Empowered by Innovation NEC

Virtualization for Kerrighed?

February 1st 2008
Kerrighed Summit, Paris

Erich Focht
NEC

Why virtualization?

“Virtualization” means many things!

- Multi-programming
  - any UNIX is virtualizing resources to allow their sharing
- Resource sharing
  - CPU, memory, disks, devices
- Machine partitioning
  - illusion of several machines running on same hardware
  - mainframes in '60s - '90s, ISPs and virtual desktops today
- Resource isolation
  - protect data from other users
  - protect from failures and faults
  - separate networks from each other

Why virtualization?

- Quality of Service
  - assigned memory, I/O bandwidth, fairness of scheduling
- Simulation and Emulation:
  - simulate different CPU and different hardware (devices)
  - testing, development, hardware, firmware
  - Example: old/new hardware, old/new software

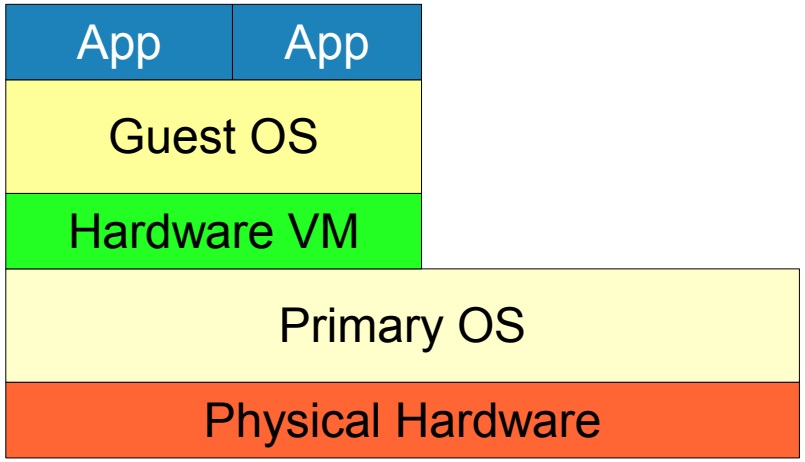
- A quote from 1974: [R. Goldberg, Survey of Virtual Machines Research]
  - "Virtual machine systems were originally developed to correct some of the shortcomings of the typical third generation architectures and multi-programming operating systems - e.g., OS/360."

Why virtualization?

- Resource joining
  - I/O virtualization:
    - many disks merged to one big virtual file space
    - user doesn't need to care of where his files are
    - admin can grow file space according to needs
  - Single System Image:
    - many machines joined to look like one single bigger machine
    - more resources
    - simpler management
  - Grid Computing:
    - join resources of many machines and allow to share them in easy way

Approaches

- Hardware Emulation
  - emulate/simulate different CPU than underlying hardware
    - accuracy level, latencies, cache behavior, ...
  - Examples:
    - SimOS, Simics,
    - Bochs, QEMU, MAME (Multi-Arcade Machine Emulator :-)
  - boot unmodified OS on virtual hardware!
  - slow :-(

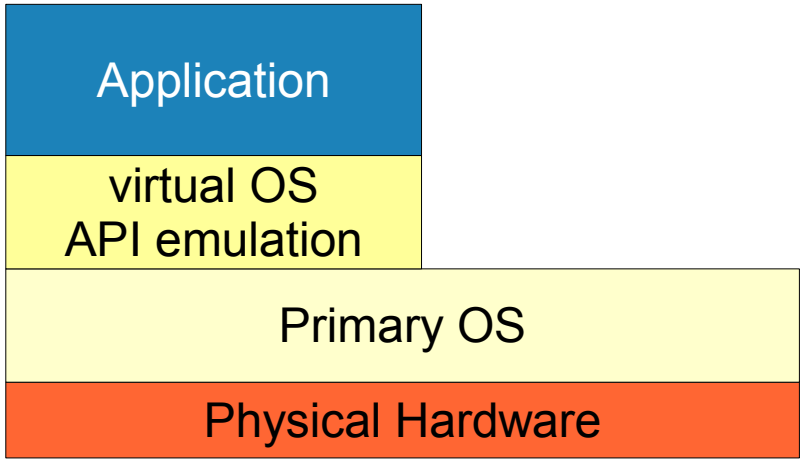


Approaches

- Instruction set virtualization (at CPU level (mostly))
  - binary translation of virtual CPU instructions to instructions of host (physical) CPU
    - Transmeta: X86 to VLIW on-the-fly instruction conversion
  - dynamic recompilation (QEMU?)
- Processor virtualization (at application level)
  - Programming language virtual machine
  - Pseudo-code, P-code, Byte-code
    - run on virtual CPU, with virtual instruction set
    - Pascal, BCPL, (concept used in compilers for intermediate language (RTL)), .NET, Parrot
    - Java, JVM

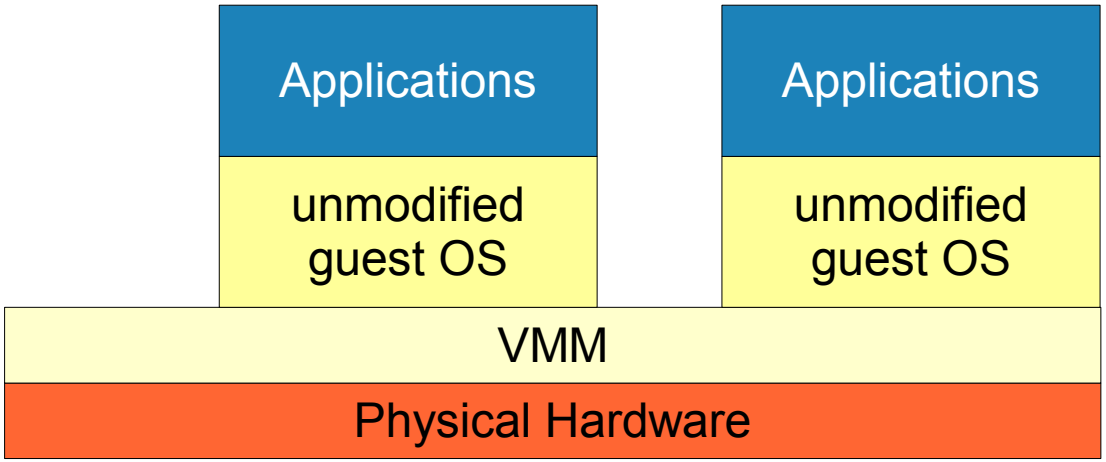
Approaches

- API/ABI Emulation
  - create execution environment that help run programs for other OS (of same hardware)
    - SUN WABI, Ixrun (SCO UNIX),
    - (MACH emulation library - Mikrokernels)
    - WINE



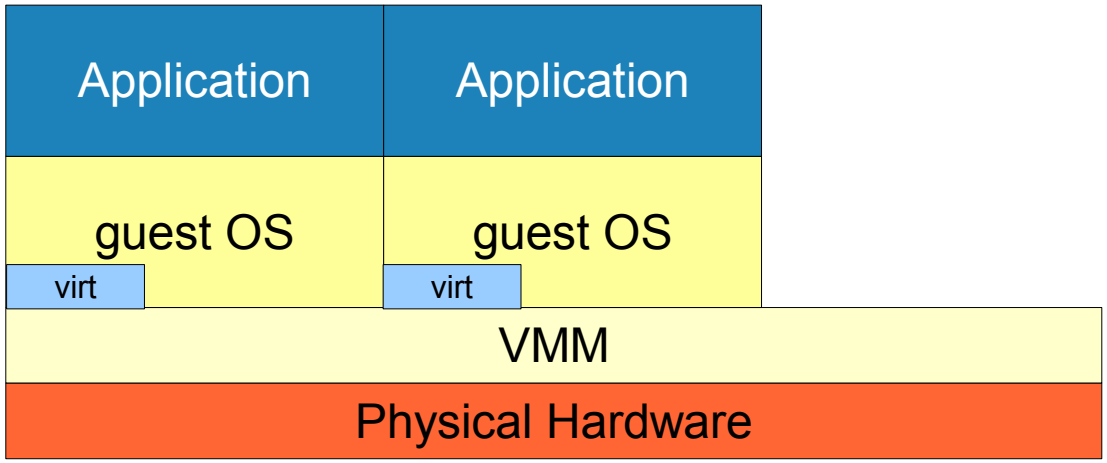
Approaches

- Full (native) virtualization
  - Hypervisor, VMM (virtual machine monitor)
    - mediates between virtual machines and hardware
    - unmodified guest OS
    - CPU needs native support for virtualization
      - remark is particularly important for x86
    - z/VM, Vmware, Xen, KVM, Virtualbox (+emulation)
    - performance: slower than native
      - catch faults
      - traps, tracing



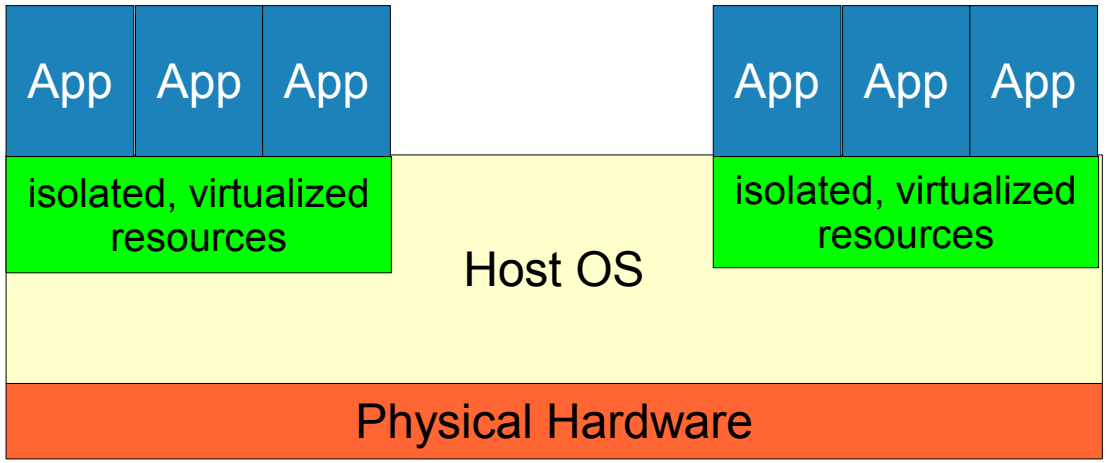
Approaches

- Paravirtualization
  - Hypervisor, VMM (virtual machine monitor)
    - mediates between virtual machines and hardware
    - guest OS is virtualization aware
    - performance: almost as fast as native, better than full virtualization
    - Xen, UML, lguest, VMware



Approaches

- OS-level virtualization
  - isolates independent “servers” from each other
  - run in one instance of operating system
    - Vserver, Viruozzo, OpenVZ, (chroot)
    - Solaris containers, FreeBSD jails, Linux containers

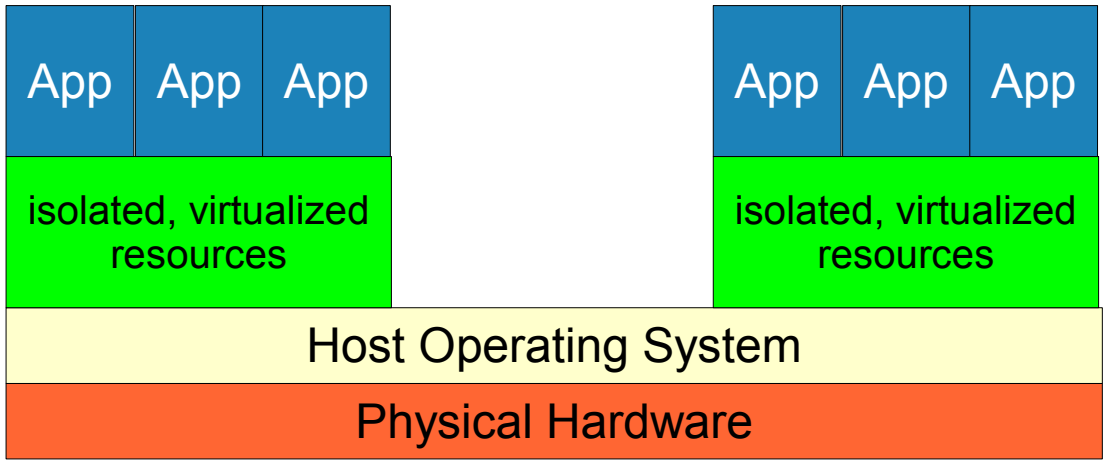


Approaches

- OS-level virtualization
  - isolates independ
  - run in one instanc
    - Vserver, Viruoz
    - Solaris containe

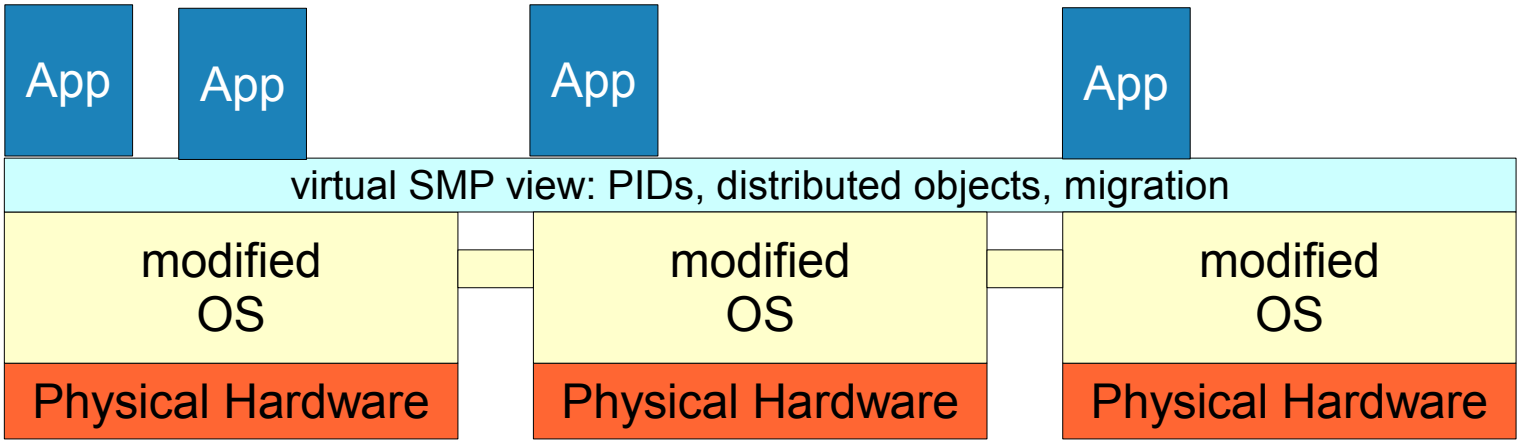
Linux containers

- resource hierarchy
- isolation
- QoS
- name spaces (PID, uname, mounts)
- migration, checkpoint
- but only across nodes with exactly same OS!



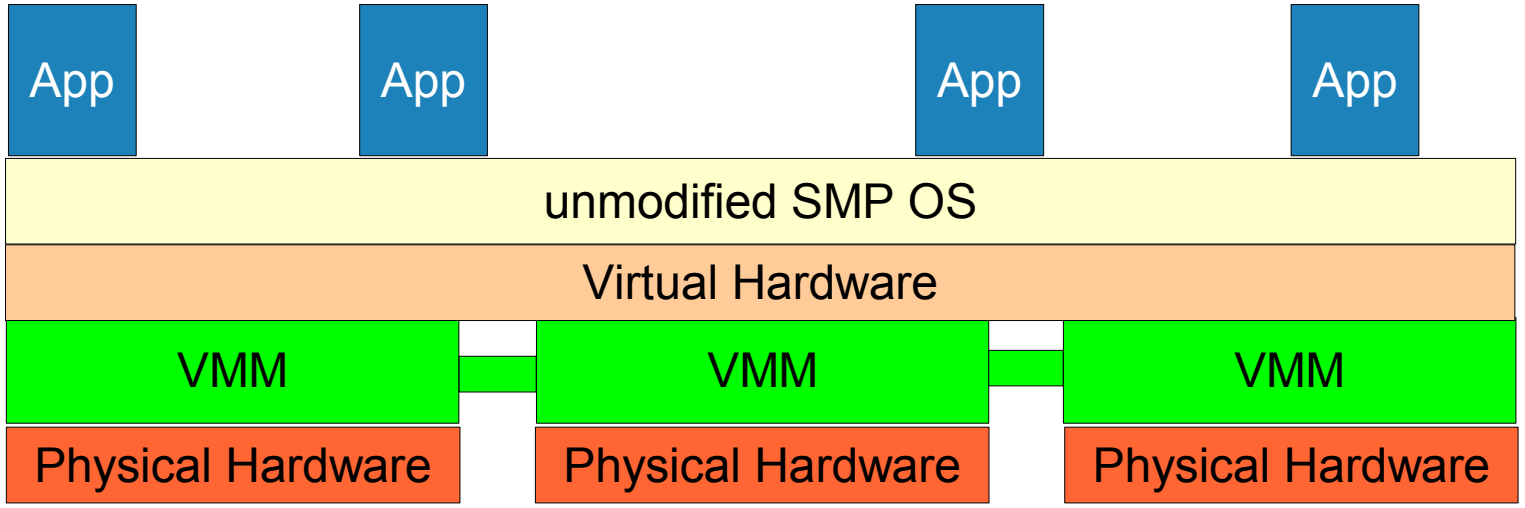
Approaches

- Single System Image
  - modified OS instances cooperate to provide distributed services
    - applications can use distributed services with a “virtual SMP” feeling



Approaches

- virtual SMP hardware
  - VMM layer as firmware
    - provides SMP hardware view
  - unmodified SMP OS
  - ScaleMP (commercial product)



Virtualization in Kerrighed?

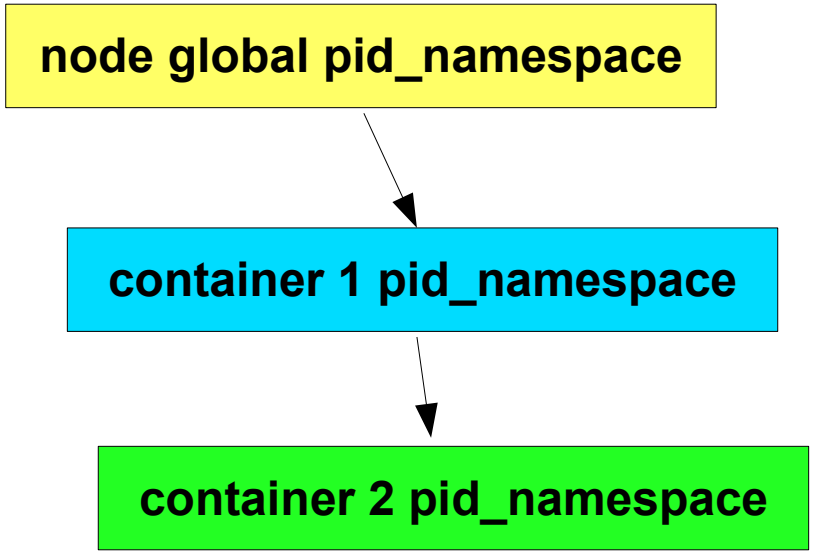
- Containers!
  - Why?
    - resource isolation & grouping, QoS
    - merged, so people will use it!
      - new scheduler
      - per container memory accounting
    - resource virtualization (PID, UID, network namespaces)
  - How?
    - container restricted to one node (easiest)
    - migrate entire containers
      - people want this, actually (on normal, non-SSI systems)
      - Application: virtual servers
    - in SSI: container resources distributed across nodes?
      - does this make sense?

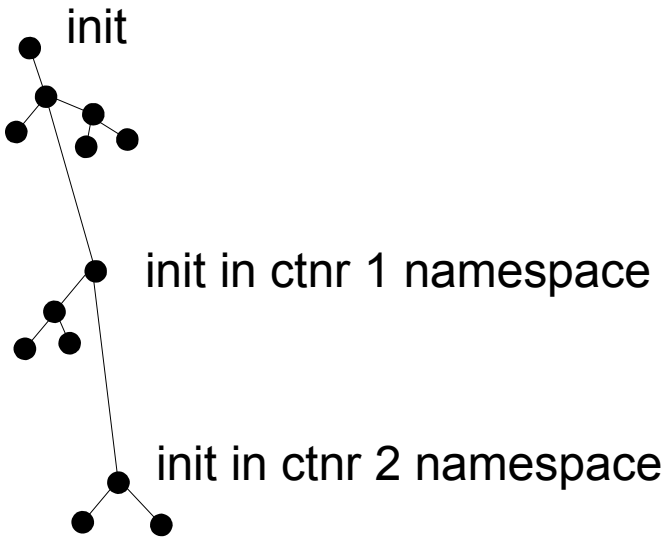
Virtualization in Kerrighed?

- Containers!
  - resource virtualization: PID
    - namespaces: global, per container, hierarchical
    - Kernel should **never** use PIDs
    - much code rewritten to eliminate numeric PIDs from Kernel
  - Kerrighed: global PIDs
    - some bits used for initial node identification
  - Can we adapt PID virtualization/namespaces to Kerrighed?
    - remove usage of numeric PIDs
    - use task_struct and PID structure instead
    - add additional namespace level for global Kerrighed view?

Virtualization in Kerrighed?

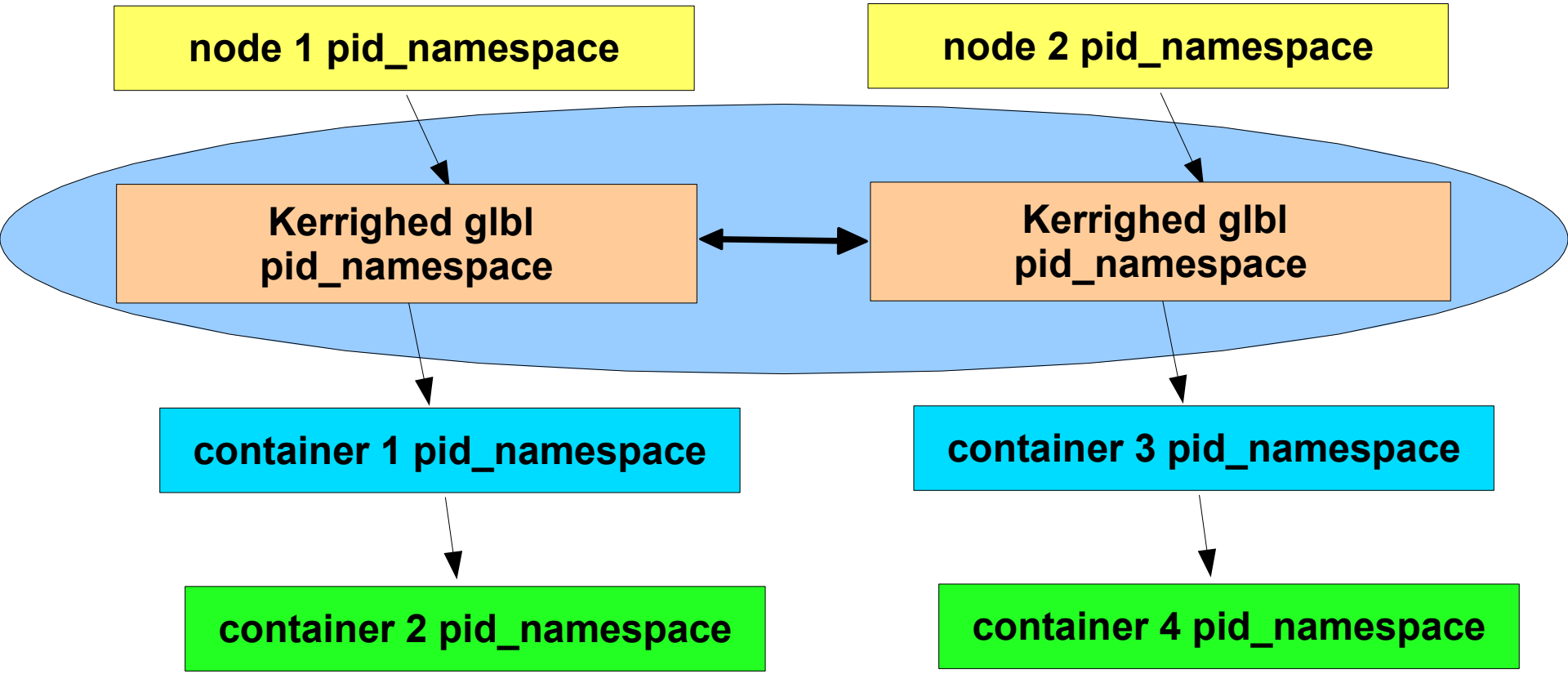
- PID namespaces





Virtualization in Kerrighed?

- PID namespaces



Virtualization...

- PID approach is usable for other namespaces, too
  - UID
    - do we care?
  - SYSV IPC (shm key IDs)
    - solves some problems with checkpoint/restart/migration
  - network
    - make containers migrateable
    - processes see same network setup
    - solves some problems that Kerrighed also has solved
  - filesystem view
    - mount points, no need for additional effort
  - utsname
    - good for us, no need for additional effort (same name per cluster?)

Other stuff...

- control groups
  - memory!
    - important for isolation and QoS
    - AFAIK, done by detailed accounting
    - per node, yet another argument for not distributing containers
    - ... per container swap & container checkpoint
  - CPU
    - uninteresting if containers are restricted to one node
  - Disk I/O, network I/O
    - no need for attention so far