

IO virtualization

Michael Kagan

Mellanox Technologies



- **Mission – non-stop services to consumers**
 - Flexibility – assign IO resources to consumer as needed
 - Agility – assignment of IO resources to consumer when needed
 - Isolation – deliver IO services to consumer without interference
 - Fault-tolerance – deliver IO services despite HW failures

Reliability, flexibility & efficiency

- **Mean – de-couple service delivery from the Hardware**
 - Overcome Hardware boundary constrains
 - Resource assignment is a matter of mouse-click
 - Consumer-transparent hardware replacement

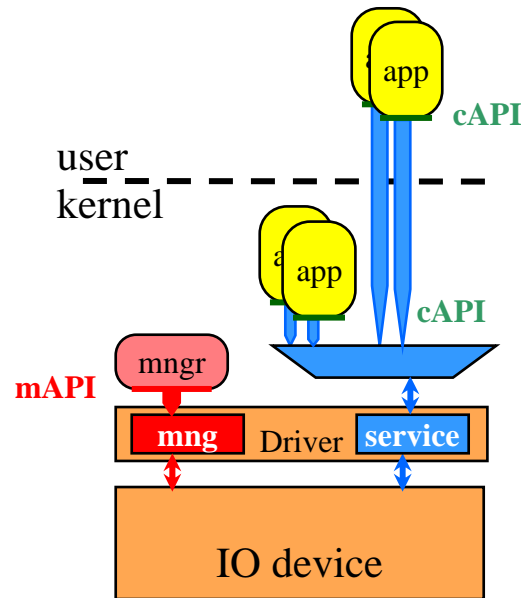
De-couple consumers' execution environment from underlying Hardware

Simple (monolithic) IO model



cAPI – soc, MPI etc

mAPI – PCI cfg, pwr etc



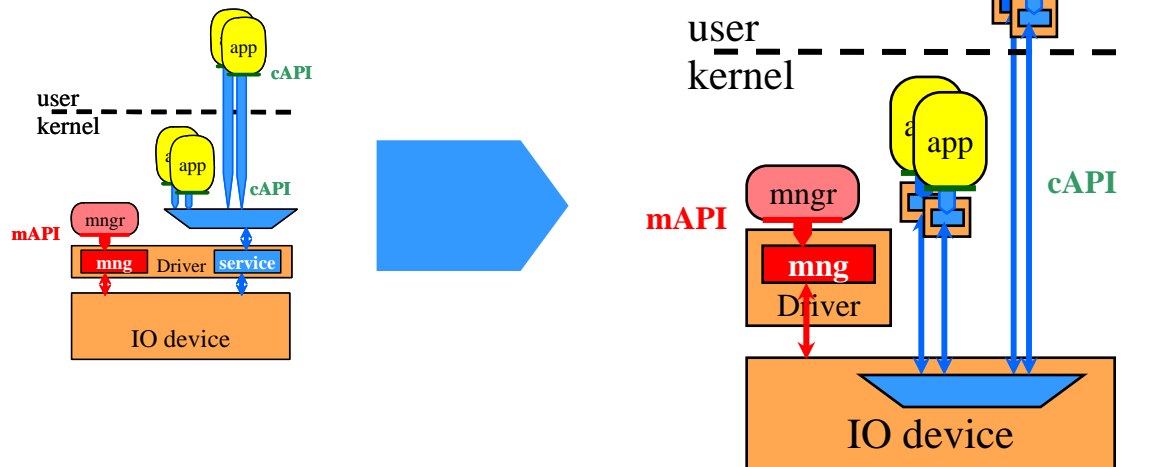
- Kernel-resident mngr owns HW
 - Initialization, configuration
 - Management interface (API)
- OS is part of IO subsystem
 - Isolation, protection, QoS
- Constrained performance
 - Kernel call, data copy etc.
- Low QoS granularity
- Agility? Flexibility? Fault tolerance? ☹
- Dumb IO device

Advanced IO model



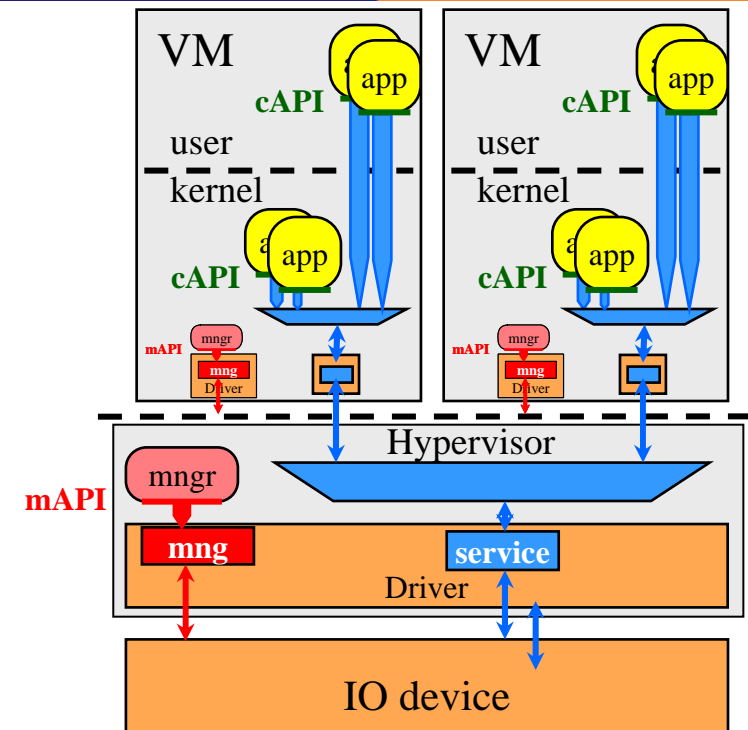
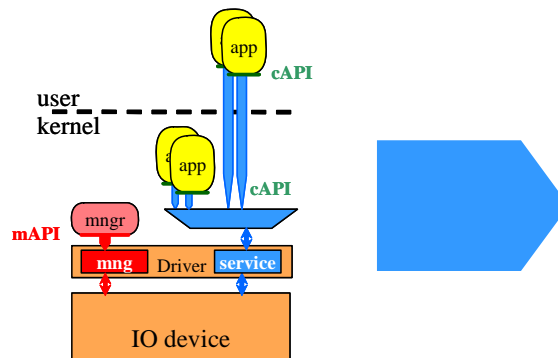
cAPI – soc, MPI etc

mAPI – PCI cfg, pwr etc



- Kernel-resident mngr owns HW
 - > Initialization, configuration
 - > IO delivery policy provisioning (QoS, protection)
- IO delivered to consumers via IO channels – OS is not a part of IO subsystem
 - > Abstract interface to post IO operation requests
 - > HW-enforced cross-channel protection and isolation
 - > Unlimited scalability
- High performance
 - > Kernel bypass, zero copy etc.
- Fine QoS granularity
- Agility? Flexibility? Fault tolerance? ☹
- Smart IO device

IO in virtual machines – take 1



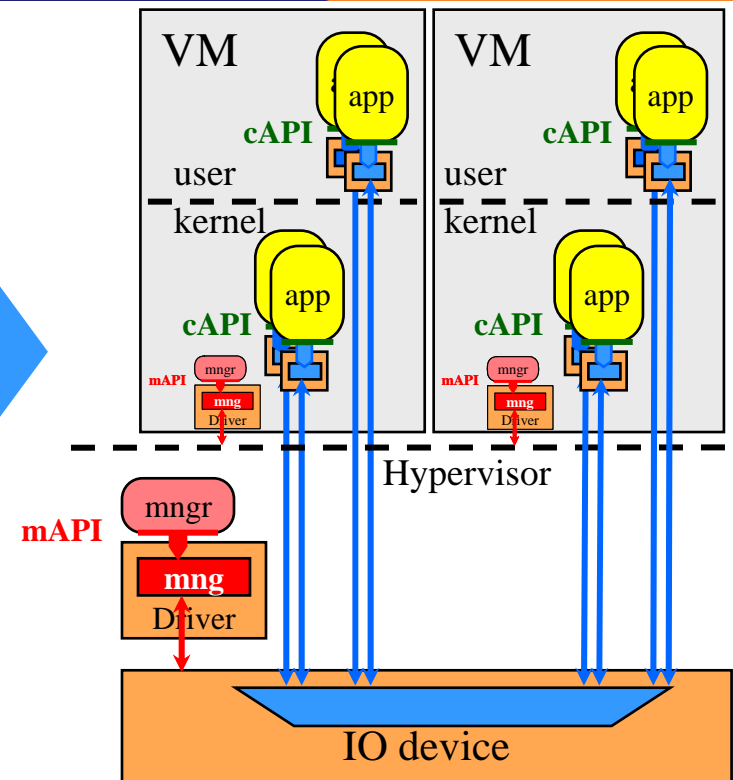
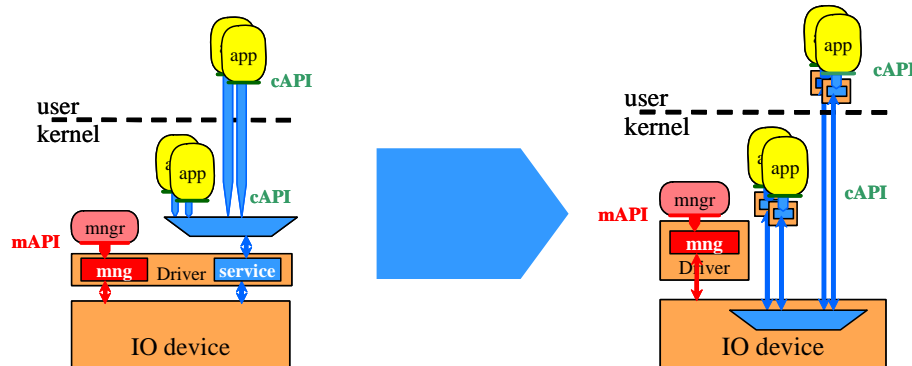
- Hypervisor-resident mng owns HW
 - IO delivered to guest via IO device emulation
 - “mng” part is faked; “service” path used for IO operation
- Improved agility, flexibility and fault tolerance
- Additional SW layer on the IO service path
 - Cross-domain isolation and protection in Hypervisor
- Virtualization cost – Guest OS and Hypervisor are part of IO subsystem
 - Lower IO performance
 - Higher CPU overhead
 - Limited scalability – architecture tied to HW
 - Low QoS granularity
- Dumb IO device

IO in virtual machines – take 2



cAPI – soc, MPI etc

mAPI – PCI cfg, pwr etc



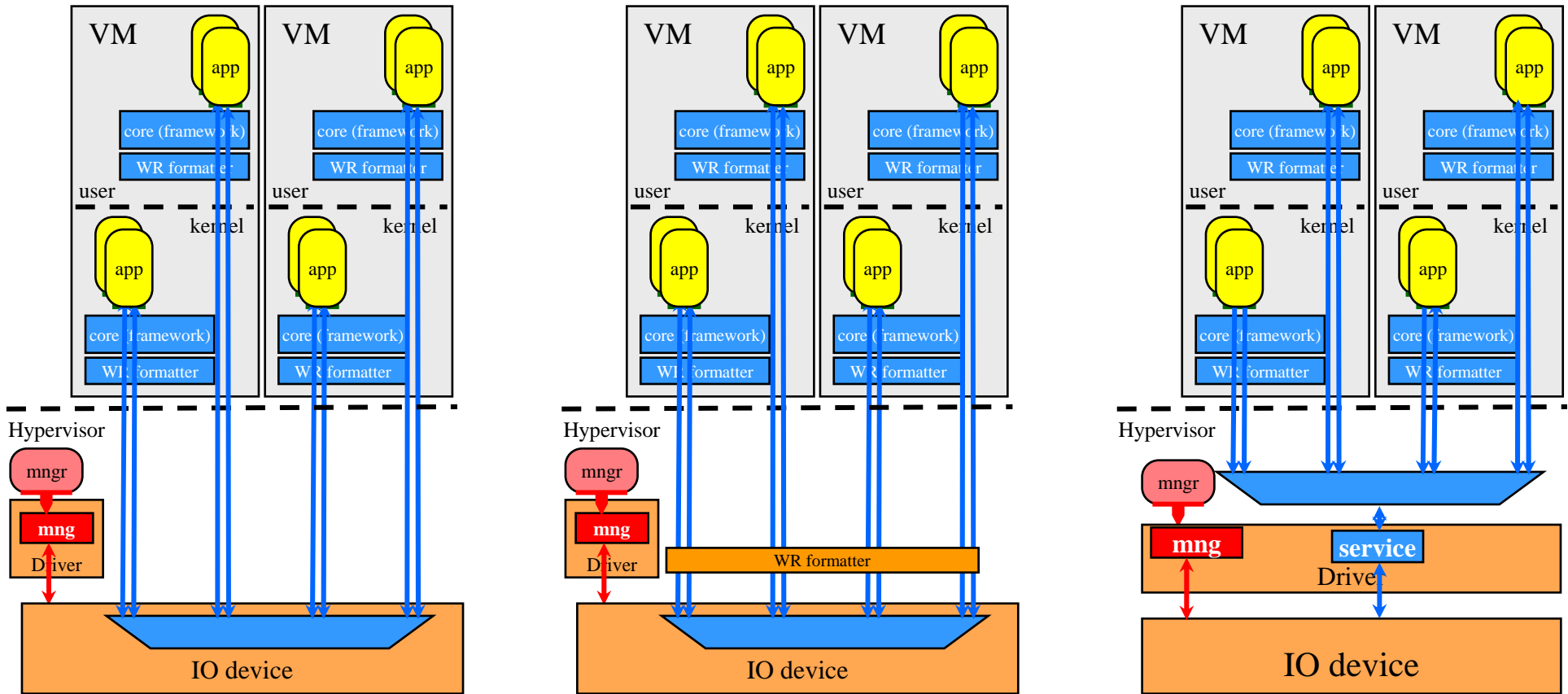
- Hypervisor-resident mngr owns HW
 - IO device(s) configuration, IO policy provisioning
- IO delivered to consumers via IO channels
 - IO delivery is an integral part of the platform
- Uncompromised IO services delivery
 - Unlimited scalability (“IO channel for every socket”)
 - HW-enforced isolation and protection
 - Direct IO channel access (pass-through)
 - Hypervisor offload on data path, zero copy etc.
 - Fine QoS granularity
 - Allows on-demand resource allocation
- Standard IO interface to enable transparent servers’ management
- Smart IO device

Open Fabrics Channel IO Interface



- **Industry-standard channel IO interface**
 - Included in Linux kernel
 - Endorsed by Microsoft
- **Supports multiple IO devices**
 - InfiniBand HCAs
 - Ethernet NICs
- **Standard IO provisioning interface**
 - Assign QoS for an IO channel or group of channels
- **Efficient and versatile IO delivery**
 - Enables direct access to IO HW – full Hypervisor offload
 - Raw datagram IO channels
 - Transport-offloaded IO channels
 - RDMA IO channels
- **Unlimited scalability**
 - Unlimited number of IO channels

Channel IO in virtual machines



Pass-through

Hypervisor acceleration

Simple IO device

- **Hardware-independent consumer-transparent guest migration**
 - Support direct access to HW

- Channel IO architecture implements IO virtualization mission
 - Flexibility, agility, isolation, management at high performance
- Channel IO production HW available
 - Multiple generations, multiple vendors
- Channel IO API standard materializing
 - Native support by standard OSes
 - Supported on proprietary OSes
- Channel IO serves IO consolidation trend
 - e.g. FC over Ethernet
- Channel IO enables on-demand resource allocation
 - IO channels, IO resources, memory