A microkernel-based orchestrator for distributed Internet services?

Alex Auvolat <alex@adnab.me>
Deuxfleurs https://deuxfleurs.fr
FOSDEM 2024
Distributed system self-hosting

Low-cost, low-power, second-hand hardware running the Deuxfleurs infrastructure

Servers in several geographical locations for redundancy
Our current stack

OS

Platform

Storage services

Applications

Distributed operating system?
Deep dive into this horrible mess

Monolithic kernel

- Device drivers (can crash your system)
- Filesystems (too complex)
- Networking (huge mess of shared state)
- Bolted-on isolation mechanisms (cgroups, ...)

All concurrently trying to mutate some shared state

IP networking (??)

Containers (isolation that doesn’t really work)

Stuff is slow, easily broken, and hard to work on

All running with highest privilege level on your CPU
Distributed systems are boxes connected by arrows
Microkernels would work

boxes = processes (incl. device drivers)

arrows = IPC (of various kinds)

what microkernels do:
- manage processes
  -> address space isolation = memory management
  -> cpu time sharing = scheduling
  -> controlled hardware access
  = multiplexing of fundamental CPU resources
- various IPC mechanisms
  -> message passing
  -> shared memory + semaphores

looks like a match!
What this could look like

Microkernel

orchestrator

ethernet driver

network multiplexing (vswitch)

disk driver

disk multiplexing (partitions)

os component, e.g. VPN (native process)

application 1 (unikernel)

application 2 (tiny linux vm)

etc.
What we need

- a way to describe boxes and their connections dynamically
  = an orchestrator

- some standard box and arrow types

<table>
<thead>
<tr>
<th>box types:</th>
<th>arrow types:</th>
</tr>
</thead>
<tbody>
<tr>
<td>hardware drivers</td>
<td>management APIs</td>
</tr>
<tr>
<td>resource multiplexers</td>
<td>observability (e.g. logging)</td>
</tr>
<tr>
<td>orchestration &amp; mgmt components</td>
<td>block storage</td>
</tr>
<tr>
<td>native components</td>
<td>networking</td>
</tr>
<tr>
<td>VM apps (unikernels, tiny linux vms)</td>
<td></td>
</tr>
</tbody>
</table>

- tooling and ecosystem
Goals and non-goals

Main focuses:
- declarative configuration
- remote management and observability
- clustering support & dynamic reconfiguration
- I/O performance (async I/O all the way)
- simplicity & minimalism

Non-goals:
- POSIX API compatibility
- desktop operating system
### Leveraging the existing

<table>
<thead>
<tr>
<th>microkernels</th>
<th>I/O models</th>
<th>frameworks &amp; OSes</th>
</tr>
</thead>
<tbody>
<tr>
<td>L4 / seL4</td>
<td>VirtIO</td>
<td>Genode</td>
</tr>
<tr>
<td>NOVA</td>
<td>io_uring</td>
<td>Redox OS</td>
</tr>
<tr>
<td>Fuschia</td>
<td>9P</td>
<td>GNU Hurd</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Where to start from?