Few Words About UniKernels

Luca Abeni
luca.abeni@santannapisa.it

The Cloud Revolution

- Cloud computing: large usage of VMs
  - Even 1 VM per application!!!
  - Single-application OSs?
  - Not general-purpose OSs...
- Traditional approach: run a GPOS in the VM, and install the application
- Dynamic workloads: some times, VMs must boot very quickly...
  - Optimization of VM and OS for virtual environments, quick boot, ...

An OS for the Cloud

- GPOS in a VM: simple solution
  - But maybe it is overkill...
  - Do we really need a whole OS just to run an application???
  - Lots of unneeded user-space applications and libraries...
  - Lots of unneeded kernel features / drivers / functionalities...
- A VM provides a protection domain
  - So, do we need address-space protection inside the VM?
- The OS kernel abstracts the hardware
  - Why do we want to abstract a VM?

A GPOS for the Cloud

- The OS kernel provides virtual filesystem, network stack, etc...
  - Do we need these subsystems even if the application does not use network or filesystems?
- The OS provides flexible boot scripts, shells, many commands, etc...
  - Do we need all of them just to run a single application?
- Result: we might need a $1GB$ filesystem image just to run a web server...
  - Not to talk about memory footprint, CPU usage, etc...

A Possible Solution: Containers

- The issues mentioned above caused the raise of contaiers
  - No need to run a different OS kernel
  - No need to boot a whole OS just to run an application...
  - ...Just put the application and its dependencies in the container!
  - Reduced VM boot times...
- Does this mean that hardware-level virtualization is doomed?
  - Isn't it possible to reduce the overhead while still running a guest OS kernel, etc?

UniKernels to the Rescue

- Very simple idea: remove all the unneeded stuff!
  - Redesign the whole gues OS to be special-purpose, for our application
- Application directly linked to the kernel code (library OS)
  - Does it sound familiar? Uh... What is the difference with a real-time executive?
  - Also, remember the “vertically structured OSs” (exokernels)
- Everything runs in the guest kernel space (protection is provided by the VM / hypervisor)
- Do not link / include unneeded kernel components and drivers

Minimizing the VMM too

- If the guest OS is designed to run in a VM, no need to implement an exact copy of a physical machine!
  - The host/guest interface can be redesigned
  - No virtualization of physical devices, but only paravirtualized devices (virtio, ...)
  - Simplified guest boot (the VMM/hypervisor is the boot loader!)
- Hypervisor/VMM/DM can be greatly simplified!
  - Better performance/lower overhead
  - More secure
- This lead to the rise of “MicroVM”s

Taking it to the Limit...

- Unikernel: designed to run in a VM
  - Application directly linked to the kernel (only the needed parts!)
  - Drivers only for virtual devices
  - Paravirtualization
  - ...
- What about (re)designing a VMM / hypervisor to only run unikernels?
  - Simplified host / guest interface
  - Do not virtualize unneeded devices
  - ...
- Advantages: simpler code, reduced boot time, better performance, ...