# Linux Containers: virtualization without overhead or strange patches

#### Sam Vilain, Catalyst IT Talk for LCA2010 SysAdmin miniconf Wellington, New Zealand

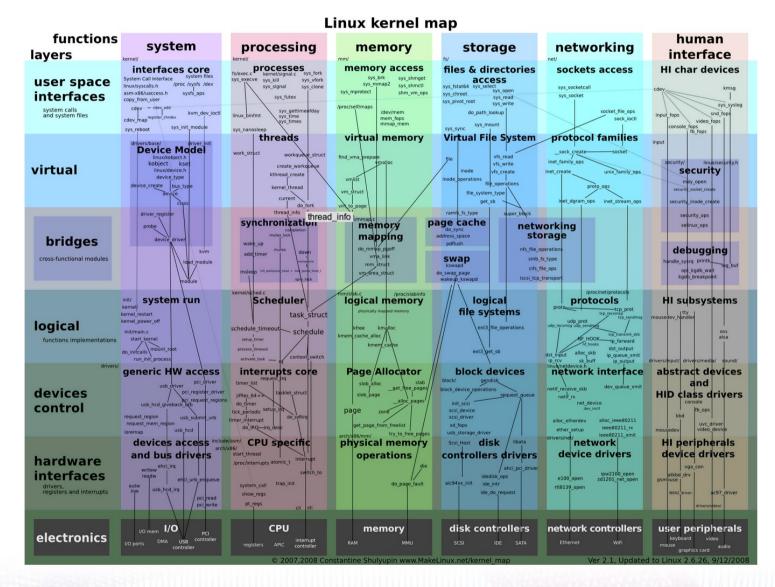
# Warning

- "miniconf" grade talk
- Always check facts/'git log'
- Refer resources at end for better facts

### Broad Approaches to Virtualization

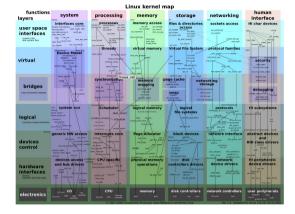
- Complete emulation eg VMWare, QEMU
- Hypervisor eg Xen, KVM, Hurd
- System call level eg VServer or OpenVZ, Containers, etc
- Application eg Vhosting
- Scale of continuum functionality vs performance

### This is your Linux



State Parts of

### This is your Linuxes on QEMU



Linux kernel map								
functions layers	system	processing	memory	storage	networking	human interface		
user space interfaces	interfaces core South Californian State South Californian State South Californian State South Californian South California	And a series of the series of	memory access review review revieway review. NV/NUMPAT reviewe NV/NUMPAT reviewe NV/NUMPAT reviewe NV/NUMPAT reviewe NV/NUMPAT reviewe	files & directories access rectant internet rectant internet department rectant department rectant rectant department rectant rectant rectant rectant department recta	sockets access	HI char devices		
virtual	Device Model Linguistics of the second seco	mit, mit unit, mit u	virtual memory	Virtual File System	protocol families	security security security and anti-anti- more real-anti-		
bridges cress-functional modules	-57	synchronization	memory P mapping	swap int	1, 1204 Nr. ANN	debugging tests, set to below		
logical functions implementations	nt system tun and sint i manuer, at and sint	Scheduler Issis Structure Indiana Indian Indiana Indiana Indiana Indiana Indiana Indiana Indiana India	Ingical memory	logical file systems economic account major.s	Protocols reading and the second sec	HI subsystems		
devices control	Seneric HW access and a seneric seneric and a seneric seneric seneric and a seneric seneric and a seneric seneric seneric seneric and a seneric and a seneric seneric and a seneric seneric and a sene	Interrupts core	Page Allocator	block devices	network interface web ys and doug be done web be done web done web	abstract devices and HID class drivers		
hardware interfaces	devices access and and bus drivers and out of the second and and out of the second and and out of the second and and and and and and and and and and	And CPU Specific Statistics Description (1) waves and the specific statistics (1) and (1) and	physical memory operations	controllers drivers dec.gr., dec.gr.,	network device drivers etm. see etm. see etm. see etm. see	HI peripherals device drivers var,ar perior and run anti-drav		
electronics	u) man. 1/0 Ki ul pan, 104 UB tanàna uninaka	CPU reptiler dit: relative	memory set set	disk controllers	network controllers	user peripherals		

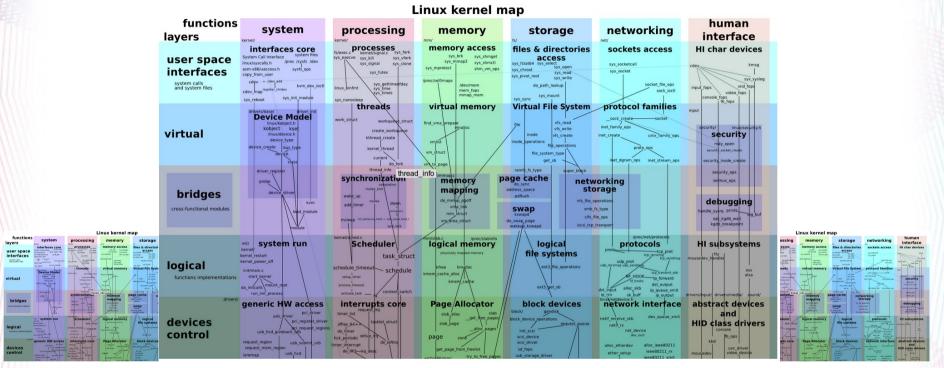
Linux kornol man

QEMU emulation

QEMU emulation

Linux kernel map							
functions layers	system	processing	memory	storage	networking	human interface	
user space interfaces system cals and system files	Interfaces core System Call Interface System (File Interface) System (File Corp. Form, user Cder, Trap replet, create Cder, Trap replet, create Cder, Trap replet, create Syst, reloct Syst, Int, Troob, e	hitesc. keneliziyal. sys.fork sys.gencer sys.fill sys.fork sys.jipeal sys.fore sys.fute sys.fute inar.genent sys.futes sys.futes	sys_brk sys_mmap2  sys_brk sys_mmap2    sys_mspace  sys_brack    sys_member  sys_member    genecisettimops	files & directories access ys_futuret ys_sect ys_futuret ys_sect ys_futuret ys_sect ys_stree ds_puth_lookup ys_syste ys_stree	sockets access sys_societ sys_societ societ_file_ops societ_file_ops	HI char devices	
virtual	erversitezer Device Model Histopert Iver Inutiverie fyse device_crease Bruce Bru	threads work_struct create_workputve kitread_create kense_broad current op. took	viritual memory	Sile Virtual File System	protocol families sec. grath sociat inet funity, ses inet create proto oss inet dyram, ses inet dyram, ses inet stream, ops	Novi security niky.com security niky.com security_inste_com	
bridges cross-functional modules	onver-reporte prete sevice driver kom racide	Synchronized thread	info immedit memory p mapping do rento speri oriti sak em janut vy fores statet	address, space poliush rhs_file_e	fs_type the_ops	seturty, ass seitrux, ass <b>debugging</b> handle, syster, prister, mag, but asj, Lasb, wait ugab, treakpoint	
Iogical functions implementations	do_initcals mount_soat	tantistice. Scheduler task struct schedule timeout schedule time time grace imment content patich	Minital Sectors and Sectors an	et3 the genations et1 the genations et13 ptr, tb	Protocolsandon protocolsandon setto post standard dial setto post standard dial standard	HI subsystems	
devices control	generic HW access ub draw bol repairs draw ub hot probable draw ub hot probable draw repairs repairs repairs repairs ub hot wereased ub hot wereased and hot repairs repairs	interrupts core	Page Allocator stot also:	block devices block generations met sol sci driver sci driver sci driver sci driver sci driver sci driver	Incluserences network interface net receive ska net receive net seco net seco ne	abstract devices and HID class drivers conste bio bio with driver rescator	
hardware interfaces drivers, registers and interrupts	devices access indus and bus drivers writee <sup>efci (10</sup> meter <sup>efci (10</sup> odw effici (10) odw effici (10)	Altern CPU Specific Set (stread) (strectionenges stand), internet settch to system, call tran, int some rogs stread tran, int some rogs stread transition of an	physical memory operations de de de de de de	Scal, Heat disk / leasts controllers drivers 	network device drivers exite gen puttor production rtilitigen	HI peripherals device drivers ysp.con pottore soc.com encoder encoder encoder	
electronics	UO mem I/O PCI I/O ports DMA USB controller controller	CPU registers APC controller	memory RAM HINU	disk controllers scsi ide sata	network controllers	user peripherals keytoord video graphics card audio	

### This is your Linuxes on Xen/KVM



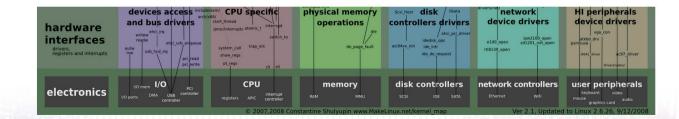
Virtualization

Hypervisor

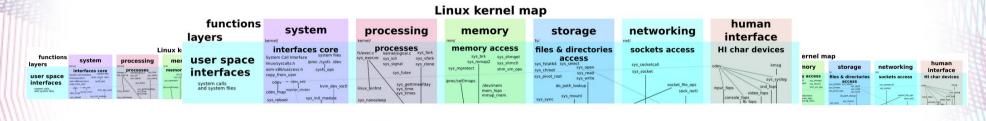
Hypervisor IRQ router Hypervisor Memory Manager

sor Virtual Block r Devices Virtual Network Devices, Network bridge

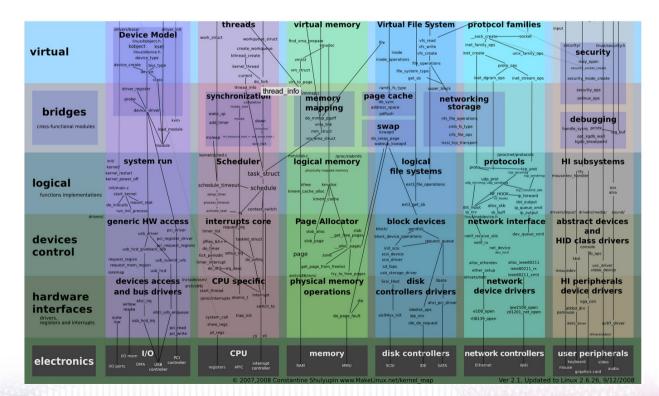
Hypervisor HID support



### **This is your Linux on Containers**



Virtualization	Container, namespace, controller	Process namespace, CPU controller	Memory Controller	Filesystem namespaces, IO controller	Virtual Interfaces or IP restrictions	Clustercuss
----------------	--	--	----------------------	--	---	-------------



### What is a container?

- What 'lxc' utilities deal with
- An abstract concept only not a concrete kernel object
  - Perhaps a single isolated daemon with minimal privileges
  - Perhaps a self-contained Linux system
- A set of namespaces logically grouped together
- Potentially, a set of *controllers* scheduling resources

### What is a namespace?

- Every task\_struct (process/thread) knows their namespace objects; cloned via clone(2)
- System calls go through the task\_struct → can provide "customised" results
- Eg, PID namespaces: processes with a particular namespace see private PIDs.
- Eric Biedermann's brainchild a radical departure from the extra syscall approach of VServer et al.

### **Restricting a process**

- chroot() changes /proc/self/root
- Capabilities de-fangs root
- Filesystem Namespaces changes /proc/self/mounts
- UTS Namespaces private hostname
- PID Namespaces private PIDs
- User namespaces private userIDs
- IPC Namespaces private messages
- Network Namespaces private interfaces
- /proc generally the way to inspect situation

### What is a controller?

 Influences scheduling decisions, a la Linux's TC for network scheduling

- (aside) "token bucket filter" CPU scheduler

- IBM engineers mostly AIUI
- Two parts:
  - Afferent: categorisation of processes into scheduling classes (control groups)
  - Efferent: actual implementation of scheduling (controller)

### What controllers exist?

- Network: groups classifier (CONFIG\_NET\_CLS\_CGROUP), then use TC
- CPU: CONFIG\_CGROUP\_SCHED etc
- Memory: RSS, Swap
- IO: CFQ group scheduling

### **Comparisons with VServer**

- Design differences: VServer restricts visibility of objects; namespaces make numbers distinct
- Enter mechanism: added later with namespaces; need to use init+getty or SSH.
- Network: network namespaces can give private network interfaces, directly bound or bridged. Private iptables.

### More VServer comparisons

- Devices: mknod whitelist allows containers to make /dev/null if they want
- User IDs: user namespaces instead of XID tagging I guess

### **Benefits of Lightweight Virtualization**

- Flexibility of management
- Filesystems, processes visible from host without stopping guest
- 100% speed
- 100% lightweight
- Freezing, unfreezing live migration, even between kernel versions

### Xen/KVM or Containers?

- Use Xen/KVM if you need:
  - hard resource partitioning → lower overall performance
  - differing kernel versions
- Use containers if you need:
  - soft resource partitioning  $\rightarrow$  maximum performance, fewer guarantees
  - process jails
  - live kernel upgrades
- Sometimes a mix is useful

#### Resources

- LXC HOWTO (vaguely useful) http://lxc.teegra.net/
- IBM page on containers http://www.ibm.com/developerworks/linux/library/l-lxc-containers
- Ixc Ubuntu package apt-get install lxc