Introduction of the Genode OS Framework

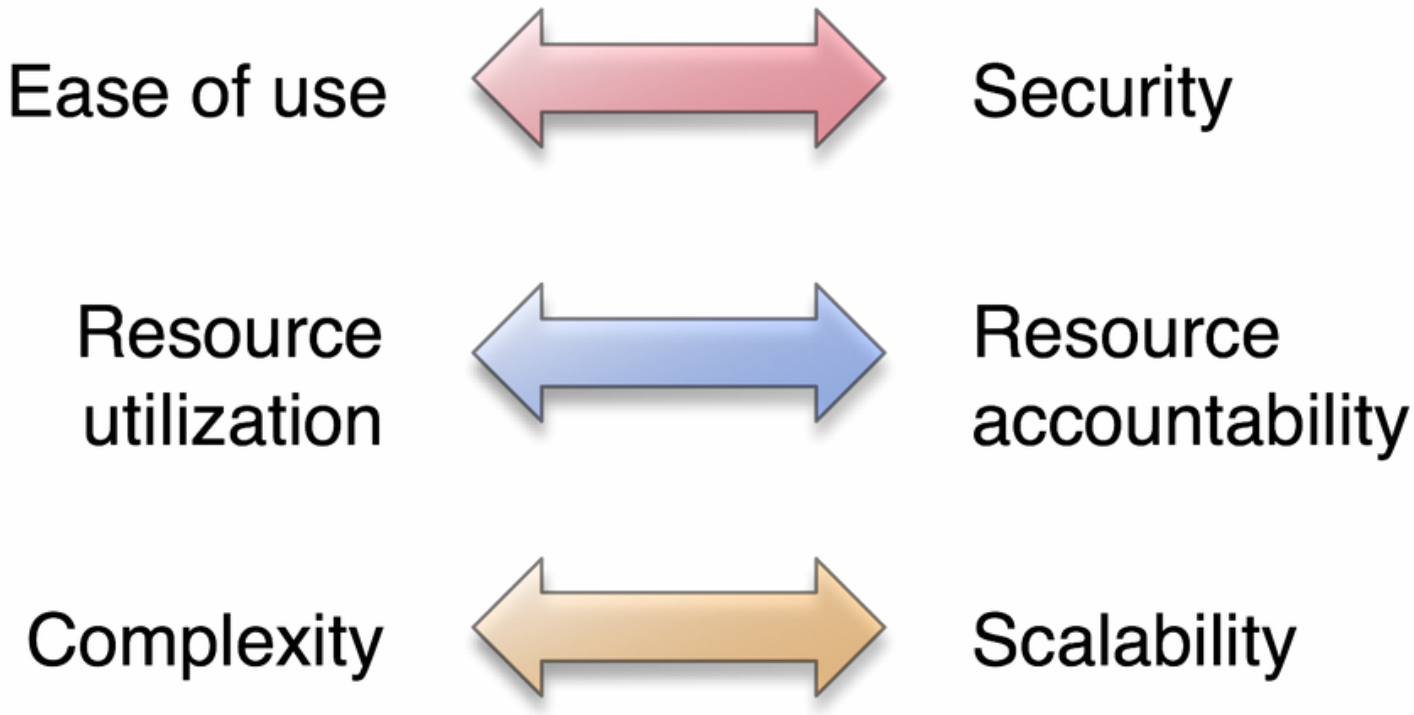
Dr.-Ing. Norman Feske
<norman.feske@genode-labs.com>

Outline

1. Background
2. Genode entering the picture
3. Architectural principles
4. OS Framework
5. Showcases
6. Current ventures

Outline

Universal truths



Problem: Complexity

Today's commodity OSes — Exceedingly complex trusted computing base (TCB)

Problem: Complexity

Today's commodity OSes — Exceedingly complex trusted computing base (TCB)

TCB of an application on Linux:

- Kernel + loaded kernel modules
- Daemons
- X Server + window manager
- Desktop environment
- All running processes of the user

Problem: Complexity

Today's commodity OSes — Exceedingly complex trusted computing base (TCB)

TCB of an application on Linux:

- Kernel + loaded kernel modules
- Daemons
- X Server + window manager
- Desktop environment
- All running processes of the user

→ **User credentials are exposed to millions of lines of code**

Problem: Complexity (II)

Implications:

- High likelihood for bugs (need for frequent security updates)
- Huge attack surface for directed attacks
- Zero-day exploits

Problem: Global names

- Many examples on traditional systems
  - UIDs, PIDs
  - network interface names
  - port numbers
  - device nodes
  - ...

Problem: Global names

- Many examples on traditional systems
  - UIDs, PIDs
  - network interface names
  - port numbers
  - device nodes
  - ...
- Leak information

Problem: Global names

- Many examples on traditional systems
  - UIDs, PIDs
  - network interface names
  - port numbers
  - device nodes
  - ...
- Leak information
- Name is a potential attack vector (ambient authority)

Problem: Resource management

- Pretence of unlimited resources
- Lack of accounting
  $\rightarrow$ Largely indeterministic behavior
  $\rightarrow$ Need for complex heuristics, schedulers

Problem: Resource management

- Pretence of unlimited resources
- Lack of accounting

  → Largely indeterministic behavior

  → Need for complex heuristics, schedulers

```
Jul 24 12:58:30 neo kernel: [72454.482259] cupsd invoked oom-killer: gfp_mask=0x201da, order=0, oom_adj=0, oom_score_adj=0
Jul 24 12:58:30 neo kernel: [72454.482264] cupsd cpuset=/ mems_allowed=0
Jul 24 12:58:30 neo kernel: [72454.482268] Pid: 1416, comm: cupsd Tainted: G        WC  3.0.0-22-generic #36-Ubuntu
Jul 24 12:58:30 neo kernel: [72454.482270] Call Trace:
Jul 24 12:58:30 neo kernel: [72454.482279]  [<ffffffff810b5c8d>] ? cpuset_print_task_mems_allowed+0x9d/0xb0
Jul 24 12:58:30 neo kernel: [72454.482286]  [<ffffffff8110df91>] dump_header+0x91/0xe0
Jul 24 12:58:30 neo kernel: [72454.482289]  [<ffffffff8110e2f5>] oom_kill_process+0x85/0xb0
Jul 24 12:58:30 neo kernel: [72454.482293]  [<ffffffff8110e69a>] out_of_memory+0xfa/0x250
Jul 24 12:58:30 neo kernel: [72454.482298]  [<ffffffff81113f3f>] __alloc_pages_nodemask+0x80f/0x820
Jul 24 12:58:30 neo kernel: [72454.482304]  [<ffffffff8120a1a0>] ? noalloc_get_block_write+0x30/0x30
Jul 24 12:58:30 neo kernel: [72454.482311]  [<ffffffff8114a0a3>] alloc_pages_current+0xa3/0x110
Jul 24 12:58:30 neo kernel: [72454.482314]  [<ffffffff8110ab4f>] __page_cache_alloc+0x8f/0xa0
Jul 24 12:58:30 neo kernel: [72454.482318]  [<ffffffff8110afae>] ? find_get_page+0x1e/0x90
Jul 24 12:58:30 neo kernel: [72454.482321]  [<ffffffff8110cea4>] filemap_fault+0x234/0x3e0
Jul 24 12:58:30 neo kernel: [72454.482326]  [<ffffffff8115f07b>] ? mem_cgroup_update_page_stat+0x2b/0x110
Jul 24 12:58:30 neo kernel: [72454.482330]  [<ffffffff8112caf4>] __do_fault+0x54/0x510
Jul 24 12:58:30 neo kernel: [72454.482334]  [<ffffffff8113021a>] handle_pte_fault+0xfa/0x210
Jul 24 12:58:30 neo kernel: [72454.482337]  [<ffffffff811306e8>] handle_mm_fault+0x1f8/0x350
Jul 24 12:58:30 neo kernel: [72454.482344]  [<ffffffff815f8913>] do_page_fault+0x153/0x530
Jul 24 12:58:30 neo kernel: [72454.482350]  [<ffffffff81011069>] ? read_tsc+0x9/0x20
Jul 24 12:58:30 neo kernel: [72454.482355]  [<ffffffff8108cd2d>] ? ktime_get_ts+0xad/0xe0
Jul 24 12:58:30 neo kernel: [72454.482361]  [<ffffffff8117bb6a>] ? poll_select_set_timeout+0x7a/0x90
Jul 24 12:58:30 neo kernel: [72454.482365]  [<ffffffff815f5615>] page_fault+0x25/0x30
Jul 24 12:58:30 neo kernel: [72454.493363] Out of memory: Kill process 22727 (oom) score 691 or sacrifice child
Jul 24 12:58:30 neo kernel: [72454.493367] Killed process 22727 (oom) total-vm:2702616kB, anon-rss:2701332kB, file-rss:172kB
```

Key technologies

- Microkernels
- Decomponentization, kernelization
- Capability-based security
- Virtualization

Tricky questions

How to...

- ...build a system without global names?
- ...trade between parties that do not know each other?
- ...reclaim kidnapped goods from an alien? (without violence)
- ...deal with distributed access-control policies?
- ...transparently monitor communication?
- ...recycle a subsystem without knowing its internal structure?

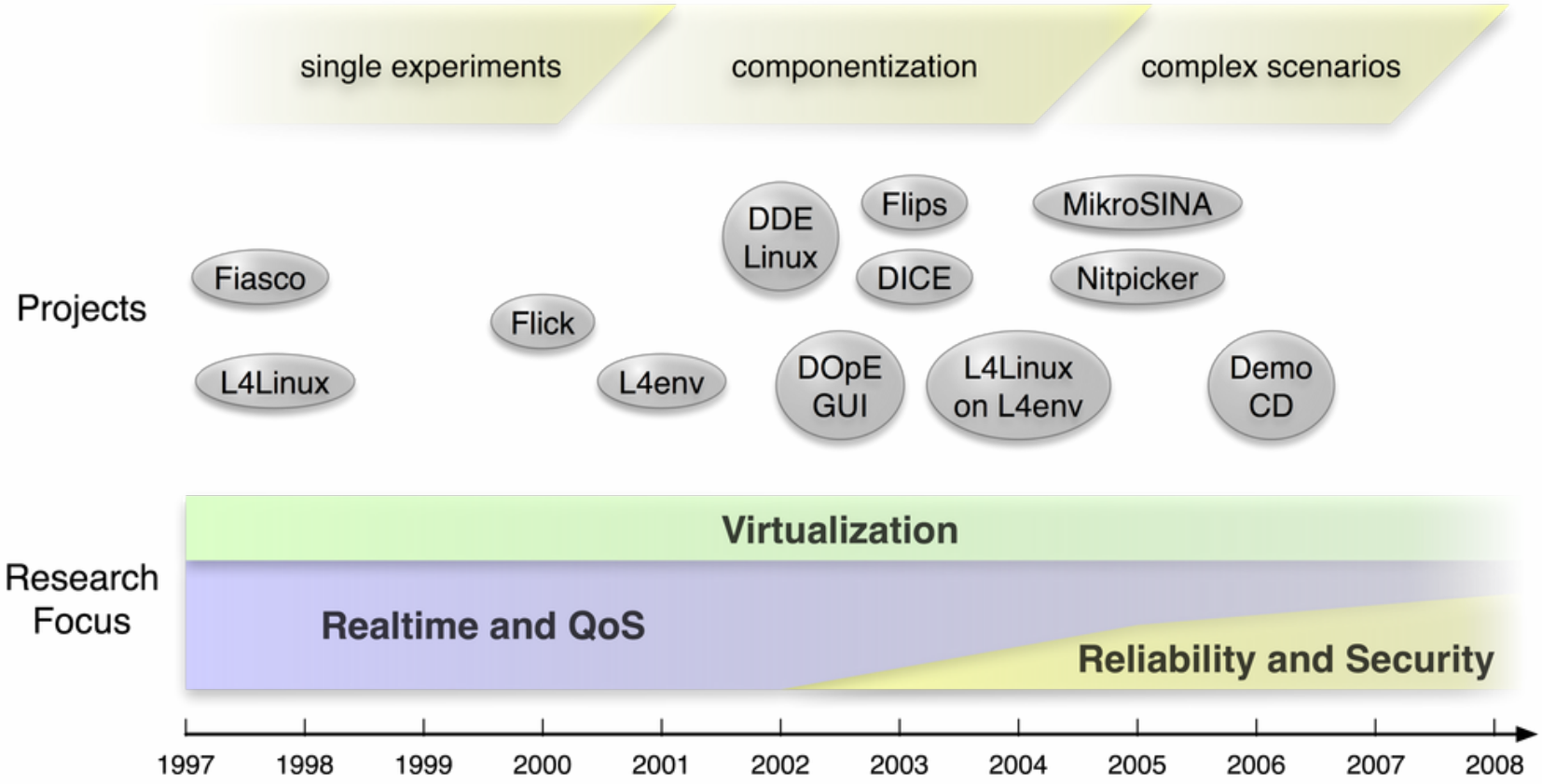
Even more tricky questions

How to...

- ...avoid performance hazards through many indirections?
- ...translate architectural ideas into a real implementation?

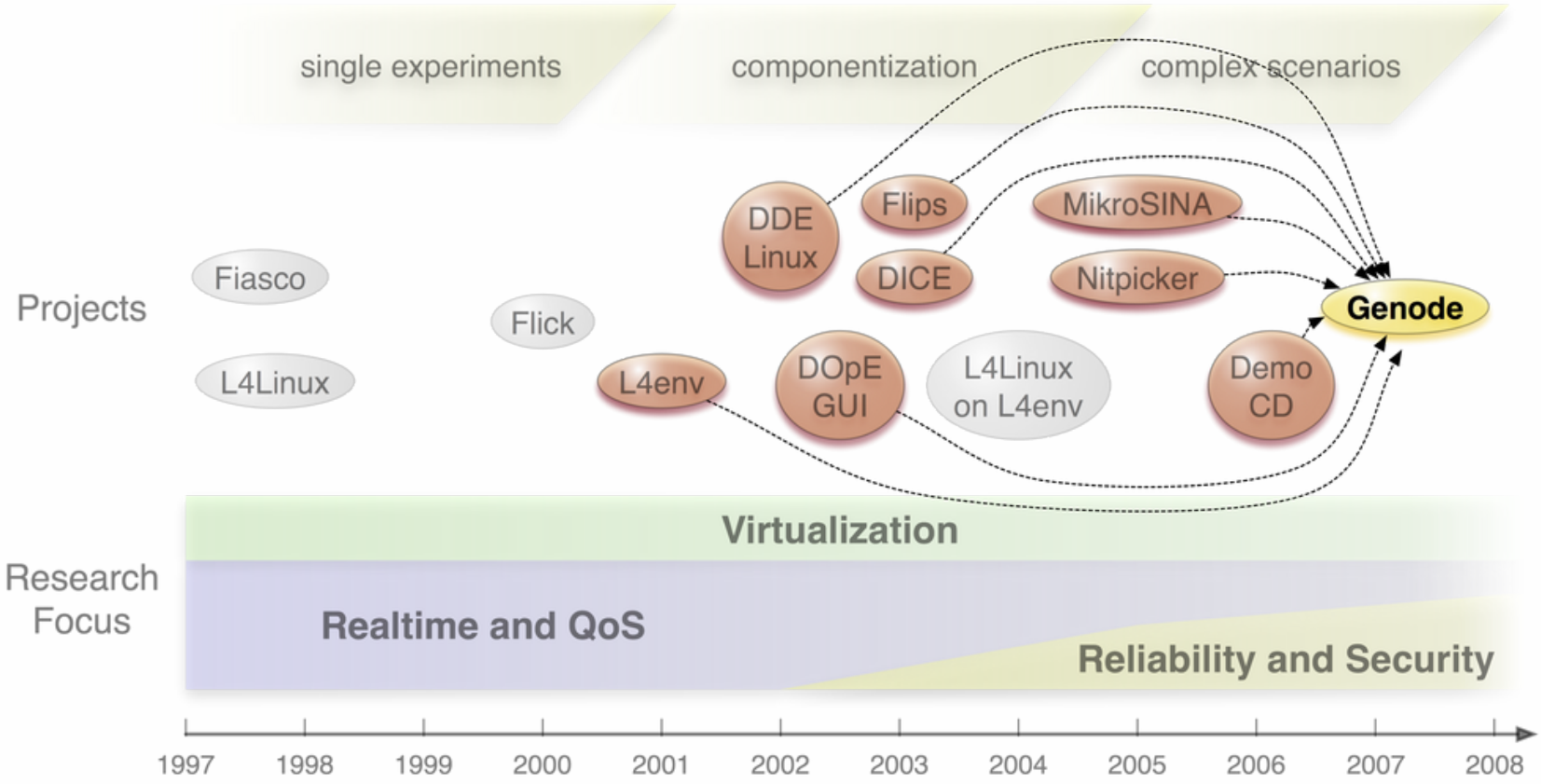
Outline

A bit of history



Research timeline at TU Dresden

A new generation of kernels on the horizon



Genode Labs

- Founded in May 2008, self-funded
- Systems research and development
- Idea: *Start small, build sustainable business, grow organically*
- Foster Open-Source community involvement
- Dual licensing, GPL and commercial licenses

Outline

Object capabilities

Object capabilities

Delegation of rights

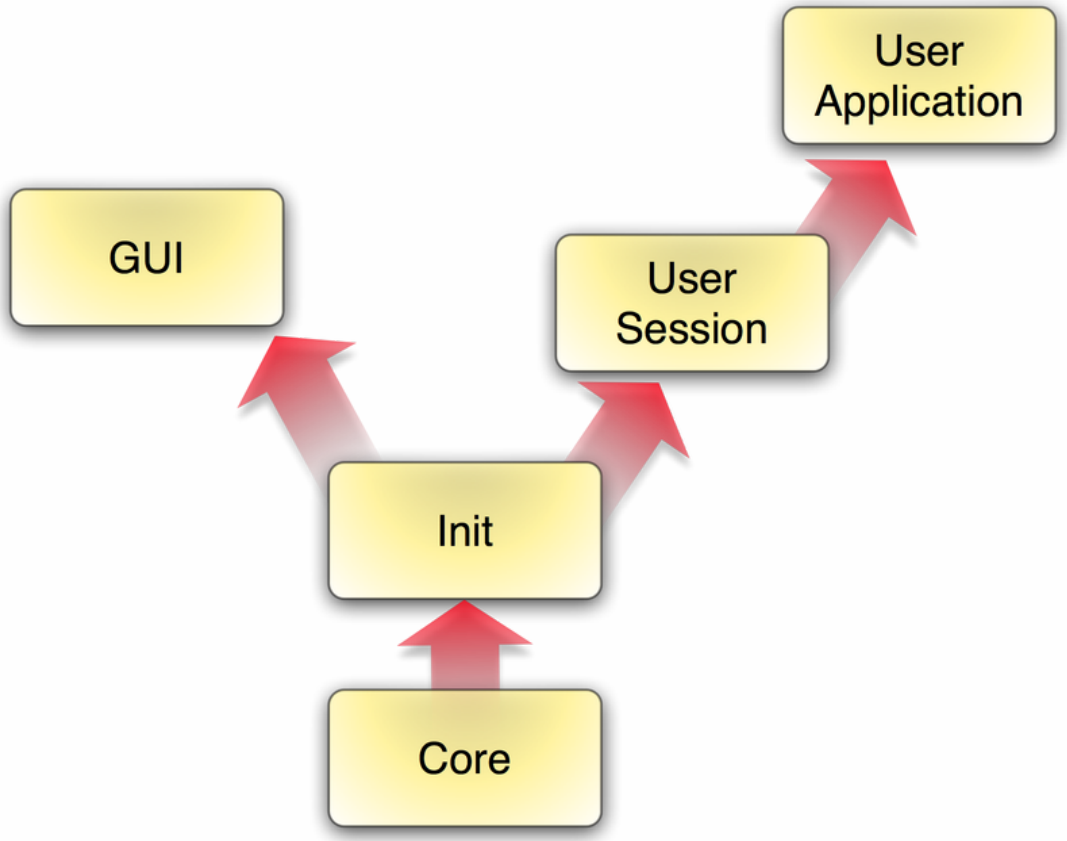
- Each process lives in a virtual environment

Object capabilities

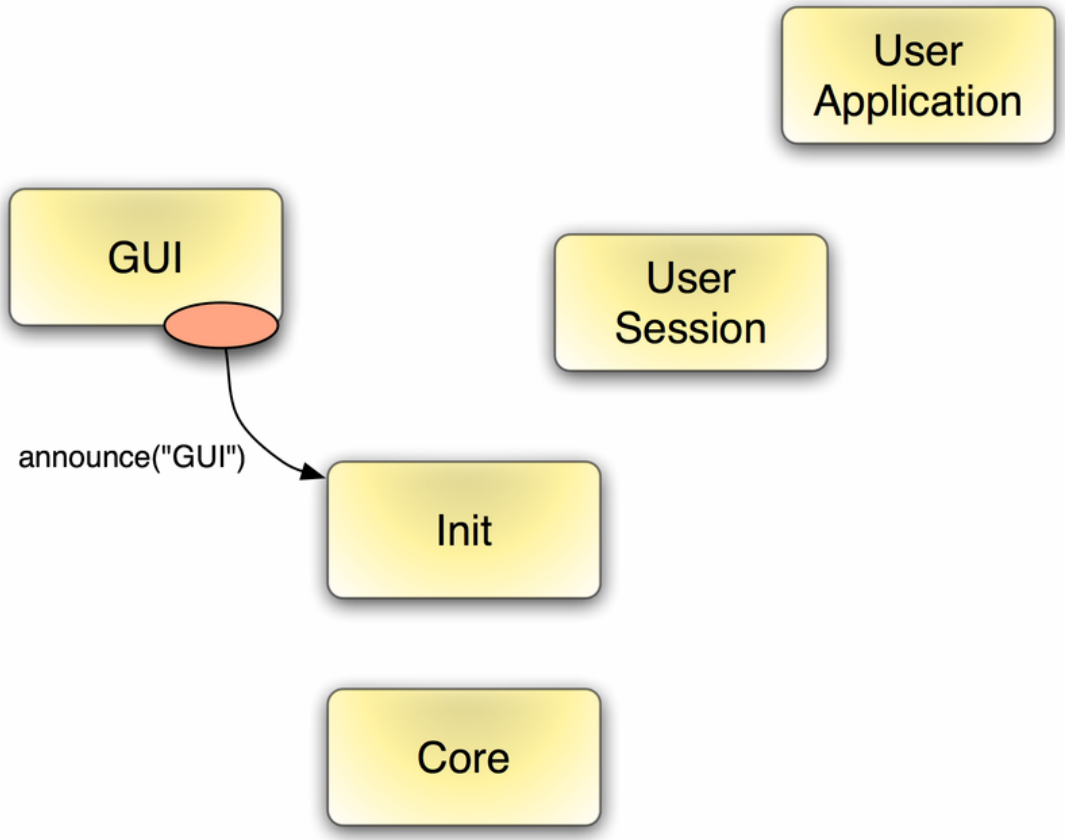
Delegation of rights

- Each process lives in a virtual environment
- A process that possesses a right *(capability)* can
  - Use it *(invoke)*
  - Delegate it to acquainted processes

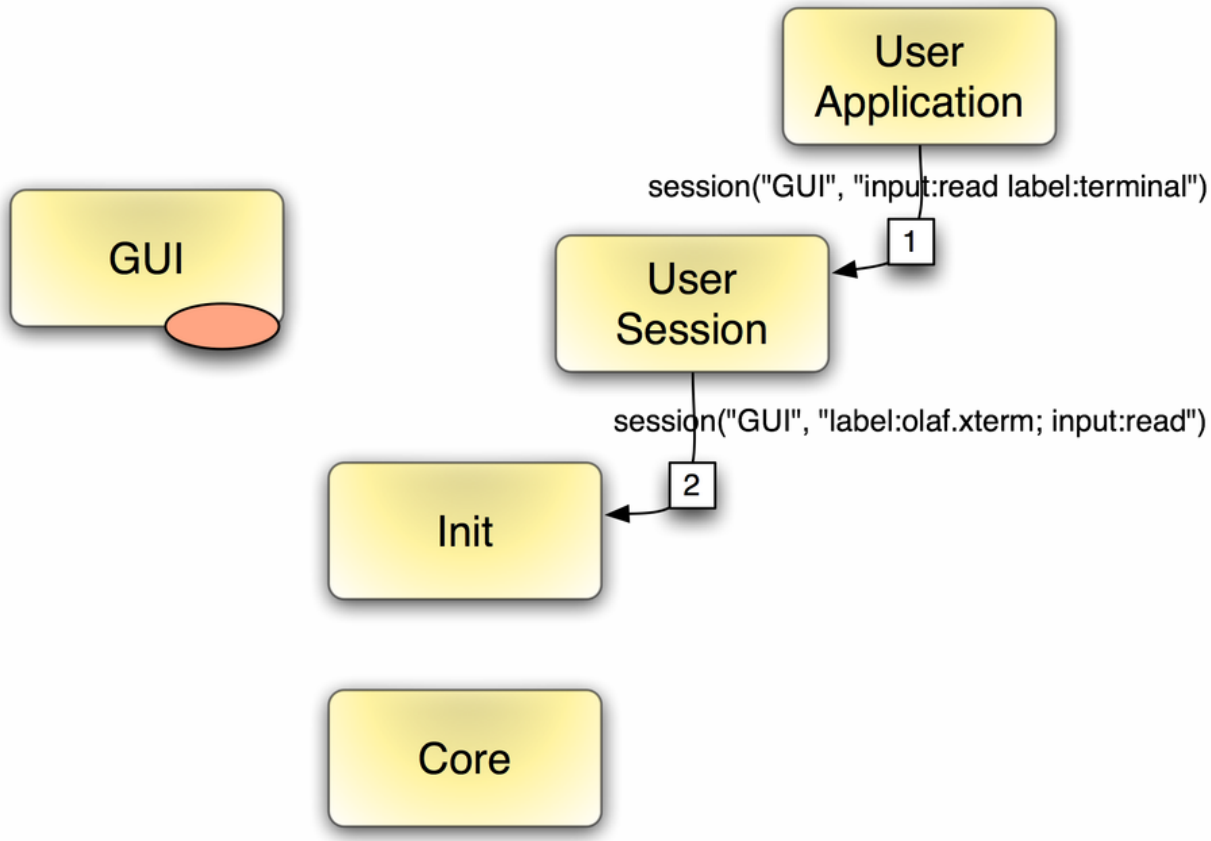
Recursive system structure



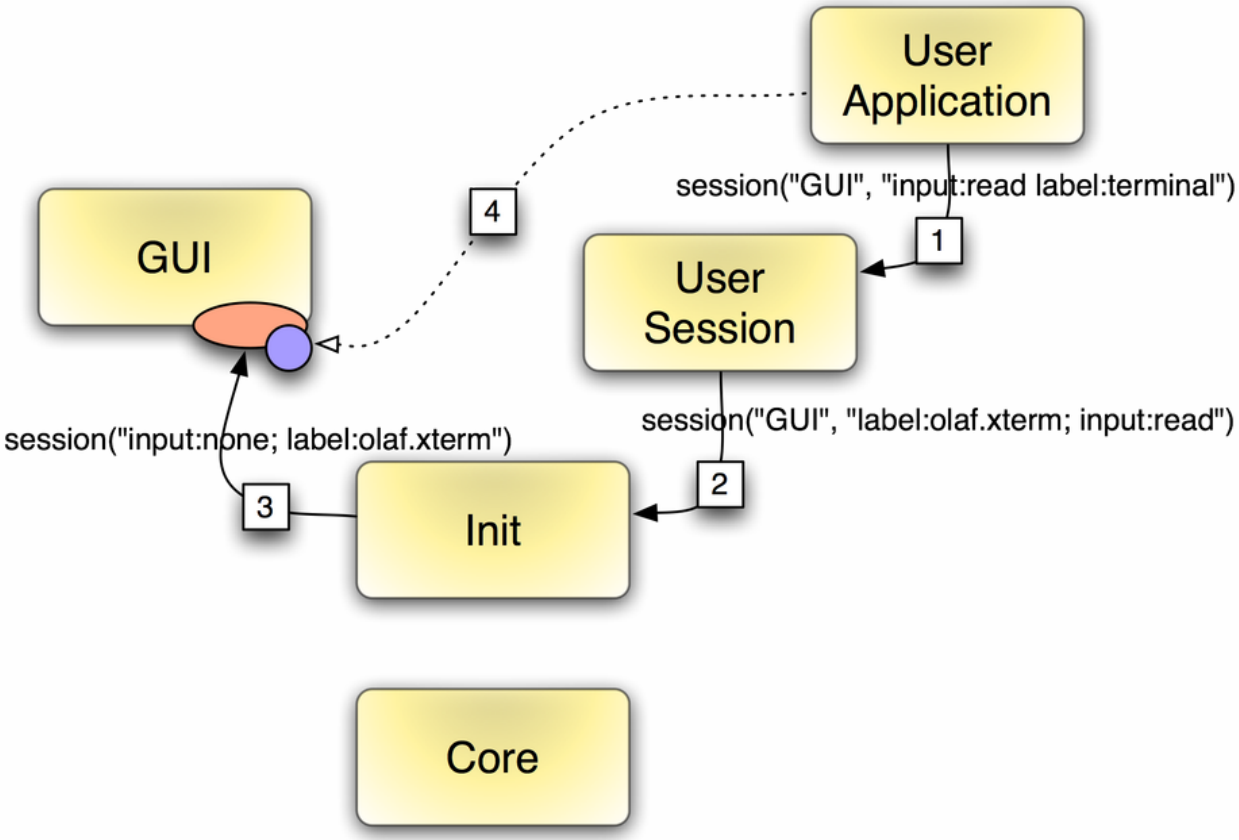
Service announcement



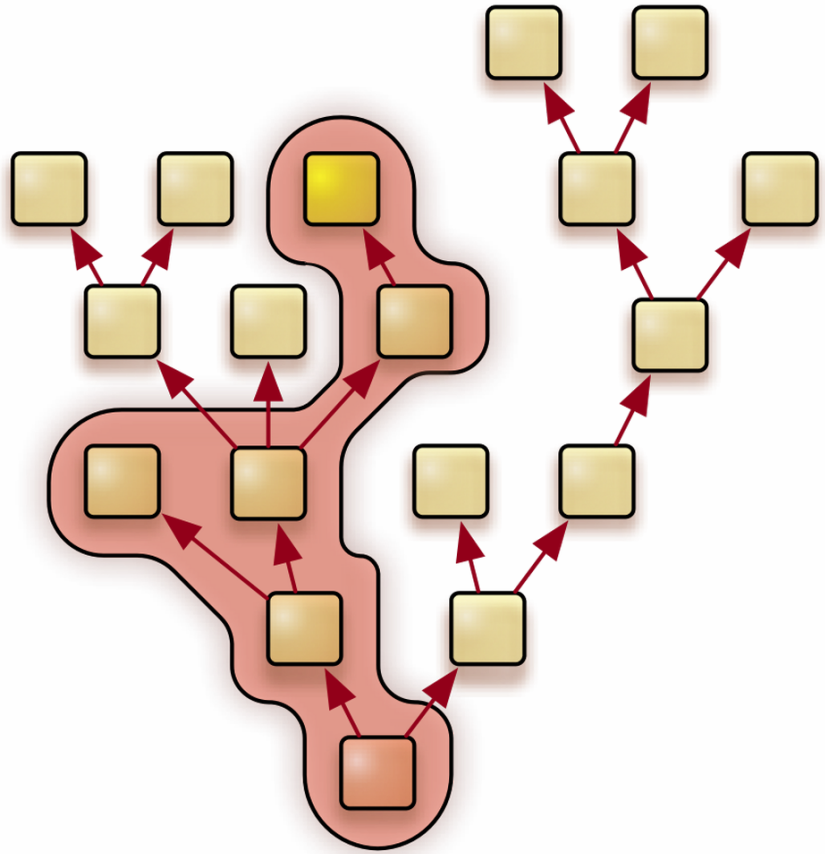
Session creation



Session creation

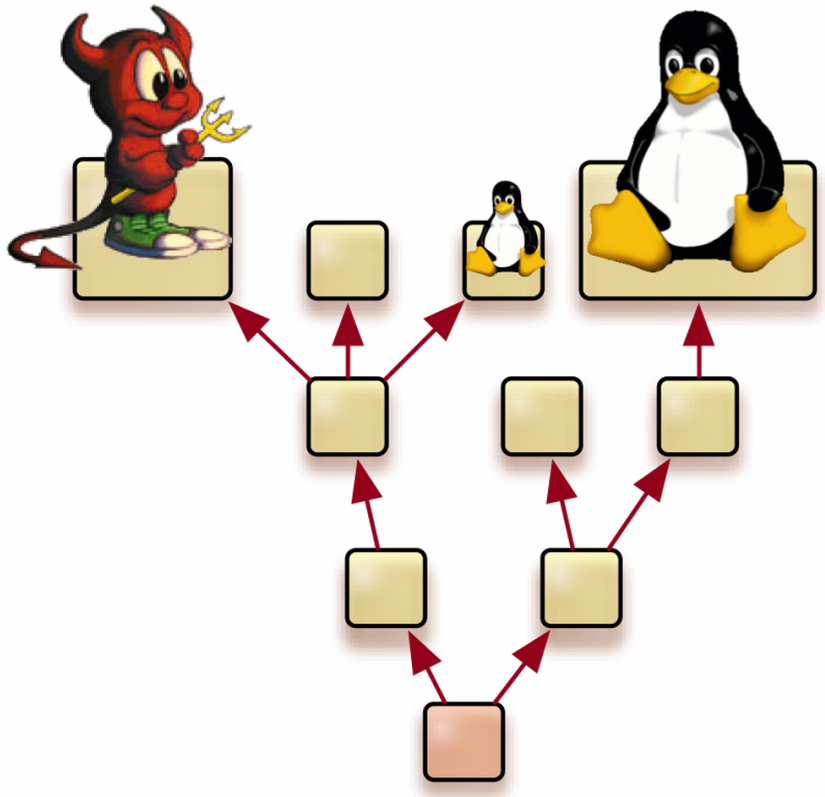


This works recursively



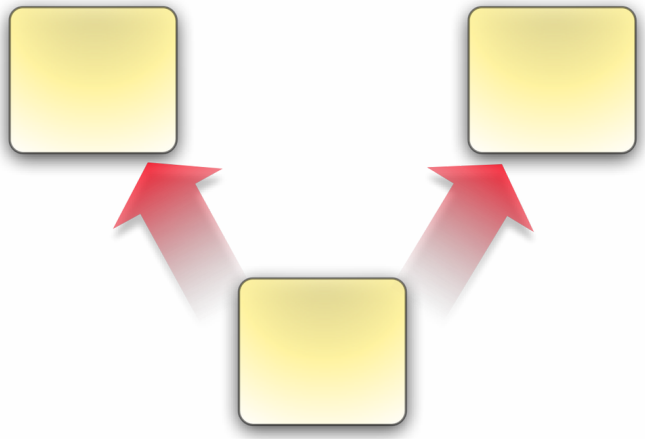
→ Application-specific TCB

Combined with virtualization



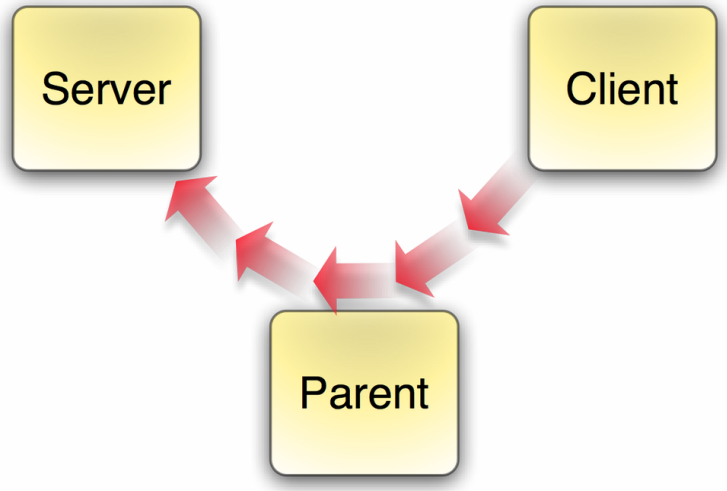
Resource management

Explicit assignment of physical resources to processes



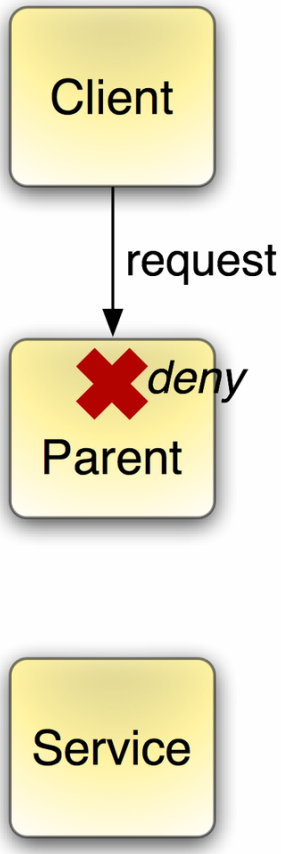
Resource management (II)

Resources can be attached to sessions



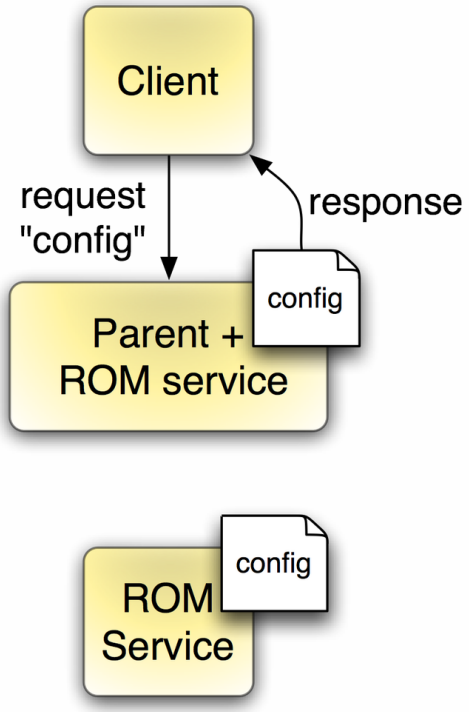
Resource management (III)

Intermediation of resource requests



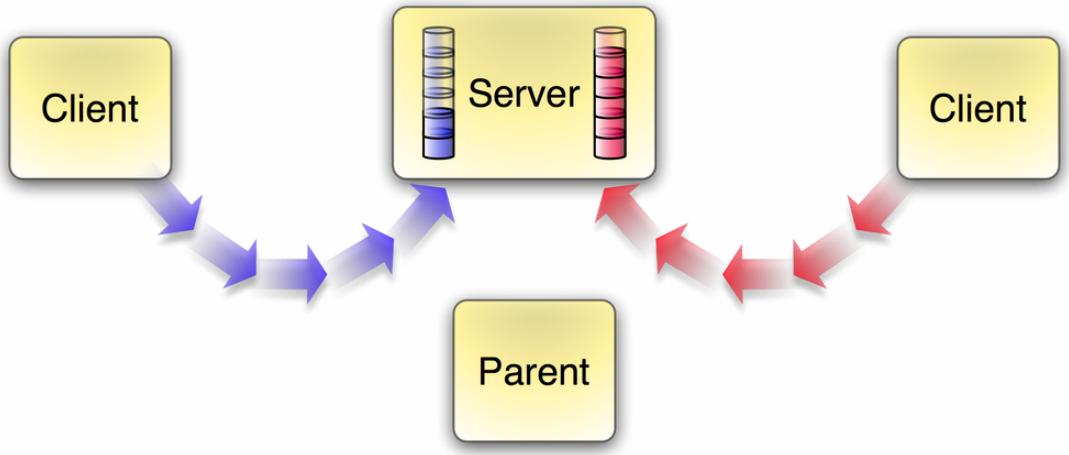
Resource management (IV)

Virtualization of resources

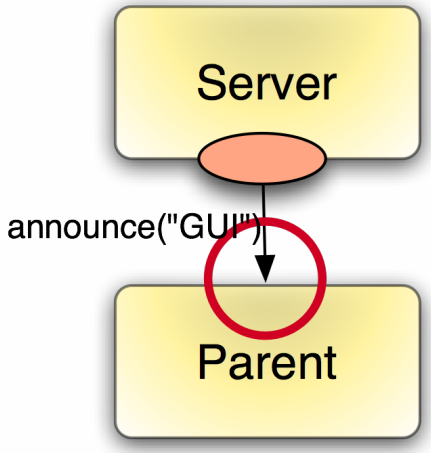


Resource management (V)

Server-side heap partitioning

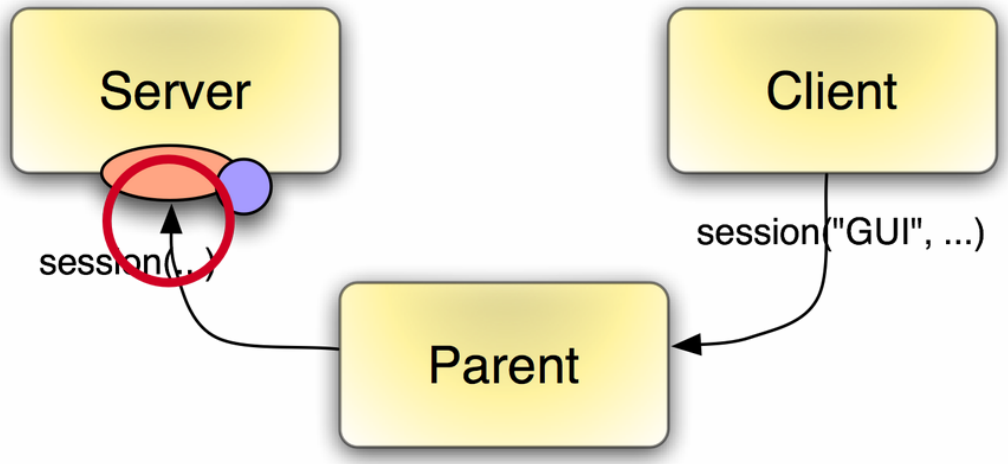


Parent interface




```
void exit(exit_value)

void announce(service_name, root_capability)

session_capability session(service_name, session_args)

void upgrade(to_session_capability, quantum)

void close(session_capability)
```

Root interface




```
session_capability session(session_args)

void upgrade(session_capability, upgrade_args)

void close(session_capability)
```

Outline

Kernels

FIASCO.OC

OKL4

CODEZERO

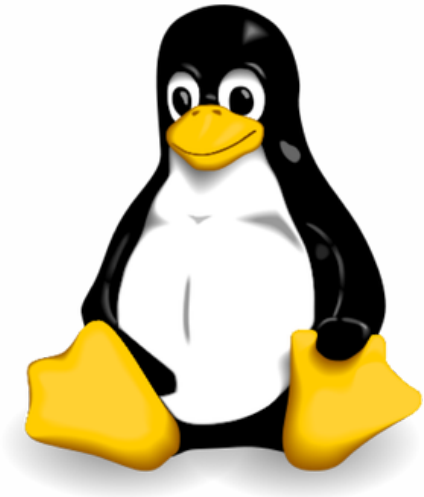
FIASCO

NOVA

Microhypervisor

L4Ka

MicroBlaze



Core - the root of the process tree

- Provides fundamental services
  LOG RAM CAP CPU IO MEM IO PORT IRQ PD ROM RM SIGNAL
- Abstracts physical platform resources
- Policy-free
- Bootstraps the init process

Building blocks

Kernel enables base platform

Device driver translates device interface to API

Protocol stack translates API to API

Application is leaf node in process tree

Runtime environment has one or more children

Resource multiplexer has multiple clients

combinations are possible

Outline

Virtualization techniques

Flavors

Faithful → virtual hardware platform

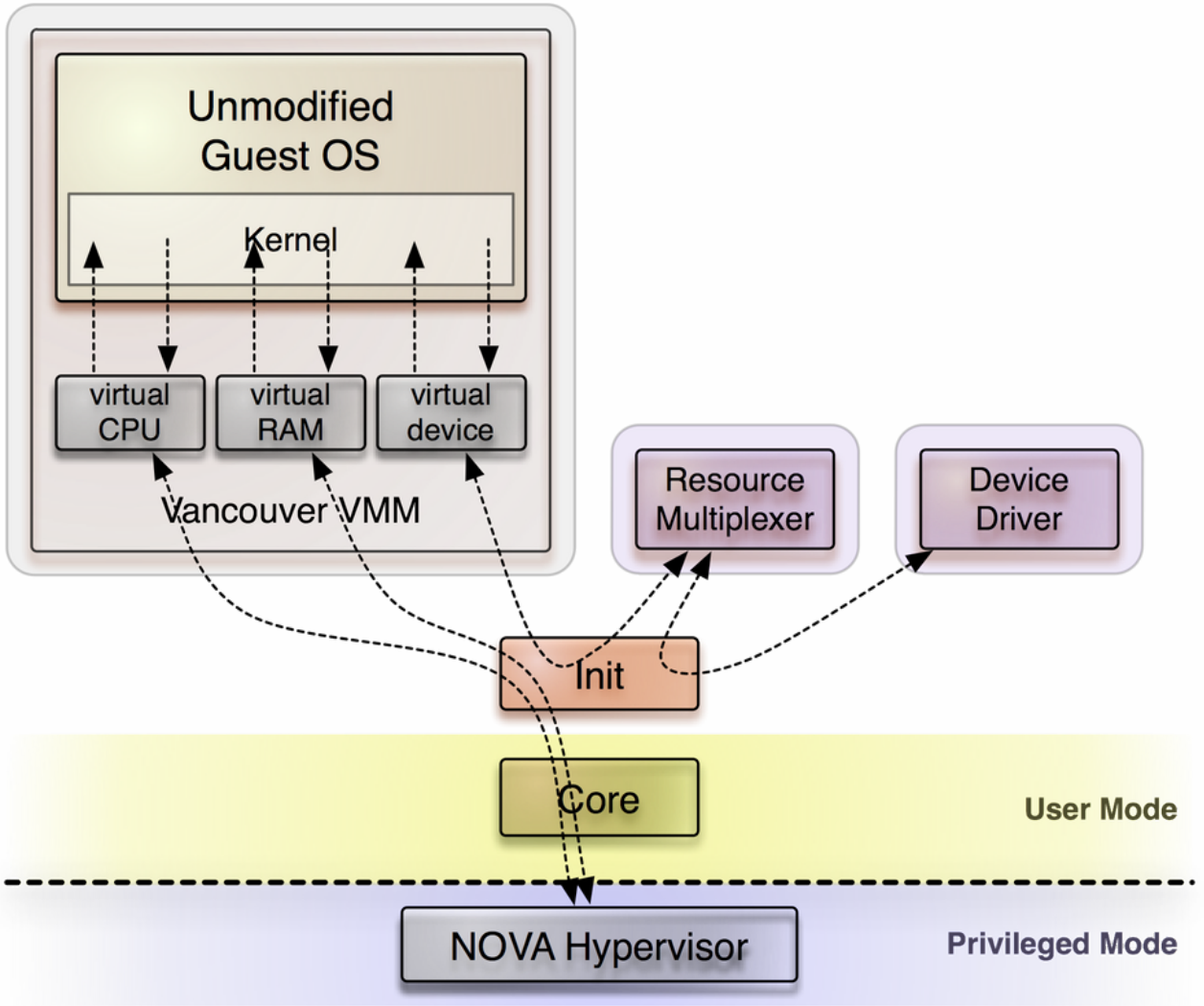
Para → modified guest OS kernel

OS-level → source-level user-land compatibility

Process-level → tailored process environment

C-runtime → tailored runtime within process

Faithful virtualization using Vancouver

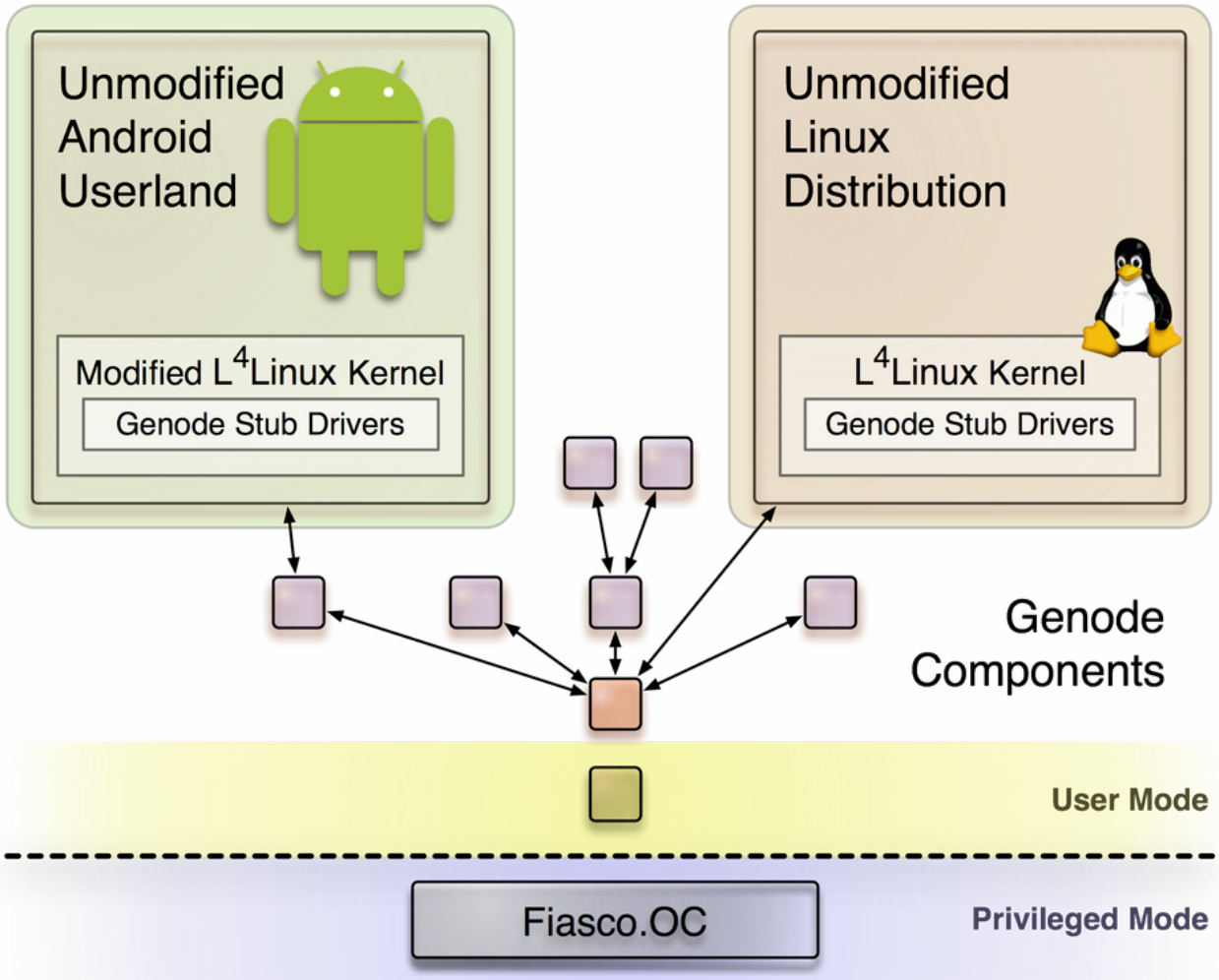


Paravirtualization

- No hardware virtualization support needed
- Guest OS ported to virtual platform
- Binary compatible to guest OS userland
- Guest OS kernel must be modified

L4Linux for Fiasco.OC
OKLinux for OKL4

Paravirtualization (2)



Paravirtualization (3)

Stub drivers

Terminal	→ character device
Block	→ block device
NIC	→ net device
Framebuffer	→ `/dev/fb*` device
Input	→ `/dev/input/*` device
Nitpicker	→ `ioctl` extension to `/dev/fb*` *(OKLinux only)*

no direct hardware access

OS-level virtualization

Idea: Provide Unix kernel interface as a service

OS-level virtualization

Idea: Provide Unix kernel interface as a service

fundamentals

- `write`, `read`
- `stat`, `lstat`, `fstat`, `fcntl`
- `ioctl`
- `open`, `close`, `lseek`
- `dirent`
- `getcwd`, `fchdir`
- `select`
- `execve`, `fork`, `wait4`
- `getpid`
- `pipe`
- `dup2`
- `unlink`, `rename`, `mkdir`

OS-level virtualization

Idea: Provide Unix kernel interface as a service

fundamentals

- `write`, `read`
- `stat`, `lstat`, `fstat`, `fcntl`
- `ioctl`
- `open`, `close`, `lseek`
- `dirent`
- `getcwd`, `fchdir`
- `select`
- `execve`, `fork`, `wait4`
- `getpid`
- `pipe`
- `dup2`
- `unlink`, `rename`, `mkdir`

networking

- `socket`
- `getsockopt`, `setsockopt`
- `accept`
- `bind`
- `listen`
- `send`, `sendto`
- `recv`, `recvfrom`
- `getpeername`
- `shutdown`
- `connect`
- `getaddrinfo`

OS-level virtualization

Idea: Provide Unix kernel interface as a service

fundamentals

- `write`, `read`
- `stat`, `lstat`, `fstat`, `fcntl`
- `ioctl`
- `open`, `close`, `lseek`
- `dirent`
- `getcwd`, `fchdir`
- `select`
- `execve`, `fork`, `wait4`
- `getpid`
- `pipe`
- `dup2`
- `unlink`, `rename`, `mkdir`

networking

- `socket`
- `getsockopt`, `setsockopt`
- `accept`
- `bind`
- `listen`
- `send`, `sendto`
- `recv`, `recvfrom`
- `getpeername`
- `shutdown`
- `connect`
- `getaddrinfo`

In contrast, Linux has more than 300 syscalls

OS-level virtualization (2)

Things we don't need to consider

OS-level virtualization (2)

Things we don't need to consider

- Interaction with device drivers

OS-level virtualization (2)

Things we don't need to consider

- Interaction with device drivers
- Unix initialization sequence

OS-level virtualization (2)

Things we don't need to consider

- Interaction with device drivers
- Unix initialization sequence
- Users, groups
  Instance never shared by multiple users
  The opposite: One user may run many instances

OS-level virtualization (2)

Things we don't need to consider

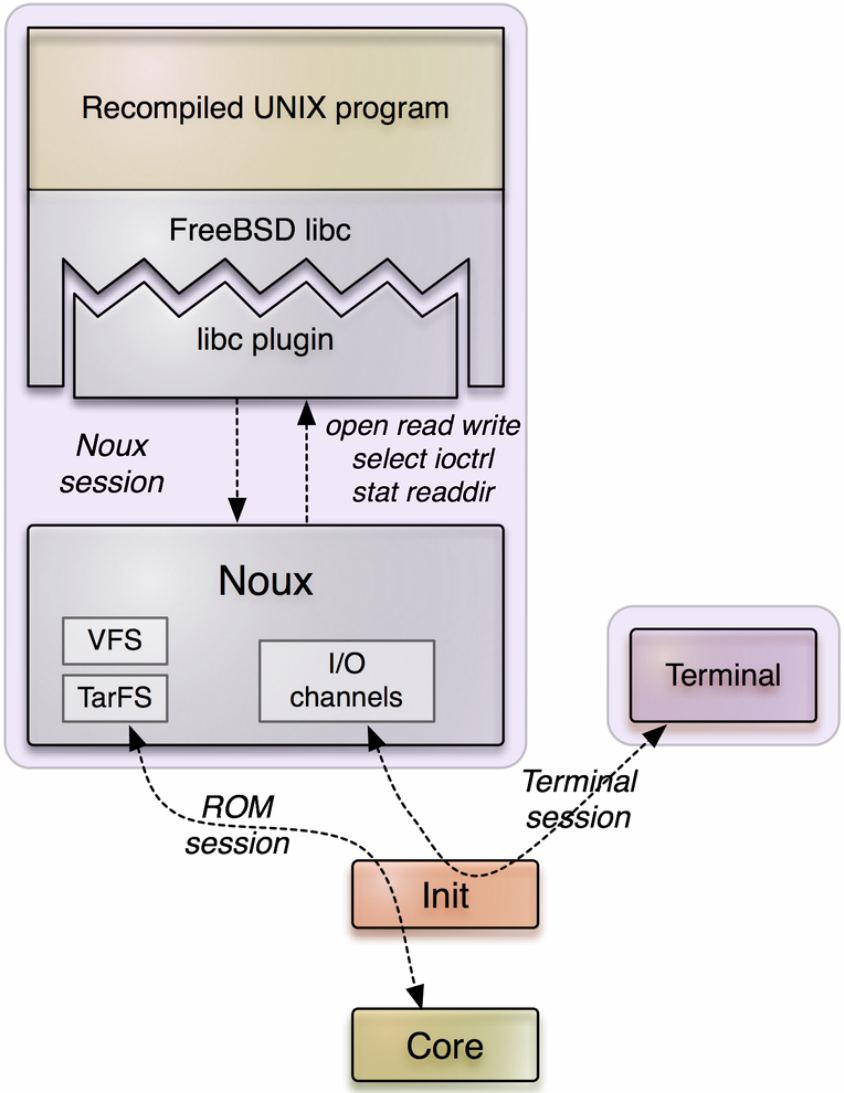
- Interaction with device drivers
- Unix initialization sequence
- Users, groups
  Instance never shared by multiple users
  The opposite: One user may run many instances
- Multi-threading

OS-level virtualization (2)

Things we don't need to consider

- Interaction with device drivers
- Unix initialization sequence
- Users, groups
  Instance never shared by multiple users
  The opposite: One user may run many instances
- Multi-threading
- Scalability of a single instance
  Each instance serves one specific (limited) purpose
  Run many instances in order to scale!

OS-level virtualization (3)



Noux features

- Executes unmodified GNU software
  Bash, VIM, GCC, Coreutils, Lynx, GDB...

Noux features

- Executes unmodified GNU software
  Bash, VIM, GCC, Coreutils, Lynx, GDB...

- Stacked file systems

Noux features

- Executes unmodified GNU software
  Bash, VIM, GCC, Coreutils, Lynx, GDB...
- Stacked file systems
- Instance starts in fraction of a second

Noux features

- Executes unmodified GNU software
  Bash, VIM, GCC, Coreutils, Lynx, GDB...
- Stacked file systems
- Instance starts in fraction of a second
- Uses original GNU build system
  $\rightarrow$ *Porting software is easy*

Noux features

- Executes unmodified GNU software
  Bash, VIM, GCC, Coreutils, Lynx, GDB...
- Stacked file systems
- Instance starts in fraction of a second
- Uses original GNU build system
  $\rightarrow$ *Porting software is easy*

less than 5,000 LOC

Process-level virtualization

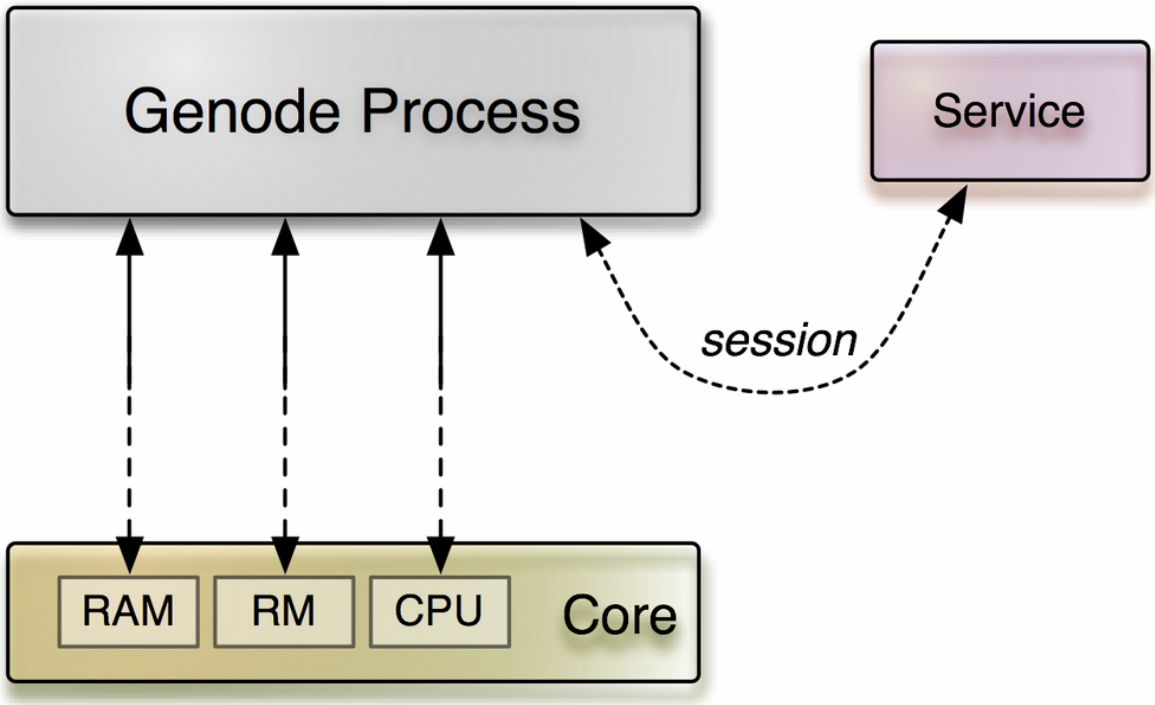
Opportunity: Virtualization of individual session interfaces

- Monitoring of session requests
- Customization of existing services

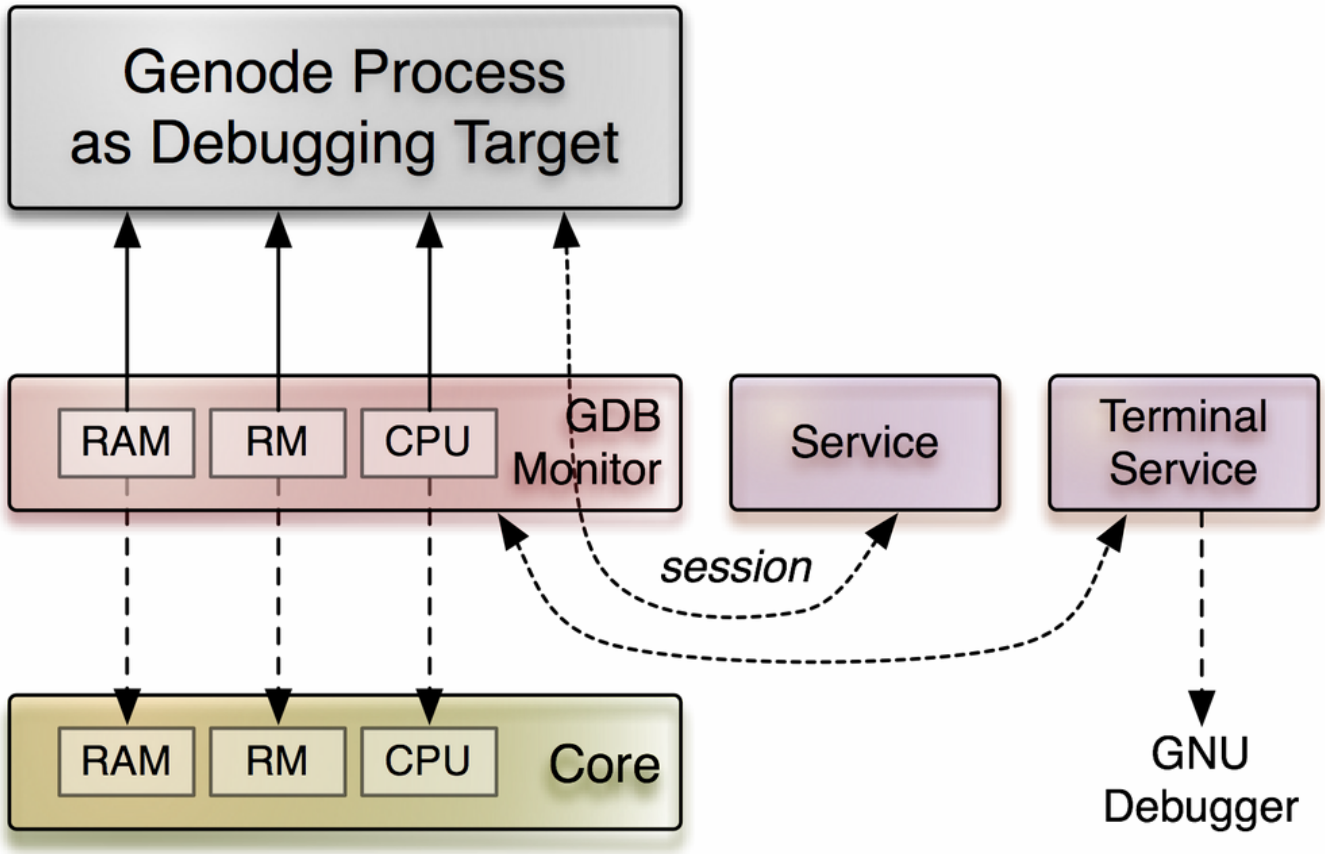
→ Reuse of existing components

→ Separation of policy and mechanisms

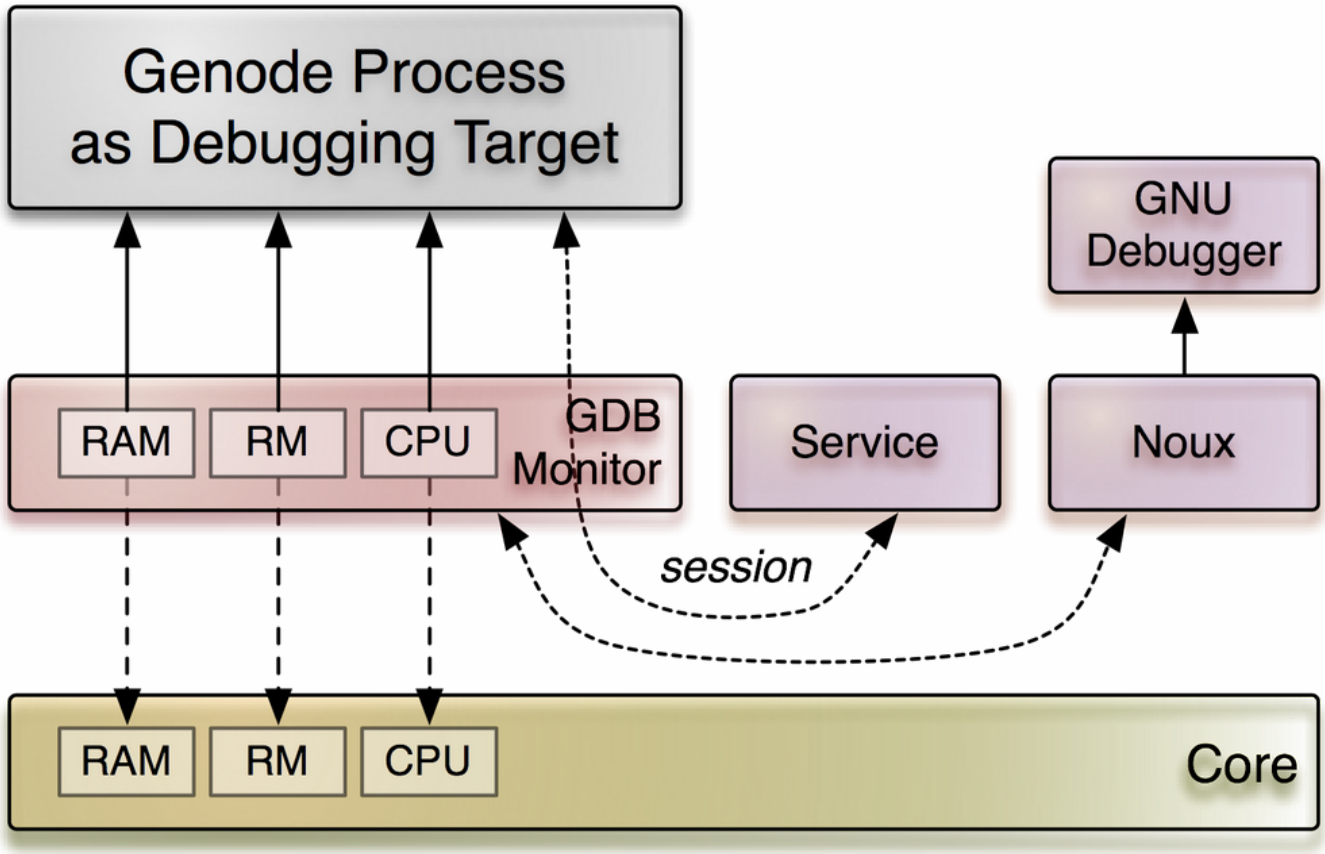
Process-level virtualization (2)



Process-level virtualization (3)



Process-level virtualization (4)



C-runtime customization

FreeBSD libc turned into modular C runtime

`libports/lib/mk/libc.mk`

`libports/lib/mk/libc_log.mk`

`libports/lib/mk/libc_fs.mk`

`libports/lib/mk/libc_rom.mk`

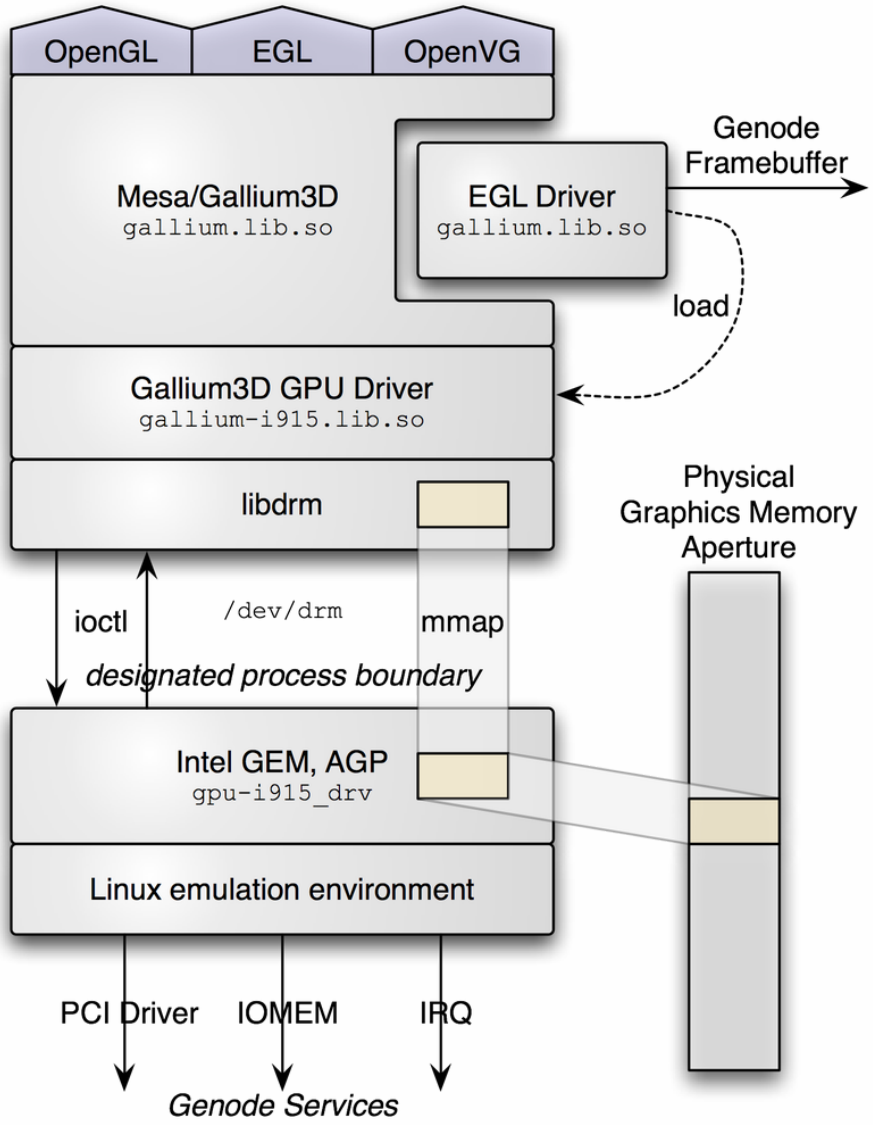
`libports/lib/mk/libc_lwip.mk`

`libports/lib/mk/libc_ffat.mk`

`libports/lib/mk/libc_lock_pipe.mk`

→ *application-specific plugins*

C-runtime customization example



Enslaving services

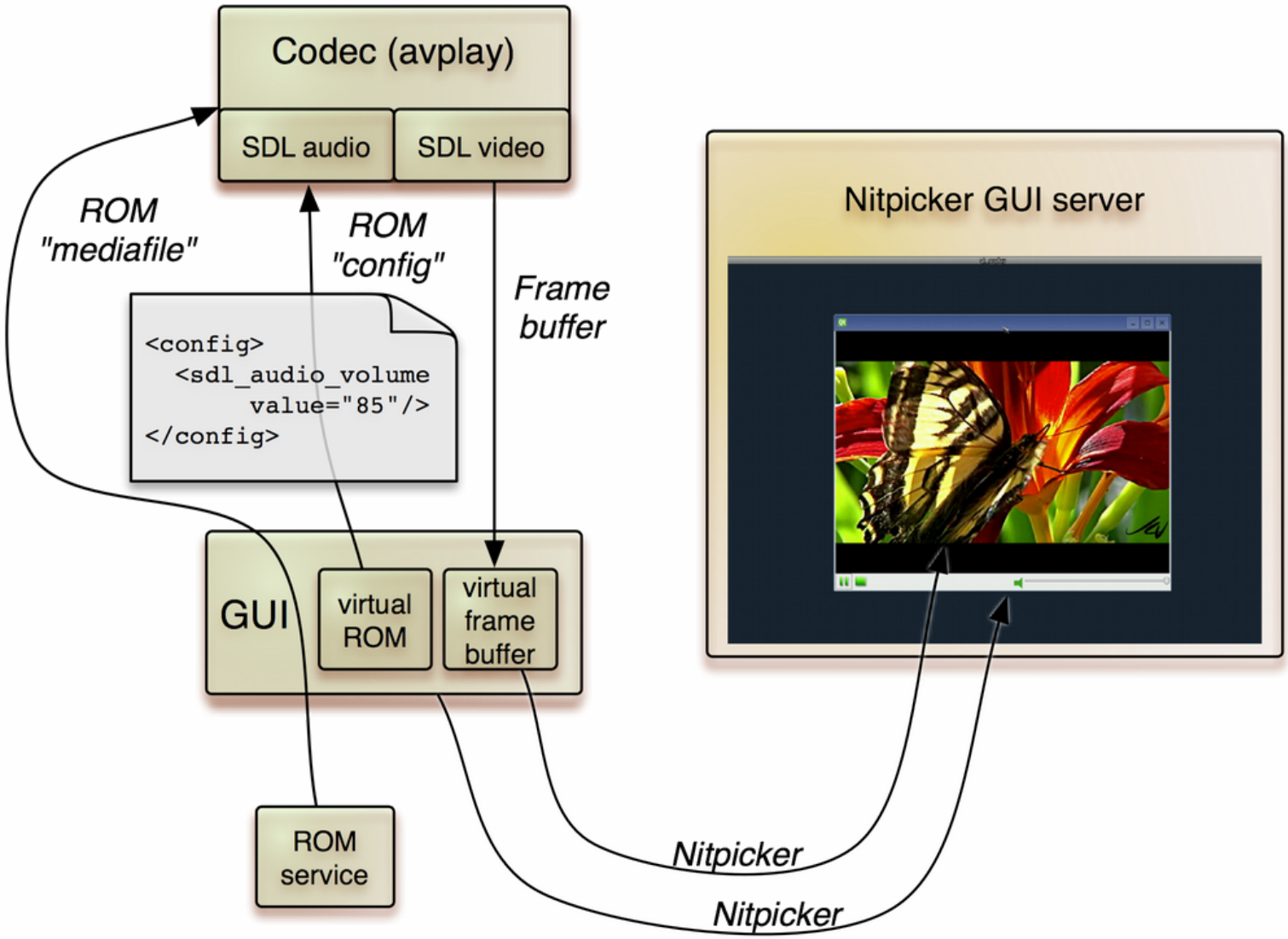
Idea: Run a service as a child

Enslaving services

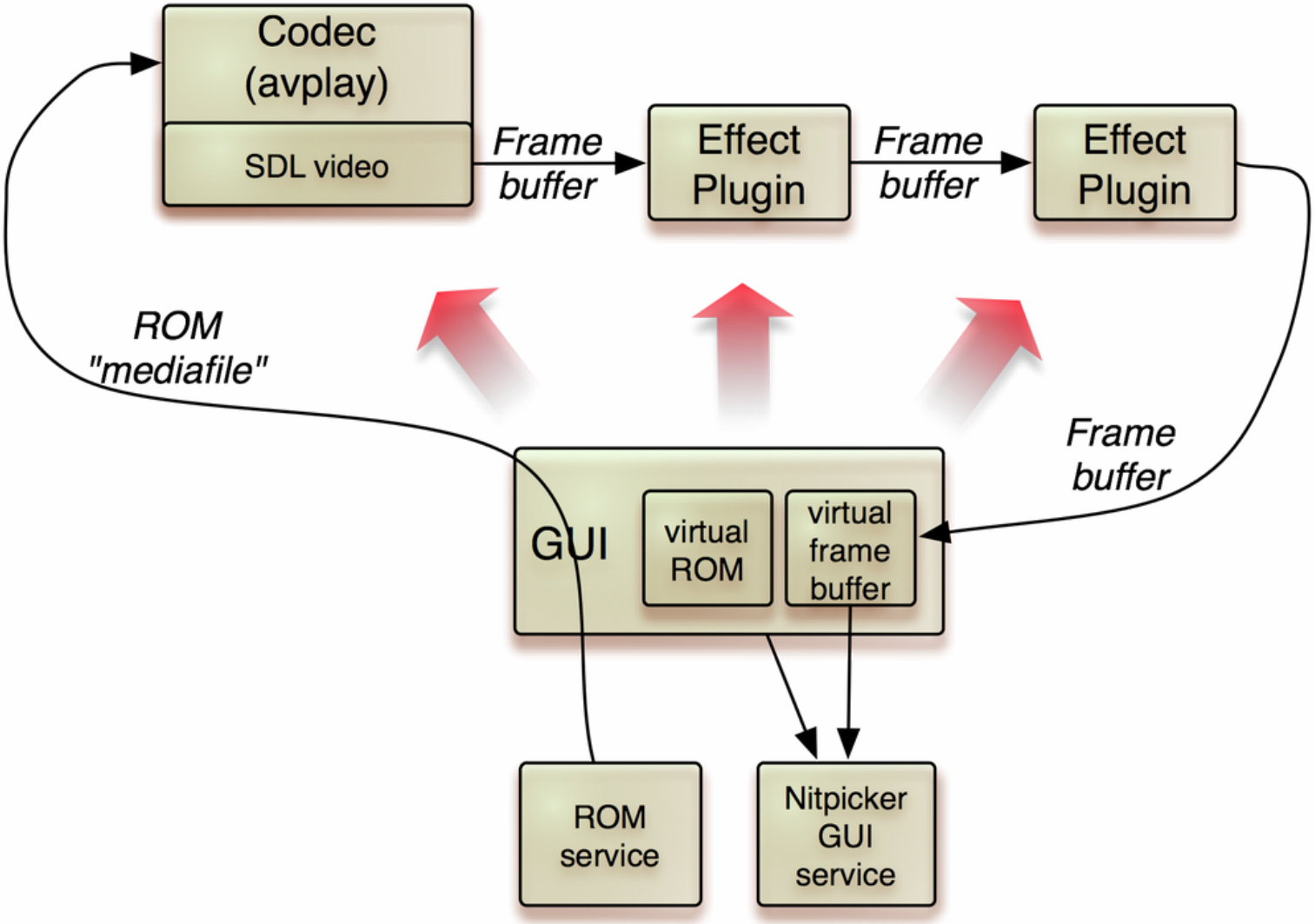
Idea: Run a service as a child

- Sandboxing
- Easy code reuse
- Plugin mechanism

Media player

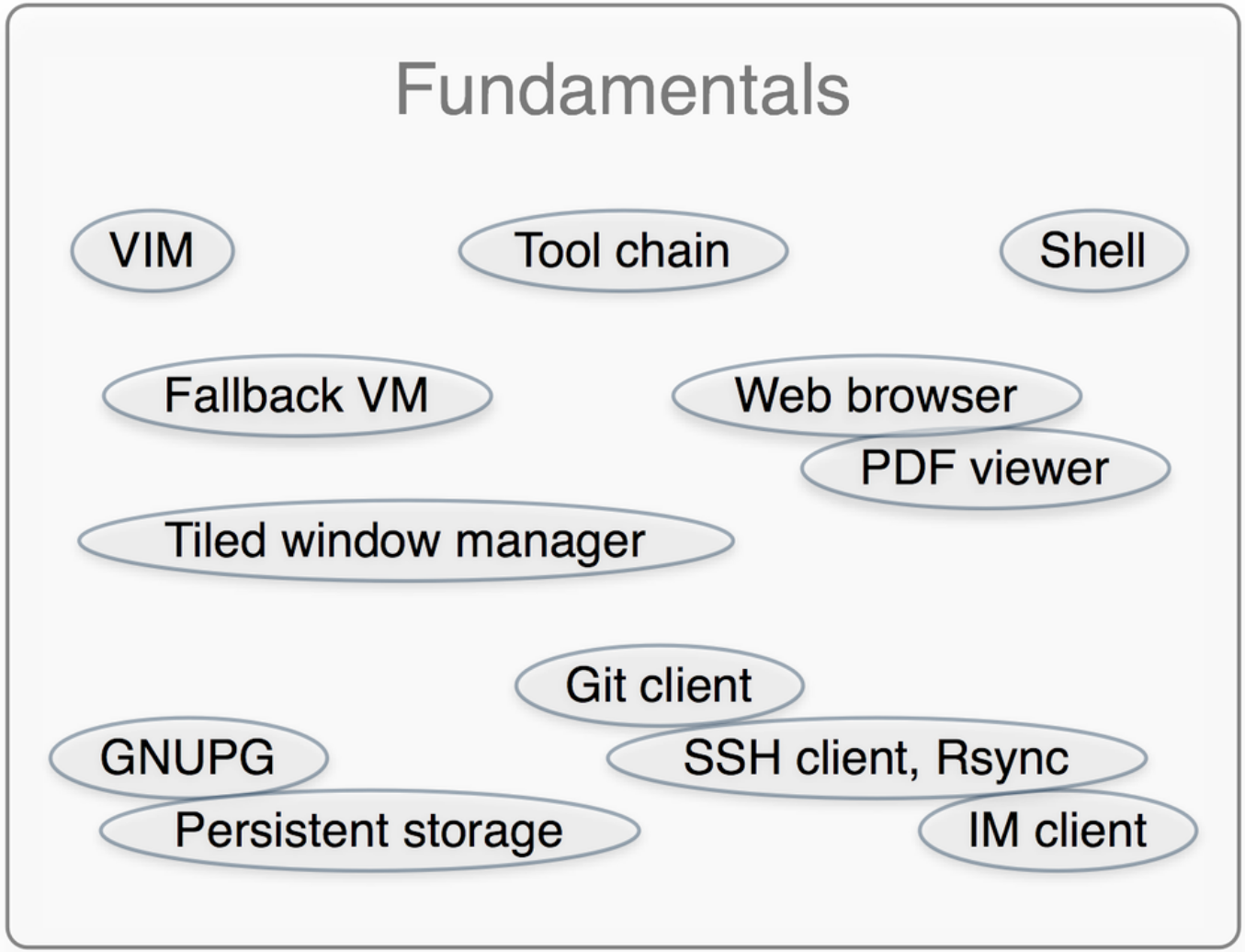


Media player (2)

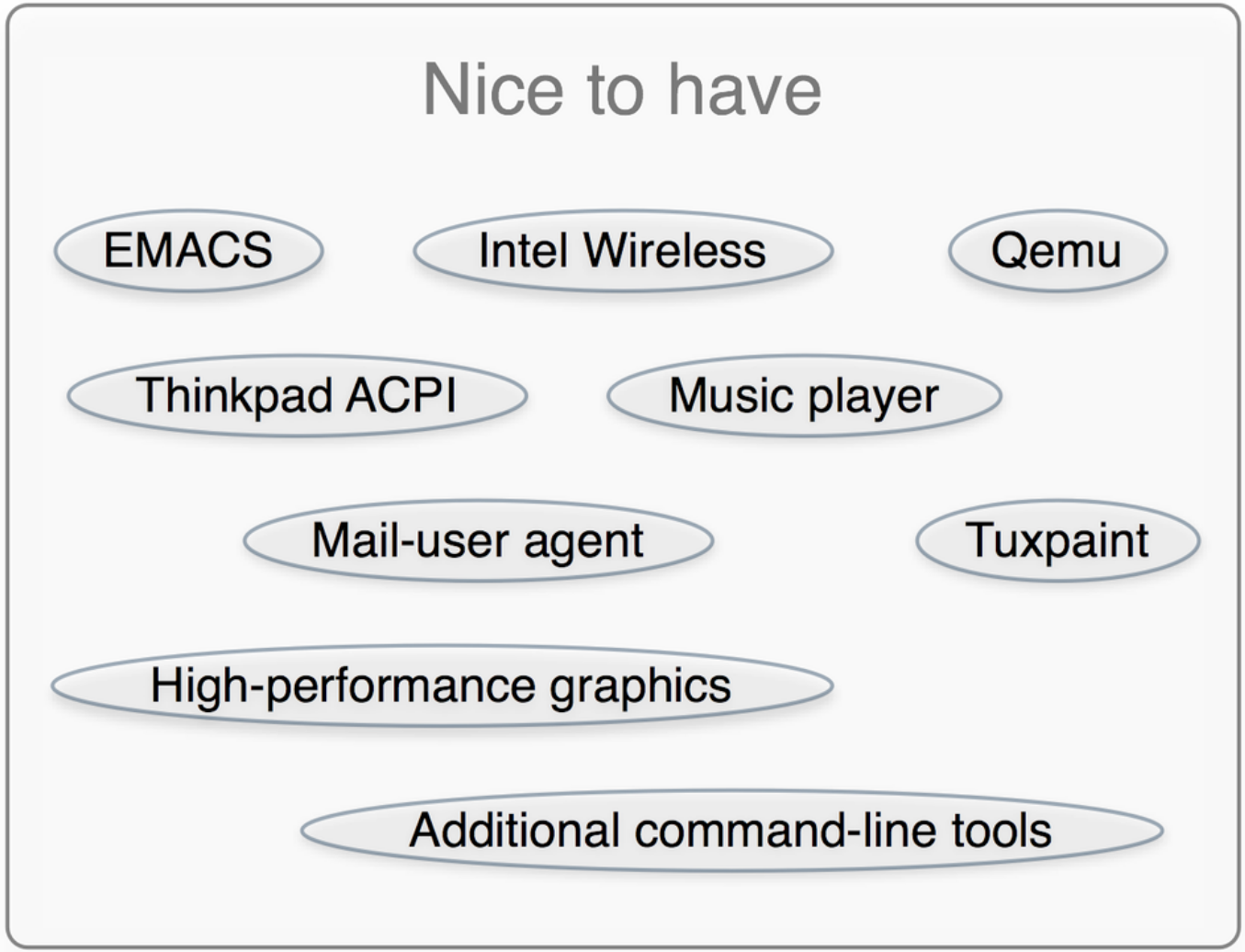


Outline

Eating our own dog food



Eating our own dog food



Performance and scalability

- Multi-processor support
  - NUMA
  - Challenge: Platform-independent API
  - Facilitating Genode's recursive structure

Performance and scalability

- Multi-processor support
  - NUMA
  - Challenge: Platform-independent API
  - Facilitating Genode's recursive structure

- Storage
  I/O scheduling, caching

Performance and scalability

- Multi-processor support
  - NUMA
  - Challenge: Platform-independent API
  - Facilitating Genode's recursive structure
- Storage
  I/O scheduling, caching
- Networking (i. e., TCP/IP performance)

Performance and scalability

- Multi-processor support
  - NUMA
  - Challenge: Platform-independent API
  - Facilitating Genode's recursive structure
- Storage
  I/O scheduling, caching
- Networking (i. e., TCP/IP performance)
- Tools
  Profiling, debugging, tracing

Networking and security

Networking and security

- IOMMU support on NOVA

Networking and security

- IOMMU support on NOVA
- Trusted computing

  → Network of Genode systems

Networking and security

- IOMMU support on NOVA
- Trusted computing

  $\rightarrow$ Network of Genode systems
- Capability-based security on Linux

Thank you

Genode OS Framework

http://genode.org

Genode Labs GmbH

http://www.genode-labs.com

Source code at GitHub

http://github.com/genodelabs/genode