# Linux Containers: virtualization without overhead or strange patches

Sam Vilain, Catalyst IT

Talk for LCA2010 SysAdmin miniconf

Wellington, New Zealand

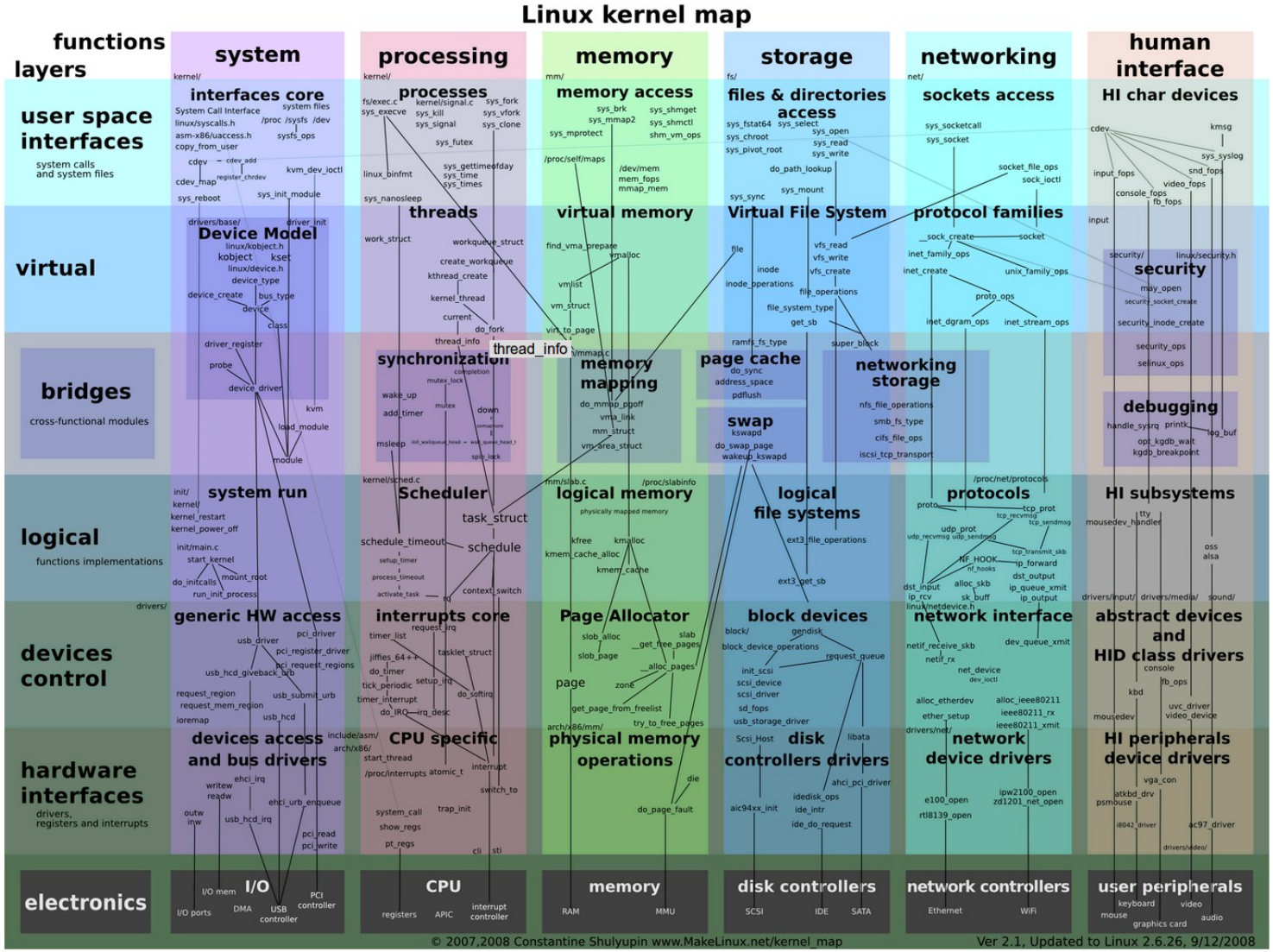
# Warning

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

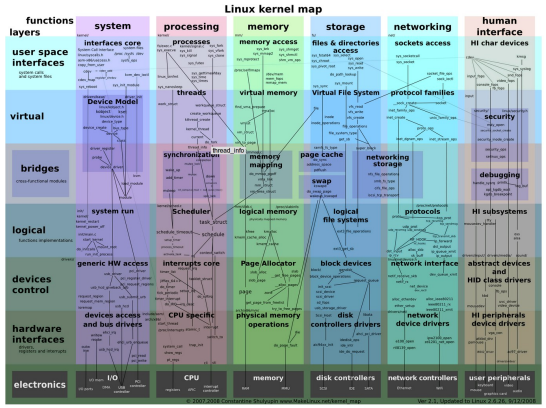
# Broad Approaches to Virtualization

- 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

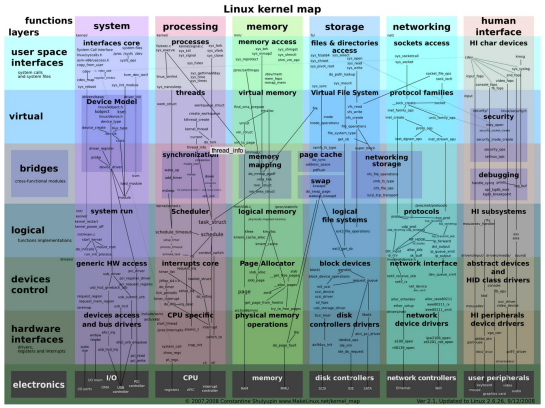
# This is your Linux



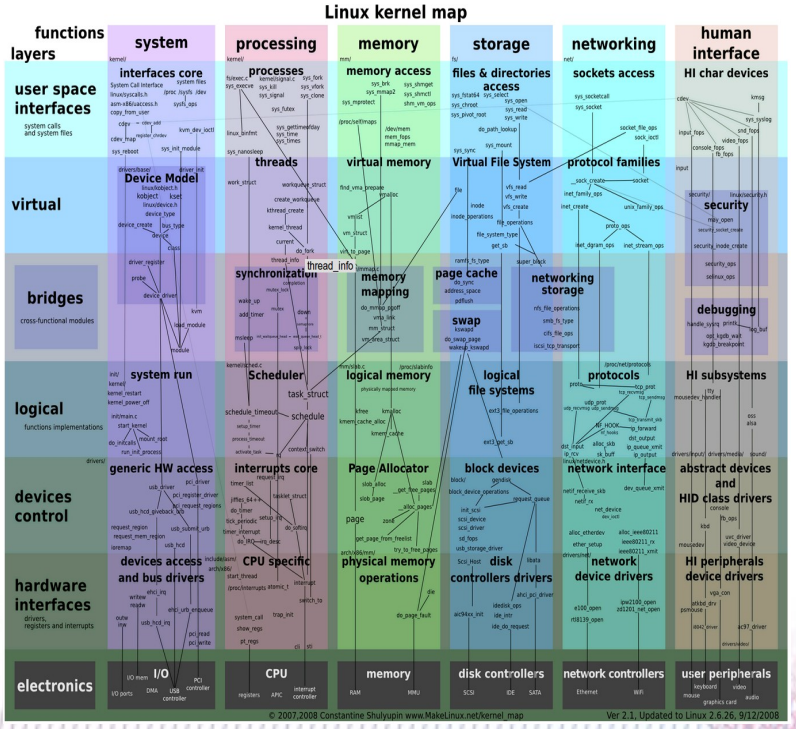
# This is your Linuxes on QEMU



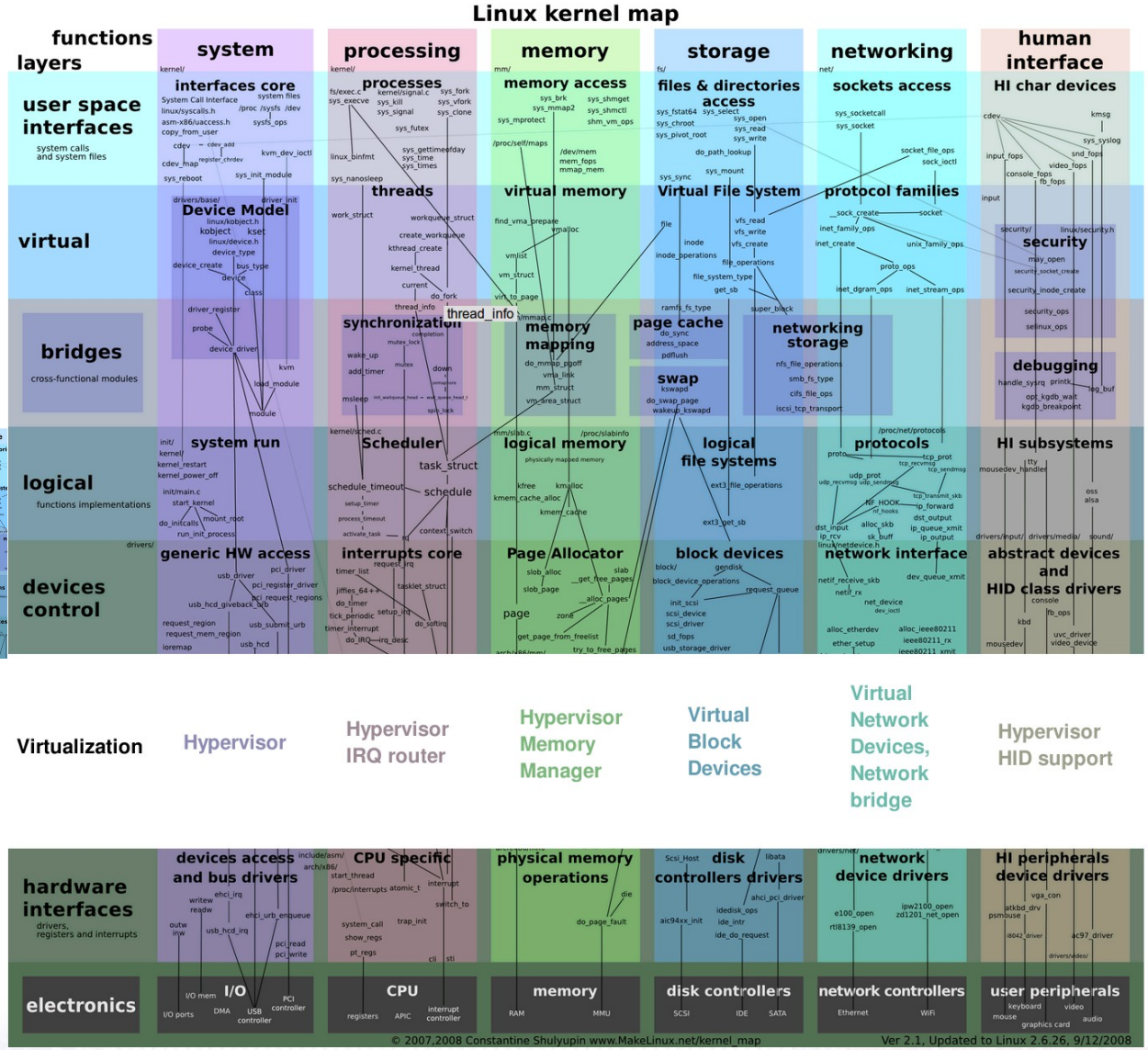

QEMU emulation



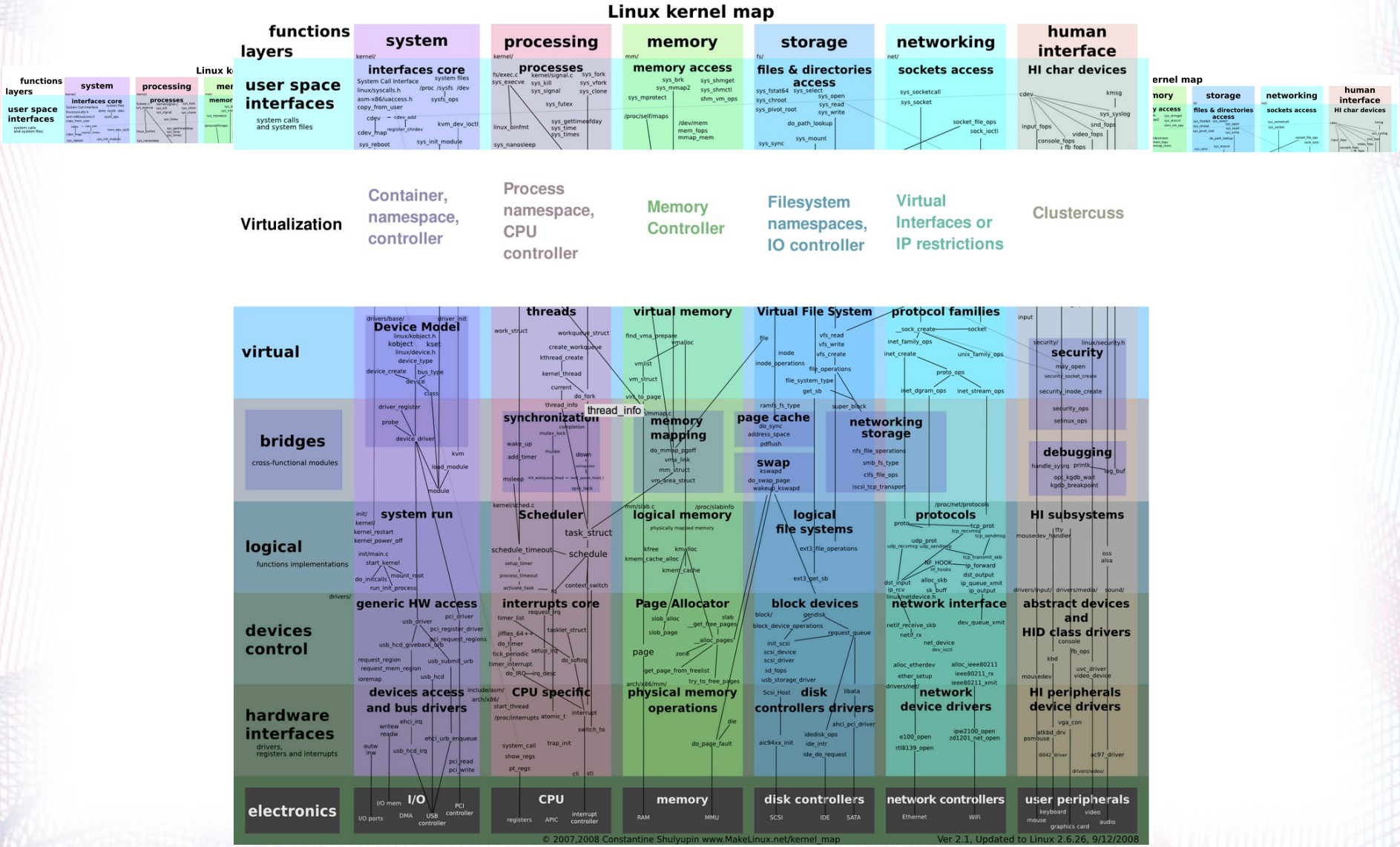

QEMU emulation



# This is your Linuxes on Xen/KVM



# This is your Linux on Containers



# 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 → 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
- lxc Ubuntu package
  `apt-get install lxc`