The failure of Operating Systems,

and how we can fix it.

Profit from the Cloud

Glauber Costa

Lead Software Engineer

August 30th, 2012 – Linuxcon

Opening Notes

- I'll be doing Hypervisors vs Containers here. But:

Opening Notes

- I absolutely don't believe Hypervisors are useless.
- I will also not say Containers are better than Hypervisors (or vice-versa). That is dependent on your use-case, and the comparison between them is not the point of this talk.
- I am mostly talking about “traditional” Linux. Some of what I am presenting is already upstream, so you may get a sense of “Oh, but Linux already does that”.

A Brief History of Operating Systems

Introduction

- History books tell us that back in the day, a computer ran a single program.
- The white bearded guys in the audience may be able to confirm that.
- This is way too inefficient.

Smarter resource usage

- Whatever equivalent of Ingo Molnar existed at the time, came up with a scheduler.
- I/O is no longer wasted time.

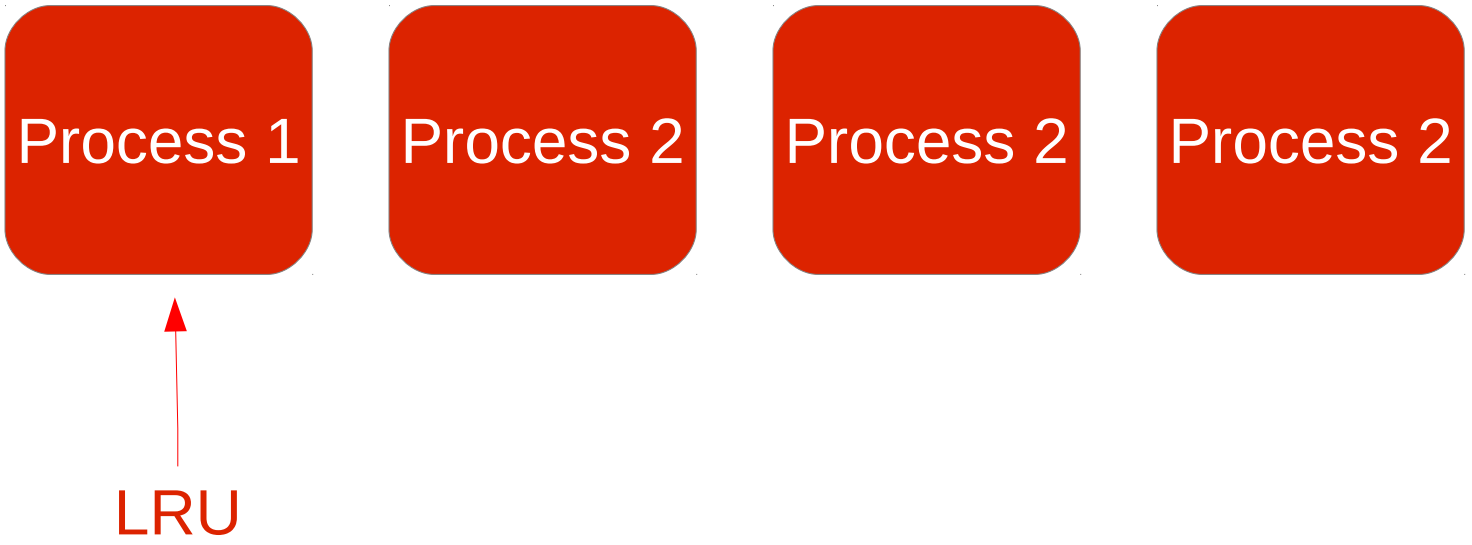
Smarter resource usage

- Programs want to start at a fixed address,
- and want to point to its code and data at fixed locations.
- Meet the VM.

It has a number of limitations, though.

- Fire firefox (or chrome chromium)
  - Start something else and use lots of memory.
  - Watch the system page out.
  - What do you think happens to your browser?
- Give an unprivileged user wanting CPU power a hand,
  - He'll surely want an arm (even if running on x86).
  - nice is not very nice for guaranteeing cpu time.
  - Basic abstraction is a process, but services have plenty of them.

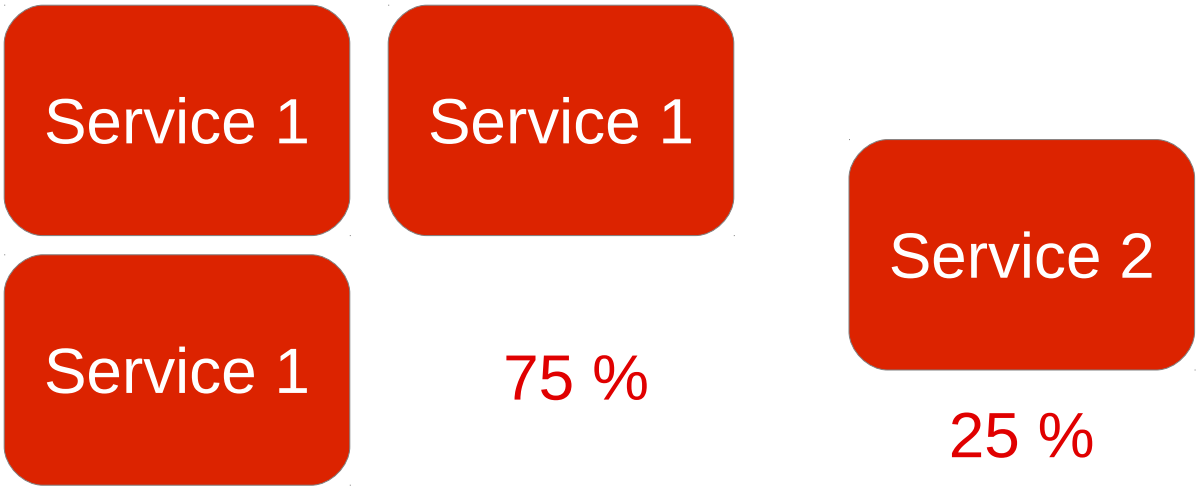
Memory eviction order



- P1 will be penalized by P2's high memory usage.

No process schedule grouping

No process schedule grouping



- Dynamic adjusting those priorities is very hard.
- May not be fine grained enough.

Example exploit

```
$ while true; do mkdir x; cd x; done
```

- Each 'x' will generate directory entry (dentry)
- Those dentries are pinned in memory.
- This is non-reclaimable, kernel memory!
- Will consume all memory before any disk quota can kick in.
- ***Unrelated*** processes will fail to be serviced.

Classical resource abuse

```
$ :() { :;:& };:
```

- Fork bomb.
- Should be ***your*** problem, not mine.

The hypervisor solution and what it allows

Enter KVM (and others)

- Started by kivity.avi,
- Designed to make the Linux Kernel itself the HV – in KVM case.
- Each VM is a (group of) process(es).
  - Provides a N:1 CPU mapping; logical grouping.
  - It also provides an fair view on memory – If you don't overcommit VMs, you will never evict them.

Use cases

- I want to run web and mail servers, and databases.
- I don't want them to interfere with each other.
  - Too much memory usage by one hurts another,
  - forks can increase relative aggregate CPU throughput.
- I want to gather accurate service statistics.
  - Preferably with standard tools like “top”.

Use cases

- I want each of them to have its own IP address.
  - And the same standard ports – 22, 25, 80, etc.
- This may also mean running an isolated userspace,
  - With root in all of them,
  - Maybe with their own init process.
  - Compatible and certified stack (for old software).

Use cases

- I want to run different versions of Linux
  - This can also be part of a certified stack.
- I want to run a heterogeneous datacenter
  - Other OSes as well.

Containers

Use cases

Usage	Should the OS do it?
Make sure processes high resource usage doesn't interfere with others.	Yes
Logically map process to a single process.	Yes
Provide logically grouped introspection.	Yes
Provide processes with flexible view of resources, such as ports.	Yes
Run different kernels.	No

New additions to the Linux Kernel

- Network namespaces.
  - Provides a unique IP per group of processes.
  - Provides raw device view per-process as well.
  - Easy packet filtering (per-device).
  - 30 processes connecting to port 80? No problem.

New additions to the Linux Kernel

- Mount namespaces.
  - Linux can chroot, but new mounts are globally visible.
  - I may want a private mount.
- User namespaces.
  - Allows more than one “root” user in the system.
  - Of course, other users as well.

New additions to the Linux Kernel

- cgroup : logical grouping of processes.
  - “WebServer1”, “MailServer3”, etc.
- Can attach controllers.
  - I will briefly introduce two of them.

New additions to the Linux Kernel

- The cpu controller.
  - The scheduler will first schedule among groups,
  - then it will schedule inside each group.
  - Inside each group, we can have another group.
  - Can limit the maximum CPU time used.

New additions to the Linux Kernel

- The memory controller.
  - Can limit the maximum amount of memory.
  - Soft and Hard Limits.
  - Hard Limits will page even if there is memory available in the system.
  - Controlling resources used by the kernel is work in progress. Essential to prevent some exploits.

Other features

- Live Migration
  - Came to be a killer feature for hypervisors.
  - Can also be done by containers by checkpoint/restore
- Specialized loop device.
  - Separate journal per-container, without heavy fs changes.
  - inode number stability.

Containers, today.

- It is possible to run production containers today.
  - Upstream Linux lacks the whole infrastructure.
  - The OpenVZ fork provides a stable, secure, and mature Open Source container offer.
- You might have heard about LXC.
  - Userspace tool, only run what Linux provides – future only.

Work Status

- A lot of it is already in, and works all right
  - cgroups basic infrastructure.
  - CPU controller.
  - Network Namespaces.
- Some of it in, needs improvements and the test of time
  - User namespaces, fully unprivileged still not possible.
  - Mount namespaces, joining still not possible.
  - PID namespaces, joining still not possible.
  - Block I/O controller.

Work Status

- To be merged
  - Slab object accounting and fork bomb prevention.
  - Group-aware kernel memory shrinking.
  - Group-aware filesystem quotas.
  - Specialized loop device (separate journal + inode# stability)
  - Live Migration (Checkpoint/Restore)

Tooling

- LXC.
- OpenVZ tools are being patched as we speak to run with functionality already present in the Upstream Kernel.
- Love it or hate it, SystemD uses cgroups as one of its building blocks.
- The “unshare” command can run an arbitrary process in a separate namespace.


Parallels®