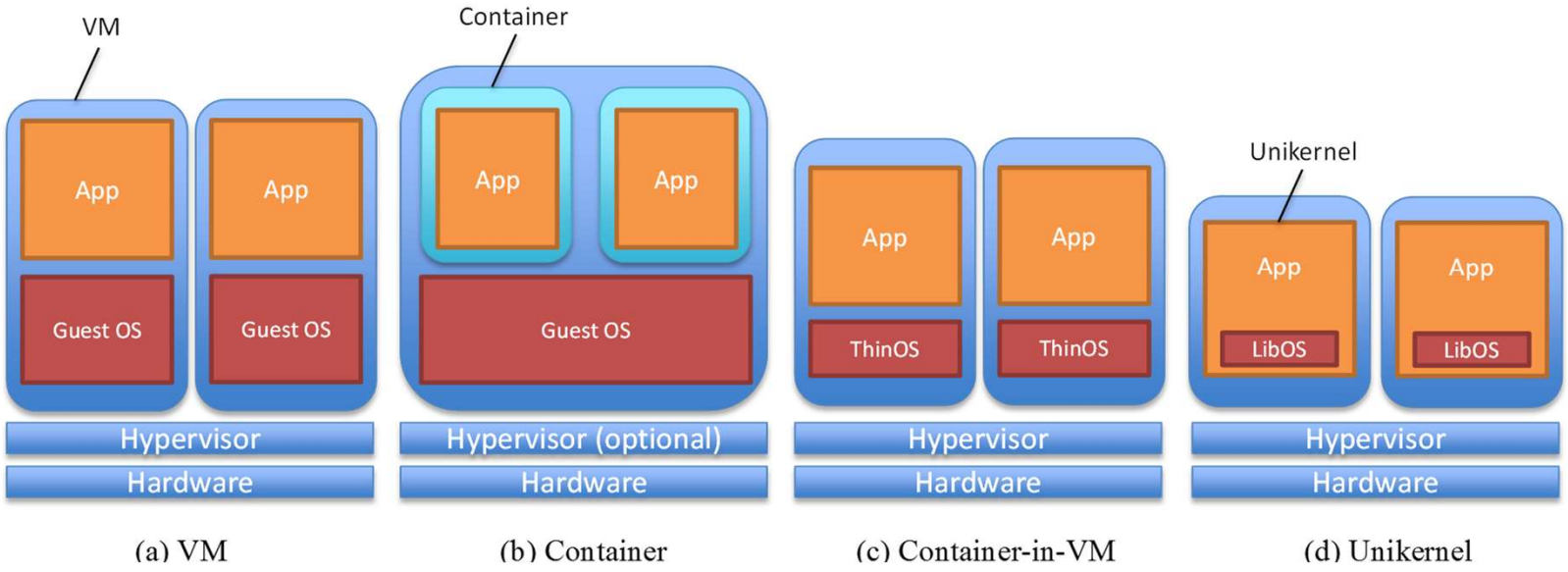
Agenda

- Unikernels Background
- IoT Background
- Unikernel exploration -- UniLinux
- Summary

vmware

Architectural Evolution Background

- VM with traditional OS
- Linux container technologies like Docker
- Container as a VM
- But now unikernels is really beginning to attract our attention !



Unikernels

- **Definition**
  - Unikernels are specialised, single address space machine images constructed by using library operating systems. *--Wiki*
- **Types**
  - General purpose unikernels
  - Language specific unikernels
- **Unikernels approaches**
  - OSv, IncludeOS, MirageOS, ClickOS, Clive, HaLVM, LING, Rump Kernels, Solo5 Unikernel and Drawbridge
  - Unik
- **Unikernels solutions**
  - Docker/Mikelangelo/NFV

vmware

Unikernel Essentials

- **The biggest characteristics**
  - Single address space: Zero-copy and huge page
  - Single running mode: Perform the efficient function call
  - One process with multiple threads: No heavy context switch and TLB flush
- **Unikernels still provide many comparable benefits**
  - Improved security
  - Small footprints
  - Fast Boot
  - Highly optimized

Research Conclusions

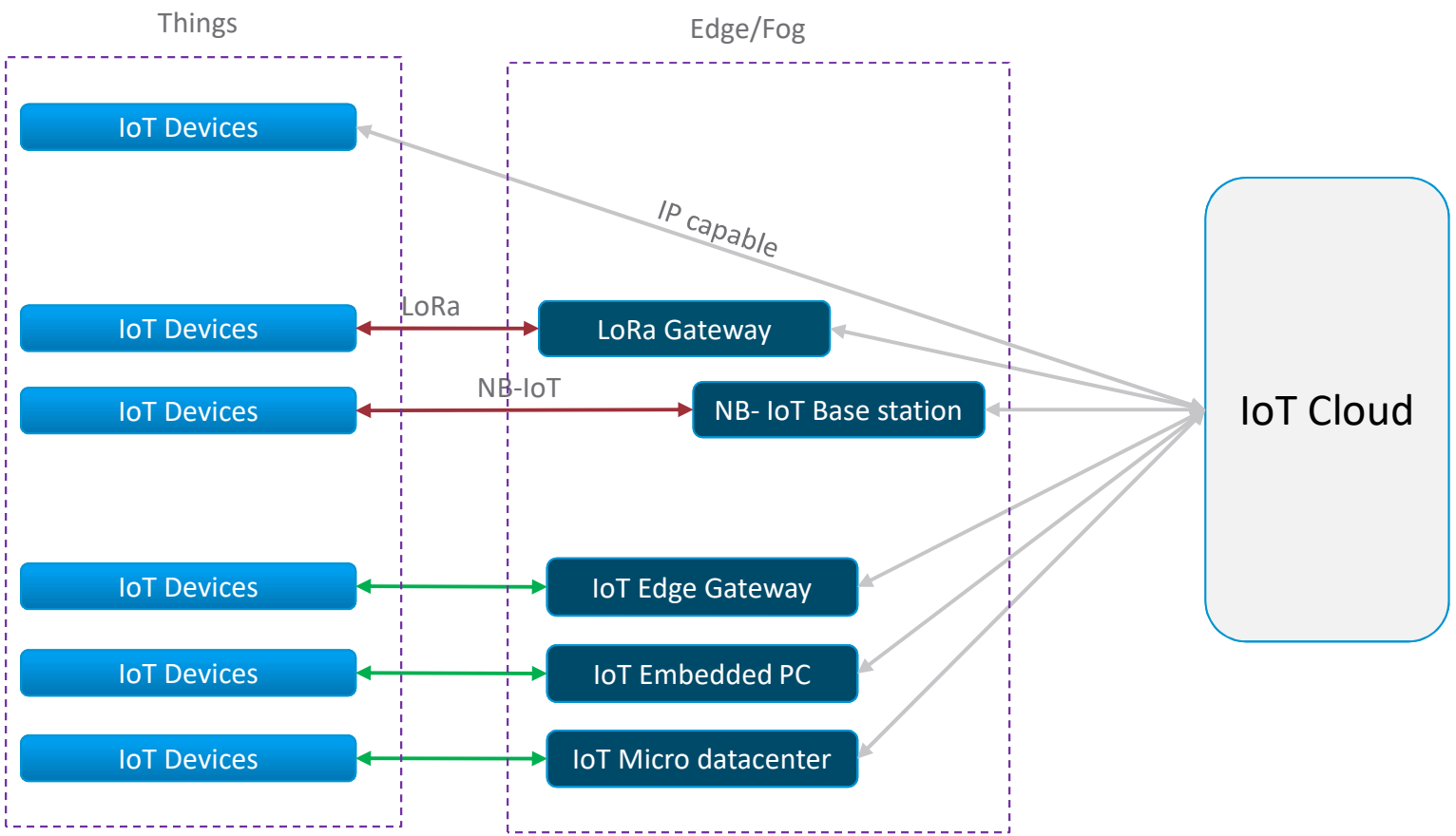
- **Status**
  - Unikernels really yield comparable performance.
  - Why existing unikernels have yet to gain large popularity?
    - Lack of compelling use cases
    - Compatibility with existing applications
    - Lack of production support (e.g. monitoring, debugging, logging)
    - *Lack of industry standard to Unikernels*
- **Conclusion**
  - Linux could be a good candidate of unikernels
    - Linux itself could help eliminate those challenges
      - Use cases & scenarios
      - Libraries
      - Tools and utilities
      - All optimizations and acceleration aimed to Linux can benefit unikernels
      - Fervent Linux community
    - UniLinux – Unikernelize Linux

vmware

Some Use Cases for UniLinux

- IO intensive applications
- Serverless
- Blockchain
- Machine Learning
- IoT

IoT Layout Architecture



vmware

Embrace the IoT through unikernels 1/2

- **IoT**
  - IoT Cloud
    - Microservice
      - Unikernels vs Container
    - Serverless
      - Data driven model & Container
  - IoT Edge/Fog
    - Pros
      - Unikernels have improved security
        - VM | smaller attack surface | short-lived lifecycle
      - Unikernels are fast small and quick boot
        - IoT devices/edge gateway/embedded PC
    - Cons
      - Unikernels mostly need virtualization technology
      - Unikernels don’t act specifically to those typical IoT characters
        - Power save, CPU Arches, Real-time requirement, etc.

Embrace the IoT through unikernels 2/2

- **Conclusion**
  - Unikernels can play for IoT
    - Virtualization will thrive at the edge/fog
      - Security issue
      - Multiple tenancy
      - Fragmentation
      - Enhanced Edge or Fog computing
    - IoT devices or platforms == Embedded systems
      - Linux already and always plays a very import role
    - Unikernelized Linux can run on bare metal instance easily
  - It's worth exploring unikernelized Linux in the case of IoT

What Could We Do?

Our target is to explore what is the best platform for running unikernels case

We will achieve this by

- **Research existing unikernels**
  - Integrate and support those major existing unikernels well
  - Integrate virtIO model into ESXi as an example
- **Build new unikernel**
  - Convert Linux kernel: UniLinux
- **Explore optimizations**
  - Provide monitoring, logging and remote debugging
  - Supporting a short lived unikernels instance
  - Resources are consumed by live unikernels

What Are The Key Challenges?

- **Convert Linux to unikernels**
  - The fundamental philosophy of Linux
    - Multiple processes
    - Two modes
  - Tightly coupled components
  - How to further improve performance
- **Reduce time of creating VM**
  - Snapshot
  - VMware Instant Clone
- **A good paravirtualized API for common unikernels**
  - Some existing pv ops
- **New scheduler**
- **Manage the lifecycle and identities of the provisioned unikernels**

vmware

How Could We Possibly Achieve This? Hypervisor basics

- **Support major existing unikernels**
  - Integrate virtIO framework into ESXi
  - Port PV drivers into them
- **Define a standard API which can paravirtualize unikernels**
  - Based on common hypercall
  - Configure/control guest OS
  - Setup Inter-VM Communication
  - Allocate/destroy memory directly
- **Add a new scheduler**
  - Address short lived unikernels VM
  - Schedule a group of unikernels instances

How Could We Possibly Achieve This? Linux basics 1/6

- **Convert Linux**
  - Single Supervisor mode
    - Force setting Ring 0
      - __USER32_CS | __USER_DS | __USER_CS
      - [GDT_ENTRY_DEFAULT_USER{32}_{CS:DS}]
    - Using Interrupt Stack Table (IST)
      - set_intr_gate_ist(X86_TRAP_PF, &page_fault, PF_STACK)
      - Interrupt and exception
    - Rephrase VDSO
      - Redirect as function call
    - *Manage Stack*
      - *Switch stack manually*
  - Single address space
    - Single process with multiples threads
      - No fork()

How Could We Possibly Achieve This? Linux basics 2/6

- **Convert Linux**
  - Optimization
    - Smaller size and footprint
      - Kconfig
      - Only keep necessary function call (system call)
    - Zero-copy
      - access_ok
      - {get,put}_user
      - copy_{from,to}_user
      - clear_user/strnlen_user/strncpy_from_user
    - Scheduler
      - scheduling classes & policies
        - fair vs rt vs deadline
    - Lightweight TCP/IP Stack
      - LWIP
      - Fastsocket
      - Seastar
      - …

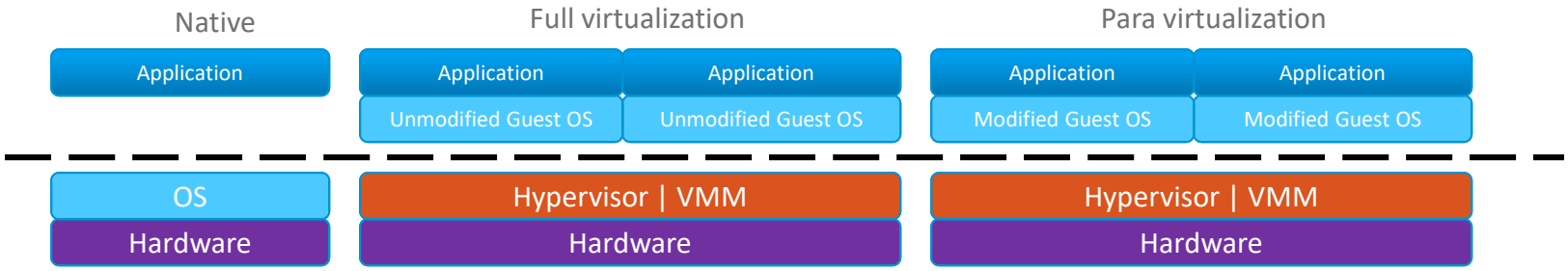
vmware

How Could We Possibly Achieve This? Linux basics 3/6

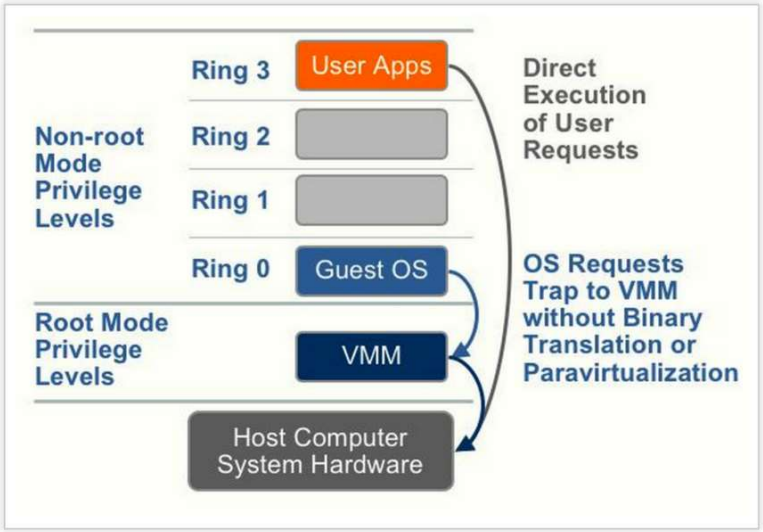
- **A variety of Unikernelized Linux Profiles**
  - Unikernelized Stand Linux
  - Unikernelized Secure Linux
    - SELinux
    - Grsecurity Linux
  - Unikernelized Preempt-RT Linux
    - Preempt-RT Linux
    - Potential use cases
      - airbag or break control in cars
      - flight systems in airplanes
    - Two scenarios
      - Bare metal environment
      - Virtualization environment

How Could We Possibly Achieve This? Linux basics 4/6

- **Virtualization Architecture**
  - Native vs Full vs Para virtualization

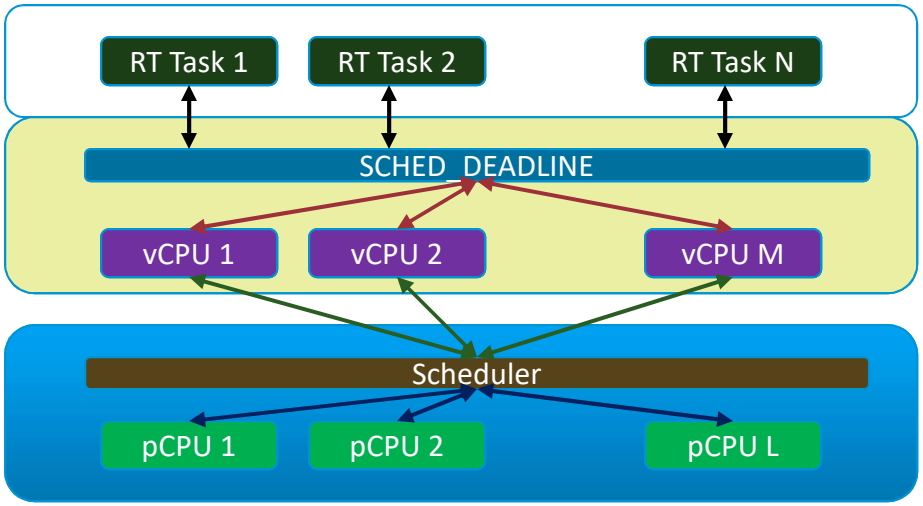


  - Hardware assited virtualization



How Could We Possibly Achieve This? Linux basics 5/6

- **Real Time (HW) virtualization Architecture**
  - There are two levels of the hierarchical scheduling structure.
    - OS + Hypervisor



Preempt-RT Linux

Hypervisor

How Could We Possibly Achieve This? Linux basics 6/6

- **Unikernelized Preempt-RT Linux & Virtualization**
  - Challenges of how to still guarantee such a correct timing behavior
    - Two levels scheduling structure
    - Hypervisors have no knowledge of tasks within VM
    - Memory management
    - The lock-holder preemption problem
  - What could we do?
    - A real-time VM with Unikernelized Preempt-RT Linux is limited to one RT task
      - One process with multiple threads with SCHED_DEADLINE
    - vCPU = pCPU
      - Para-virtualization with hypercall
        - vCPU cannot be preempted out
    - Page allocation
      - Para-virtualization with hypercall
        - Reserve physical memory pool
    - Direct interrupt

How Could We Possibly Achieve This? Compatibility

- **Support existing applications**
  - Different code circumstances
    - Source code
      - New standard library like glibc
      - Function Call
    - Binary
      - –shared –pic
        - LD_PRELOAD | ld.so.preload
      - Others
  - Multiple processes
    - One fork = one UniLinux instance
      - IPC = Inter-VM Communication
    - PCID – Process-context identifiers
      - Limited bits
      - Linux's own debug/monitor/log tools and utilities
    - *uClinux*
      - *Multitasking without an MMU*

vmware

How Could We Possibly Achieve This? Debugging, monitoring and logging

- **Debug unikernels**
  - Log info
    - virtual serial port
    - Dynamic buffer memory allocation
  - Linux's own utilizes
    - ssh/gdb/ftrace/perf/kprobe/kdump/…
    - PCID & the balloon driver
- **Monitor unikernels**
  - A mini-httpd as a stub connecting those Linux utilities
    - Inspired by OSv
- **Log unikernels**
  - rsyslog
  - vRealize Log Insight

How Could We Possibly Achieve This? Enhancements

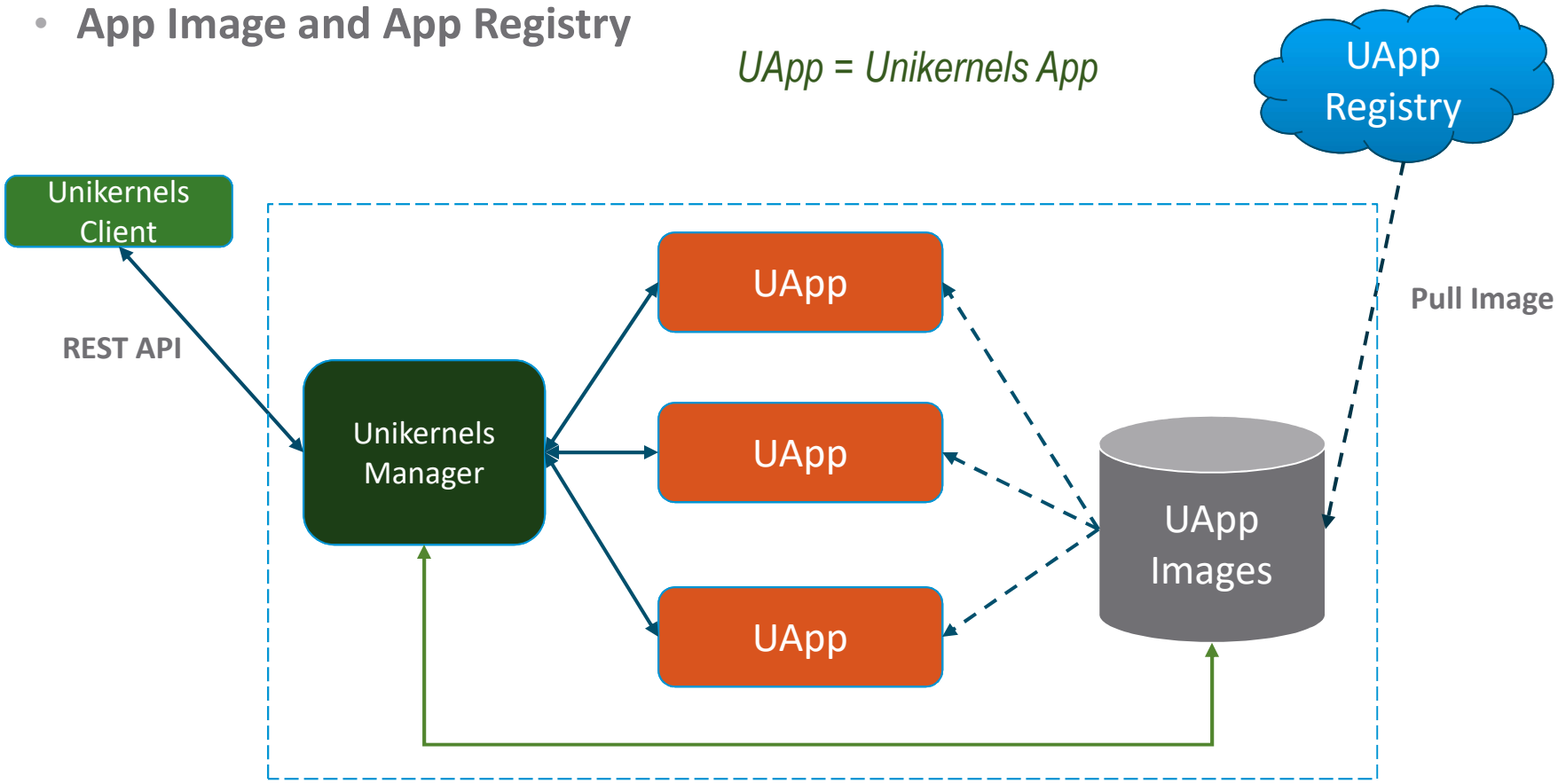
- **Offer faster boot**
  - Explore ESXi to further reduce the time of creating VM
  - Skip BIOS with a small integrated bootloader
  - Replace ACPI with DTB
  - Adopt 1:1 Bus/device initialization
    - No any redundant bus scanning and device probing
- **Utilize hardware virtualization**
  - VT-X Instructions
    - VMFUNC
      - Pre-construct EPT table to get a faster and secure way to communicate between unikernels
  - VT-X Features
    - VPID (Virtual processor ID)
      - The tagged TLB to reduce cost of performance
    - Preempt Timer
      - A feature which count down in unikernels without too much external timer injected by hypervisor

How Could We Possibly Achieve This? Others

- **Construct an efficient toolchain**
  - Build and deploy Unilinux like Docker
  - Customized components management
    - Configuration
    - Basic Kernel image
    - User App
    - Dependencies
- **Support orchestration**
  - **Unik**
- ***Integrate Source Code Analyzer tool***
  - *This can help us enhance security from code level*

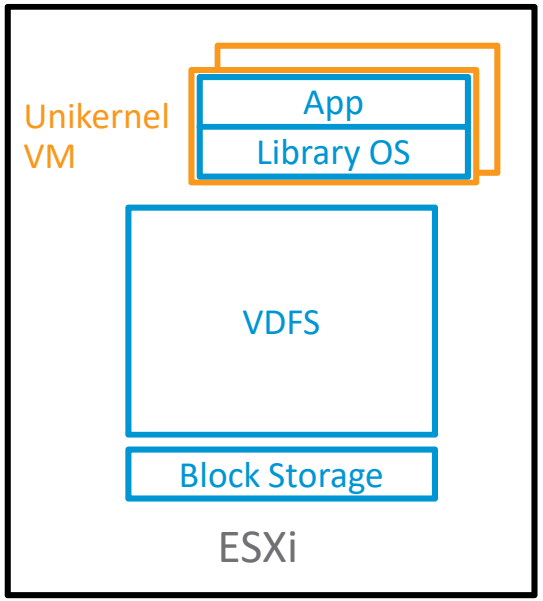
How Could We Possibly Achieve This? Management

- Unikernels Manager
- App Image and App Registry



How Could We Possibly Achieve This? VDFS

- VDFS is a hyper-converged distributed file system with modern features
- Key features:
  - POSIX compliant
  - Support advanced features:
    - Distributed and scale-out
    - Zero config (no network to manage)
  - Multi-backend
- VDFS is an ideal platform for unikernels
  - Zero config
  - Shared file system cache
  - No need to manage disk images for unikernels



Hypervisor + Unikernel VMs

vmware

Summary

- **Unikernels Overview**
- **Those existing unikernels have yet to gain very large popularity. UniLinux probably can boost unikernels.**
- **Preempt-RT UniLinux is very well suited for IoT.**
  - IoT Cloud
  - IoT Edge + IoT Fog

References

- http://unikernel.org/projects/
- https://wiki.xen.org/images/3/34/XenProject_Unikernel_Whitepaper_2015_FINAL.pdf
- https://www.linux.com/news/7-unikernel-projects-take-docker-2015
- https://www.usenix.org/node/184012
- https://www.deepdyve.com/lp/institute-of-electrical-and-electronics-engineers/includeos-a-minimal-resource-efficient-unikernel-for-cloud-services-J43NrzQ7fn
- https://www.usenix.org/system/files/conference/nsdi14/nsdi14-paper-martins.pdf

Thank You!

tiejunc@vmware.com

vmware