Cloud Native

Microservices Best Practices

Yalun Lin Hsu

2019.05.14

Outline

Why Microservices?

Introducing Cloud Native

Unikernel & LightVM

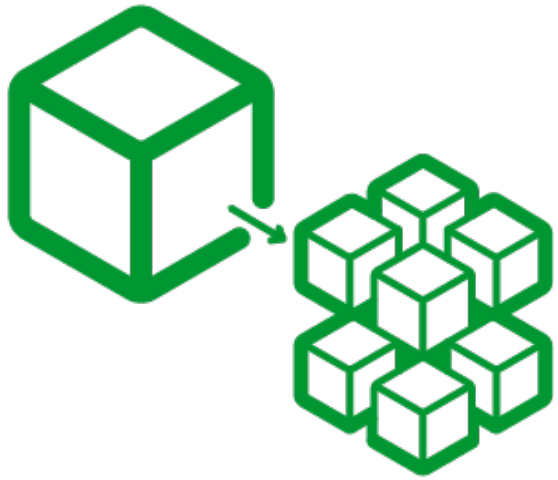
References

Why Microservices?

Why Microservices?

Microservices have been getting more and more popular in recent years

Why Microservices?



Motivations, Issues and Benefits for Migrating to Microservices

Motivations

- Maintainability
- Delegation of Team Responsibilities
- Fault Tolerance
- Because everybody does
- Devops Support

Issues

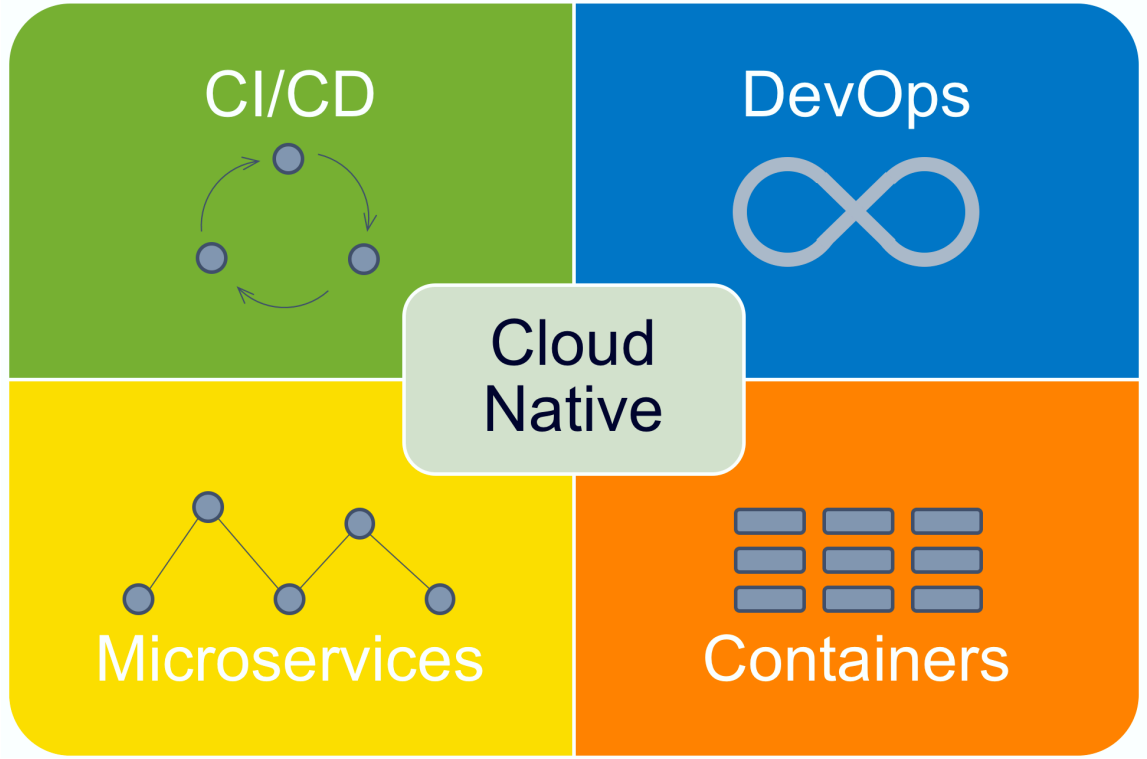
- Database Migration and Data Splitting
- Effort Estimation and Overhead
- People' s mind
- Effort Required for Library Conversion
- Effort Required for the DevOps Infrastructure

Benefits

- Maintainability Improvement
- System Understandability
- ROI
- Architectural Complexity Reduction
- Simplifies Distributed Work

Introducing Cloud Native

Cloud Native Features



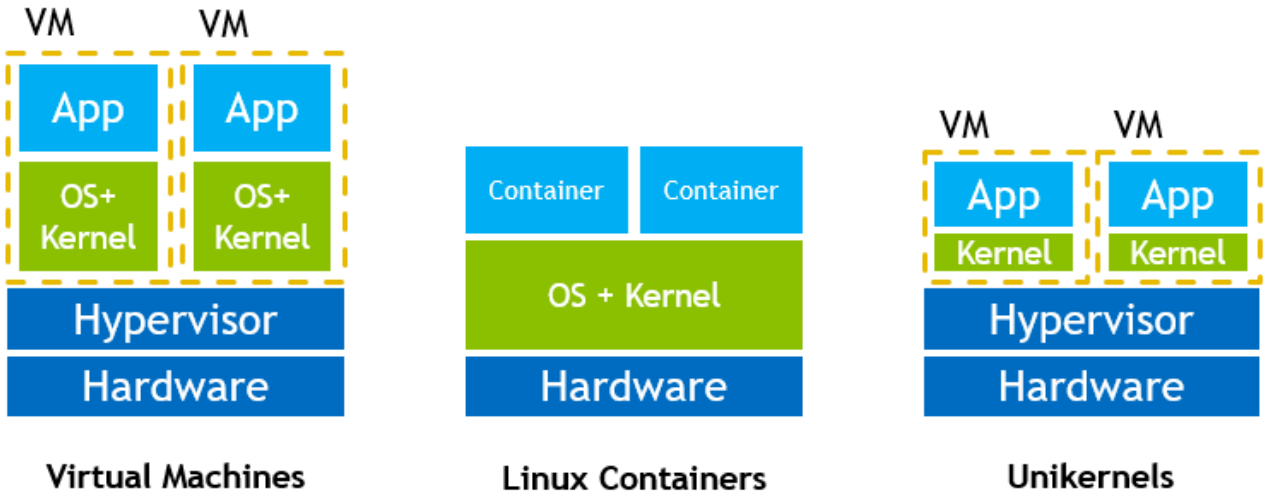
A Cloud Native Example

Open Issues in Scheduling Containers

- Application Topology Specification and Composition
- Performance Characterization and Isolation
- Microservice Monitoring

Unikernel

What is Unikernel ?



Unikernels are specialised, single-address-space machine images constructed by using library operating systems.

Unikernel Features

- Small
- Fast
- Safe

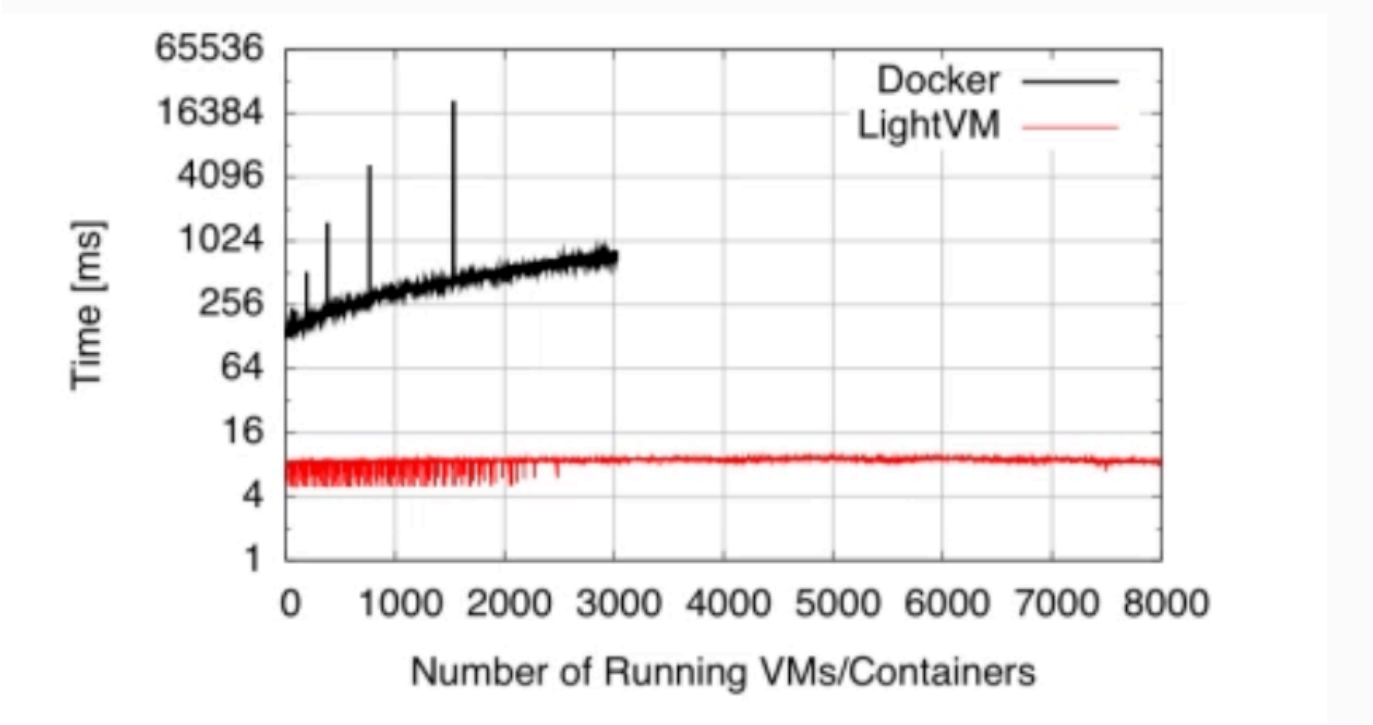
Container VS Unikernel

Image Size	Dozens MB ~ Hundreds MB	KB level
Boot time	Several to dozens seconds	Ms level
Feature	Mul containers in one kernel	Every process has one build-in kernel
Environment	LXC	Hypervisor
Address space	Isolation with different containers address space	Single address space
Process	One container can run a set of processes	Only one process in a unikernel
Ecosystem	Easy to build, mature ecosystem	Difficult to build, immature ecosystem

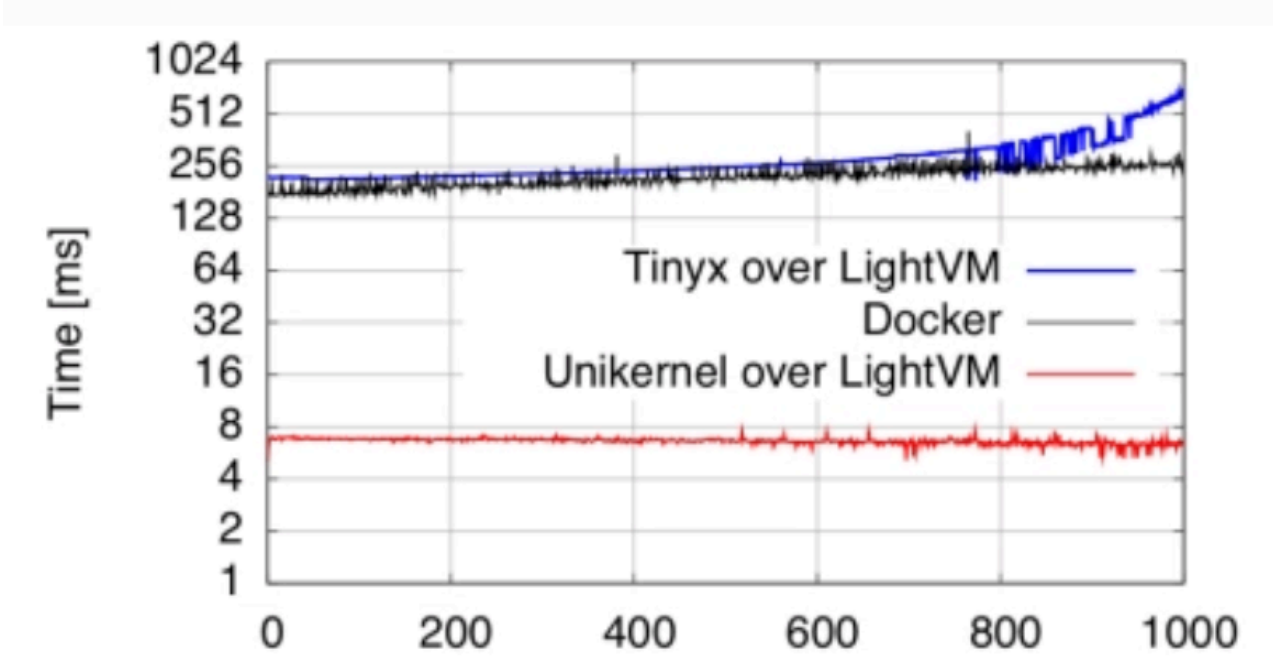
LightVM

Can we have the improved isolation of VMs, with the efficiency of containers?

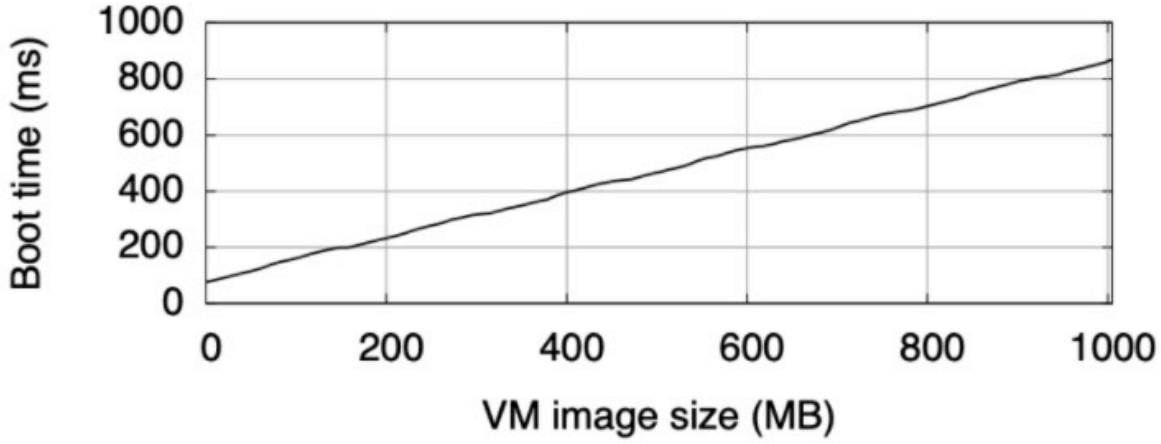
Hello, LightVM!



Hello, LightVM!



About boot time

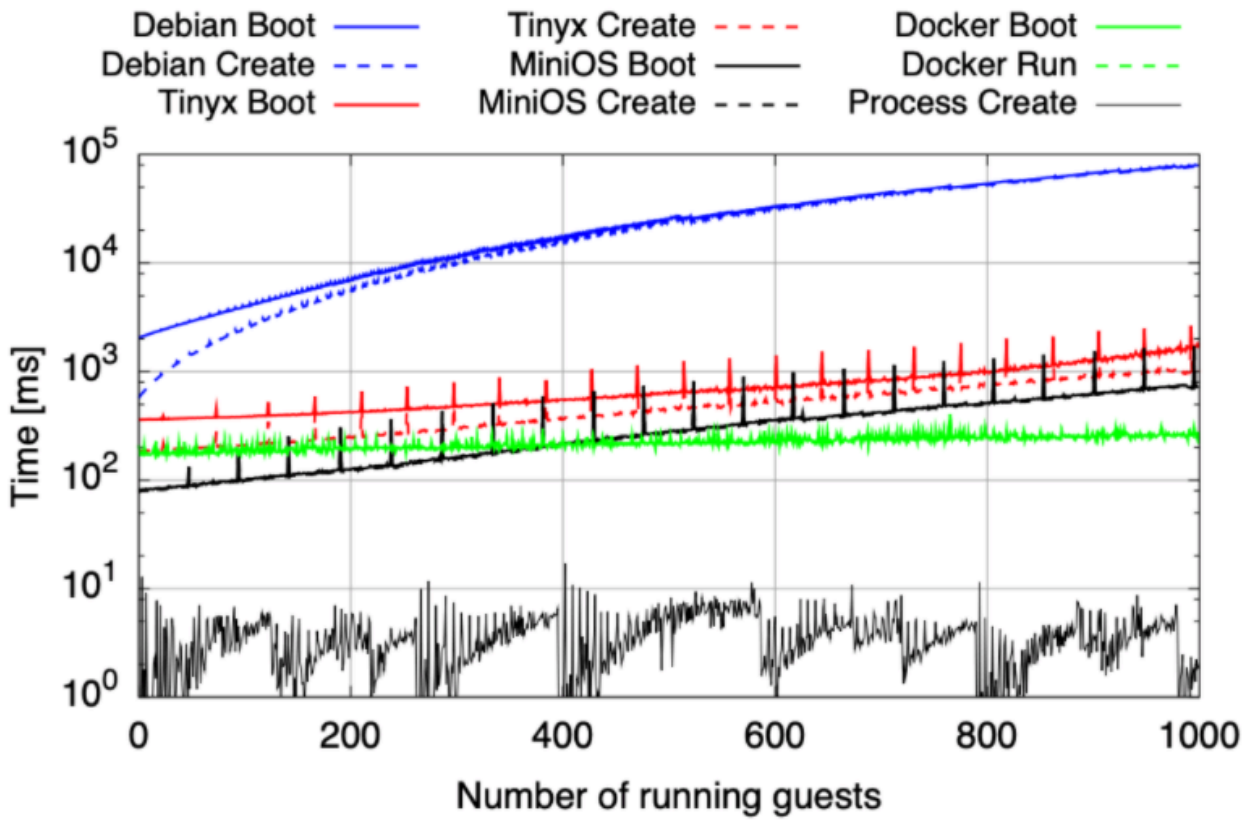


TinyX

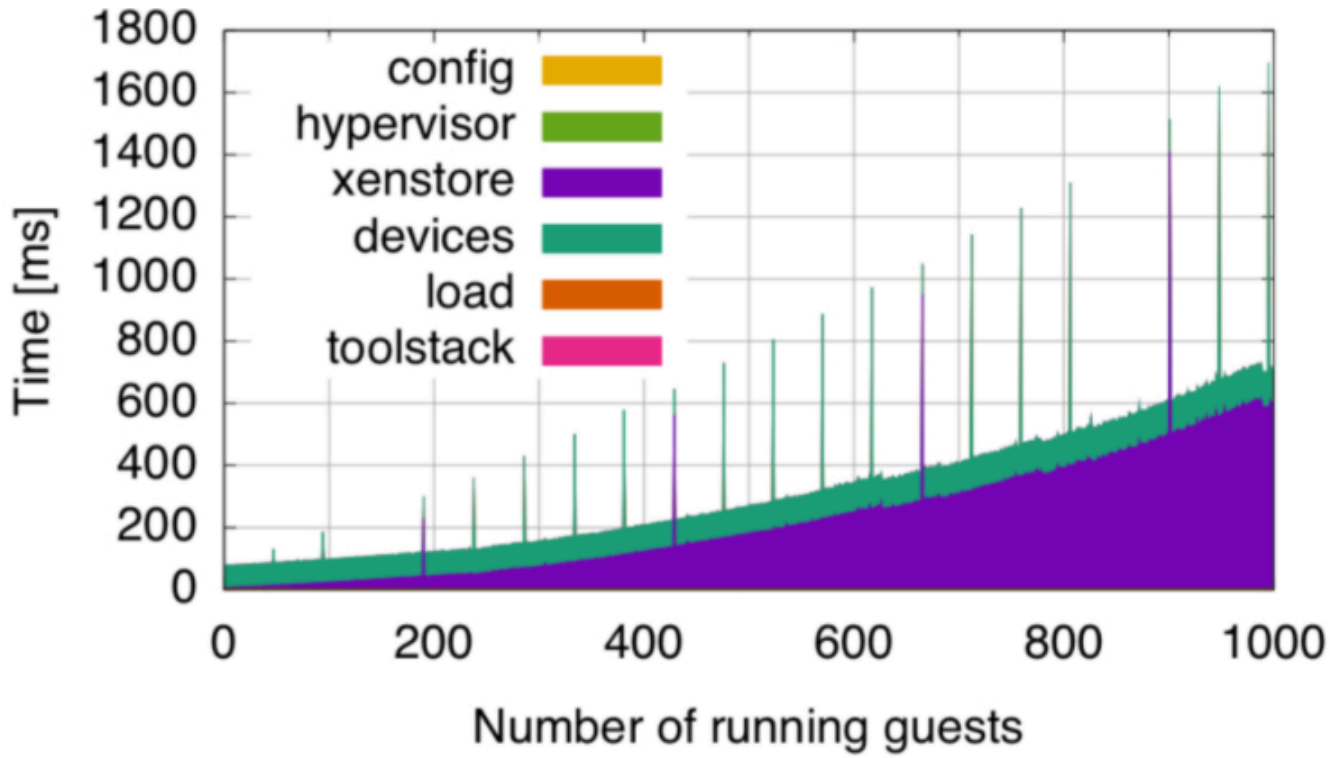
- Use *objdump* command
- Debian package manager

Tinyx creates kernel images that are half the size of typical Debian kernels, and have significantly smaller runtime memory usage (1.6MB for Tinyx vs 8MB for Debian).

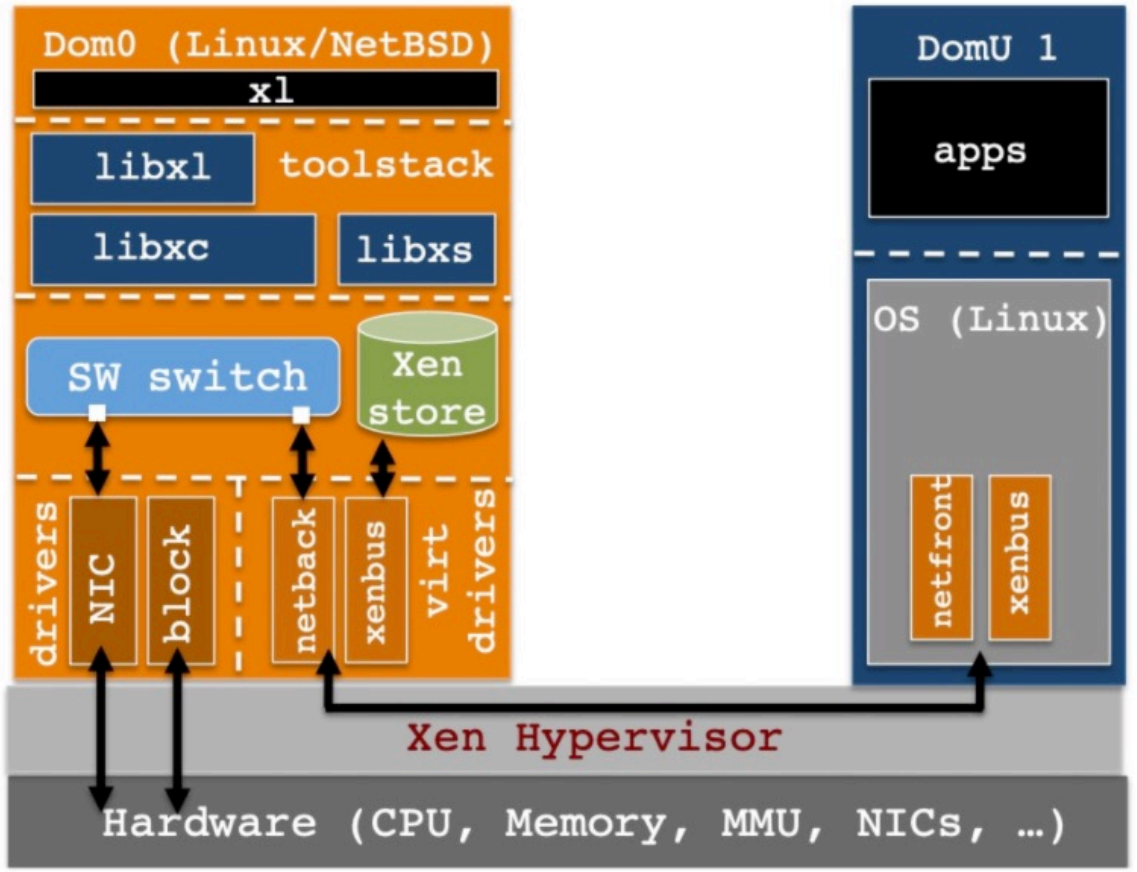
About create time



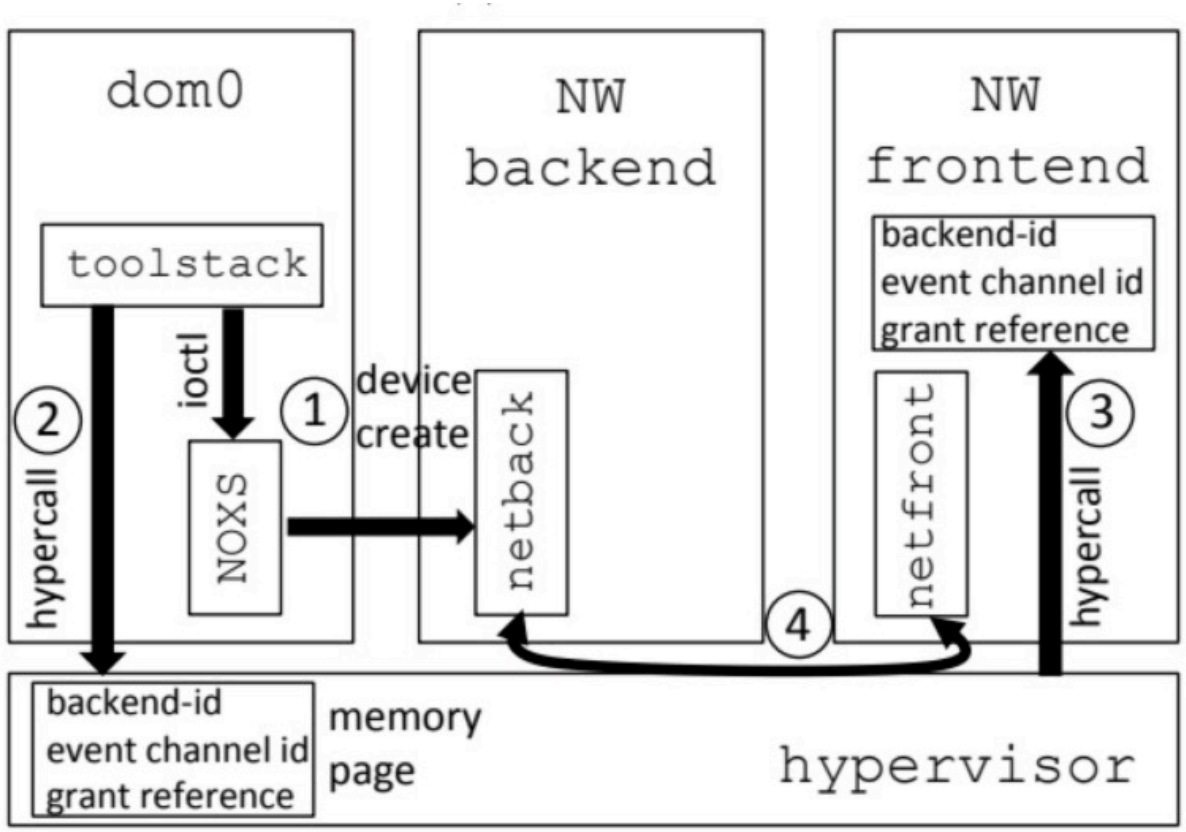
About create time



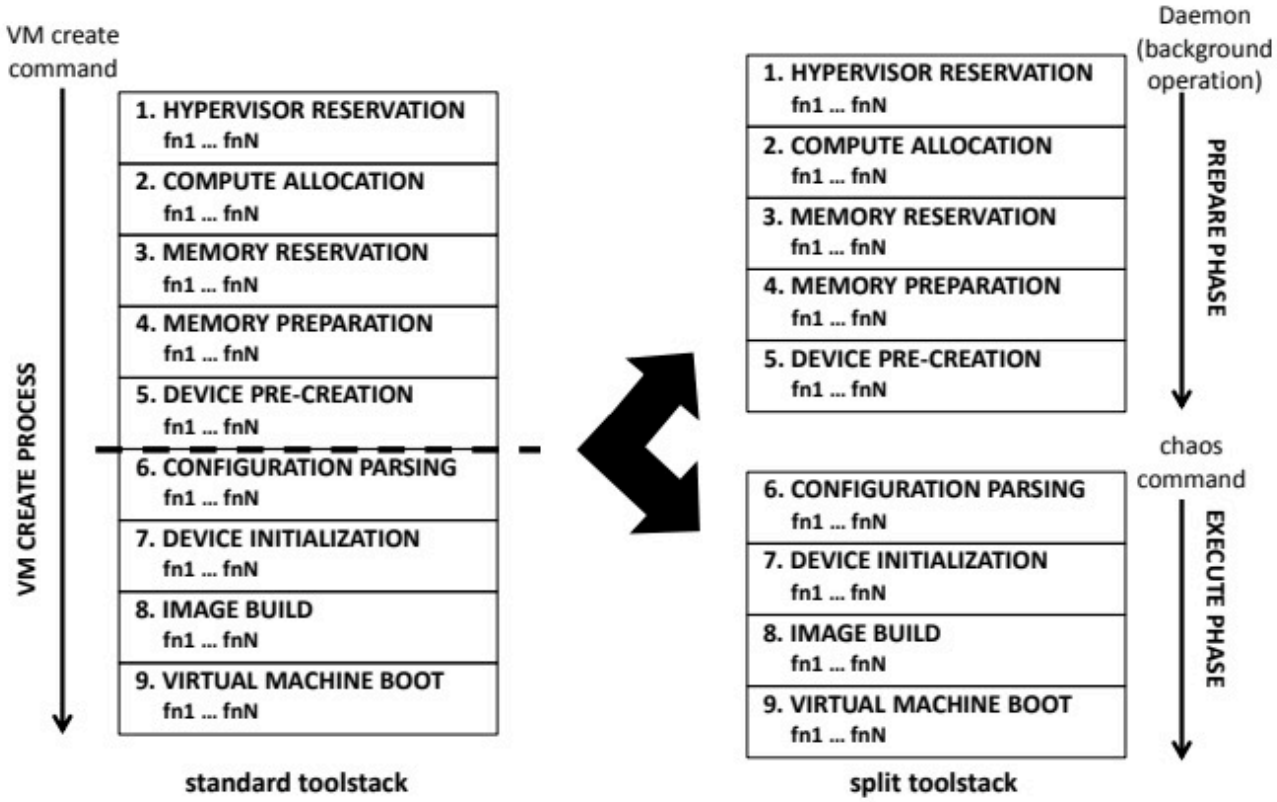
Xen Architecture



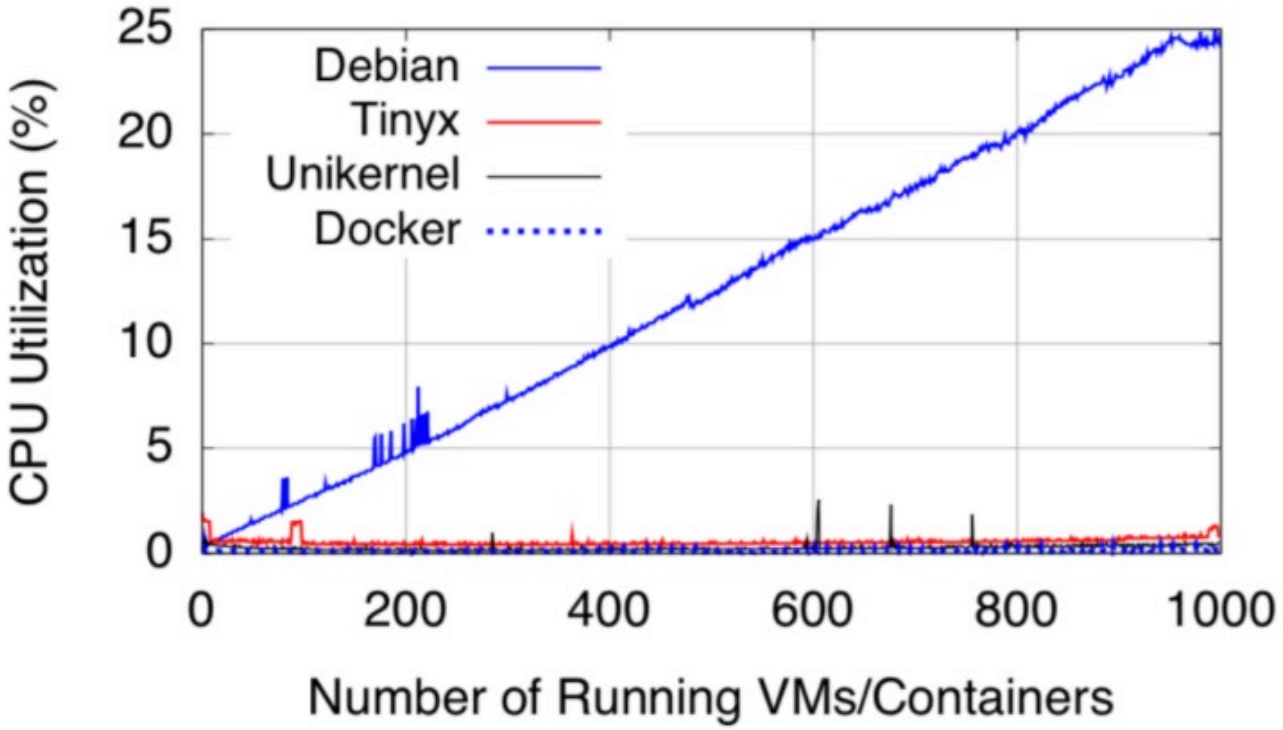
Noxs


dom0
NW backend
NW frontend
toolstack
ioctl
1
device create
2
hypercall
NOXS
netback
backend-id
event channel id
grant reference
netfront
3
hypercall
4
backend-id
event channel id
grant reference
memory page
hypervisor

Toolstack split



Conclusion


CPU Utilization (%)
25
20
15
10
5
0
Debian
Tinyx
Unikernel
Docker
0
200
400
600
800
1000
Number of Running VMs/Containers

References

References

Processes, Motivations, and Issues for Migrating to Microservices Architectures
IEEE Cloud Computing, 2017

Microservices Architecture Enables DevOps
IEEE Software, 2016

Open Issues in Scheduling Microservices in the Cloud
IEEE Cloud Computing, 2016

Unikernels vs Containers: An In-Depth Benchmarking Study in the Context of Microservice Applications
SC2, 2018

My VM is Lighter(and Safer) than your Container
SOSP, 2017

Thanks