No More Turtles

Alternative to Nesting

Mengmei Ye (mye@ibm.com), **Angelo Ruocco (ang@zurich.ibm.com)**,
Daniele Buono (dbuono@us.ibm.com), James Bottomley (jejb@linux.ibm.com),
Hubertus Franke (frankeh@us.ibm.com)

Industry problem

- There's an increasing need from kubernetes customers to spawn VMs:
  - Kubevirt: VM-based workloads scheduled through Kubernetes
  - kata-containers et al: increase container boundaries through VMs
- This creates a significant problem in common cloud-based kubernetes deployments, where worker nodes are themselves deployed in VM

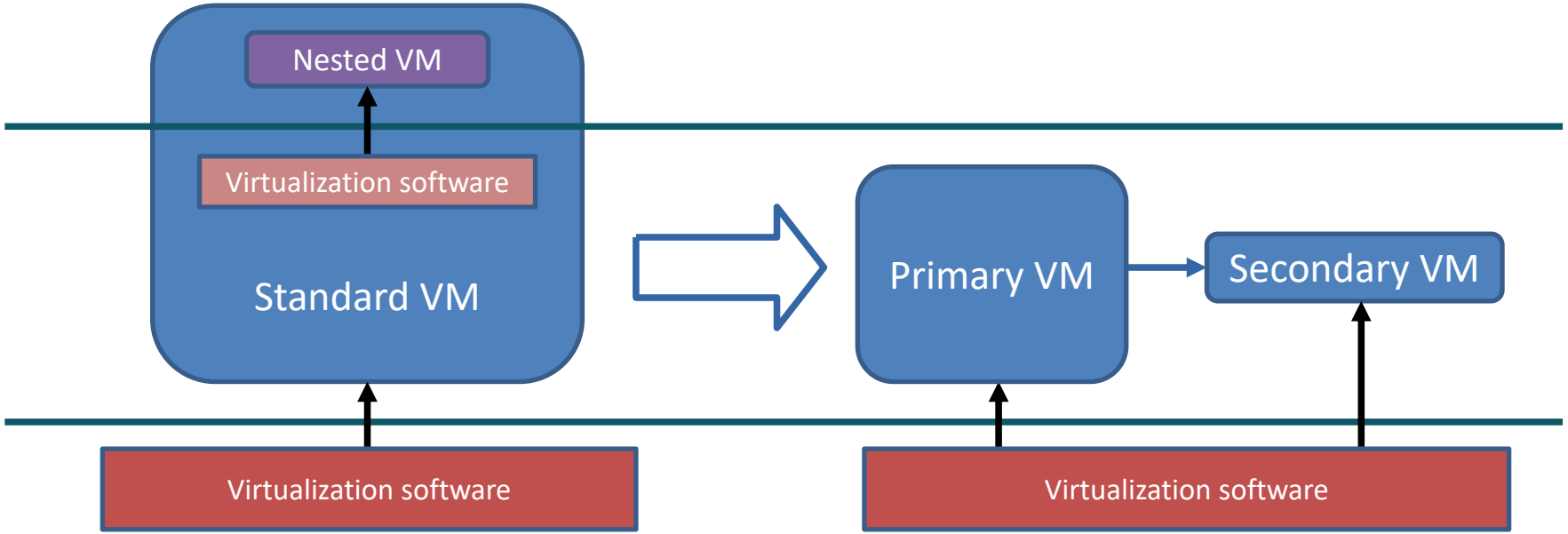
Present options and drawbacks

1. VMs on Baremetal systems
   - Expensive
   - Very low customizability
2. Nested VMs
   - Confidentiality (TDX/SEV/PVM encryption unavailable)
   - Security (large codebase for nested = higher chance of bugs)
   - Low performance
   - Not allowed by most Cloud Providers

Solution – Flatten the hierarchy

- A (Primary) VM is able to ask the host to spawn a (Secondary) VM
- The Primary VM is able to access the Secondary VM
- The Primary VM has some basic control over the Secondary VM

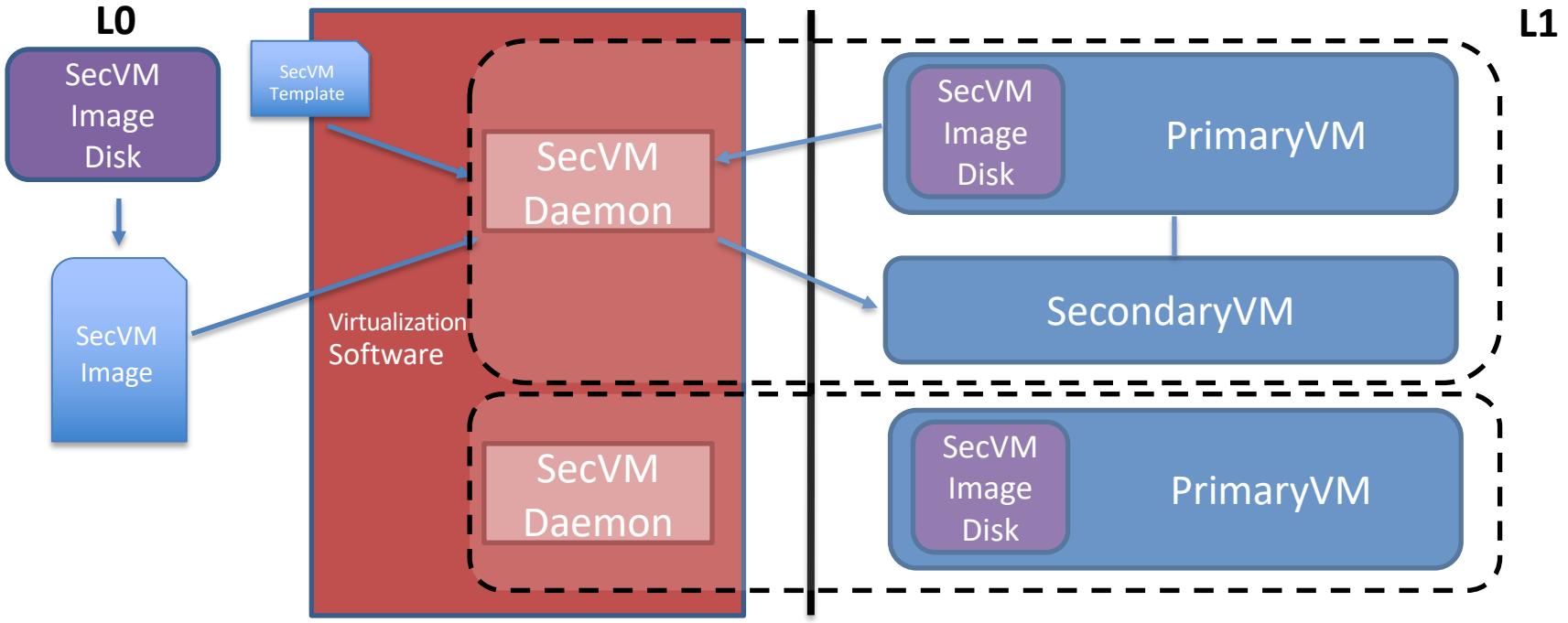
Solution – Flatten the hierarchy



Flatten the hierarchy – Challenges

- Security
  - The rest of the system should not be affected by secondary VMs. Resources must be carved out from the Primary VM
  - Primary VM only has access to some pre-defined actions. The control plane is kept in L0
- Isolation
  - Communication channel Host <-> Primary <-> Secondary has to be only accessible by the Primary VM
  - Secondary VMs must be invisible to VMs in other namespaces

Solution – Flatten the hierarchy



Solution – Flatten the hierarchy

Std VM + Nested	**Primary + Secondary VMs**	**Baremetal + Std VMs**
Cheap	Cheap (?)	Expensive
High flexibility	Medium flexibility	Low flexibility
No encryption	Encryption	Encryption
No true device passthrough	True device passthrough	True device passthrough
Slow(ish)	Fast	Fast

The Secondary VM Daemon

- Talks to the Primary VM via VSOCK
- Controls the Secondary VMs
  - Create
  - Modify
  - Destroy
  - Show/List

Enforcing the limits – Cgroup

- Primary VM, Secondary VMs and Daemon live inside a cgroup with memory and cpu limits

Enforcing the limits – Storage

- Primary VMs have an additional disk where they put the images for Secondary VMs
- The disk gets unplugged from the primaryVM, mounted on host, the images is copied, and disk shrunk to new size before re-plug

Enforcing the limits – Network

- A virtual network is created for each Primary - Secondary VMs partition.
- All the requests go to the same physical interface accessible by the Primary VM

Proof of Concept – Why Libvirt

- Open Source
- Supports multiple hypervisors
- Uses cgroups via systemd integration, easy to add implementation for limits enforcement
- Easy way to create and add virtual networks
- Easy, standard way to attach-detach devices at runtime

Secondary VM – Libvirt

DEMO

Future work

- Define standard APIs Host <-> Primary VMs
- Clean up – upstream code
- Improve cgroup <-> libvirt synergy
- Improve PrimaryVM isolation in host
  - Improve cgroup cpuset
  - Enforce guarantees over shared resources
- Evaluate alternative storage solutions

KVM FORUM