Using Genode as Enabler for Research on Modern Operating Systems

Bruxelles, February 5, 2023

Michael Müller
About me

- Studied computer science at TU Dortmund
- Graduated with MSc in 2018
- Since 2018 PhD student at ESS Group at Osnabrück University
- Full-time research assistant in the MxKernel project at TU Dortmund and Osnabrück University
- Focused on research for many-core OSes
Assumptions from the past … that shaped today’s OSes

- There is only a single CPU.
- Only the CPU computes.
- Context-switches are cheap.
- Main memory is scarce.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

- There is only a **single** CPU.
- Only the CPU computes.
- Context-switches are cheap.
- Main memory is scarce.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

There are only many CPUs.

Main memory is scarce

Only the CPU computes

The memory architecture is uniform

Context-switches are cheap

I/O is slower than the CPU
Assumptions from the past ... that shaped today’s OSes

- There **are** only a single many CPUs.
- Only the CPU computes
- Context-switches are cheap
- Main memory is scarce
- The memory architecture is uniform
- I/O is slower than the CPU
Assumptions from the past … that shaped today’s OSes

- There are only a single many CPUs.
- Only Not just the CPU computes
- Context-switches are cheap
- Main memory is scarce
- The memory architecture is uniform
- I/O is slower than the CPU
Assumptions from the past … that shaped today’s OSes

- There are not just many CPUs.
- Only the CPU computes.
- Context-switches are cheap.
- Main memory is scarce.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

- There are only **many** CPUs.
- Only **not just** the CPU computes.
- Context-switches are **not** cheap anymore.
- Main memory is scarce.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

- There are only many CPUs.
- Only not just the CPU computes.
- Context-switches are not cheap anymore.

- Main memory is scarce.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

- There are only **many** CPUs.
- Only **not just** the CPU computes.
- Context-switches are **not cheap** anymore.
- Main memory is **scarce abundant**.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past ... that shaped today’s OSes

- There **are** only many CPUs.
- Only **Not just** the CPU computes.
- Context-switches are **not** cheap anymore.
- Main memory is **scarce** abundant.
- The memory architecture is uniform.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

- There are **only** many CPUs.
- Only **not just** the CPU computes.
- Context-switches are **not** cheap anymore.
- Main memory is **scarce** abundant.
- The memory architecture is **uniform** heterogeneous.
- I/O is slower than the CPU.
Assumptions from the past ... that shaped today’s OSes

- There are only many CPUs.
- Only not just the CPU computes.
- Context-switches are not cheap anymore.
- Main memory is scarce abundant.
- The memory architecture is uniform heterogeneous.
- I/O is slower than the CPU.
Assumptions from the past … that shaped today’s OSes

- There are only a single many CPUs.
- Only Not just the CPU computes.
- Context-switches are not cheap anymore.
- Main memory is scarce abundant.
- The memory architecture is uniform heterogeneous.
- I/O is slower than as fast as the CPU.
The truth about modern computers

- There are is only a single many CPUs.
- Only Not just the CPU computes
- Context-switches are not cheap anymore
- Main memory is scarce abundant
- The memory architecture is uniform heterogeneous
- I/O is slower than as fast as the CPU

⇒ We need further research on operating systems
Hurdles on the way … what puts OS research at risk

Non-free licensing
- prevents full understanding
- modified system not publishable

Hardware blackboxes
- hinder implementing drivers
- hamper in-depth evaluation

NDAs
- severely restrict publications
- may suppress unfavored results

Missing documentation
- increases evaluation and implementation effort
- leads to reverse engineering

Lack of manpower
- tight limit on what can be done
- may endanger project success

Complexity of modern hardware
- increases effort needed for OS engineering and implementation
- impede comprehension
Hurdles on the way … what puts OS research at risk

Non-free licensing
• prevents full understanding
• modified system not publishable

Hardware black boxes
• hinder implementing drivers
• hamper in-depth evaluation
Hurdles on the way … what puts OS research at risk

**Non-free licensing**
- prevents full understanding
- modified system not publishable

**Hardware black boxes**
- hinder implementing drivers
- hamper in-depth evaluation

**NDAs**
- severely restrict publications
- may suppress unfavoured results

**Lack of manpower**
- tight limit on what can be done
- may endanger project success

**Complexity of modern hardware**
- increases effort needed for OS engineering and implementation
- impede comprehension
Hurdles on the way … what puts OS research at risk

Non-free licensing
- prevents full understanding
- modified system not publishable

Hardware black boxes
- hinder implementing drivers
- hamper in-depth evaluation

NDAs
- severely restrict publications
- may suppress unfavoured results

Missing documentation
- increases evaluation and implementation effort
- leads to reverse-engineering

Lack of man-power
- tight limit on what can be done
- may endanger project success

Complexity of modern hardware
- increases effort needed for OS engineering and implementation
- impede comprehension
Hurdles on the way … what puts OS research at risk

- **Non-free licensing**
  - Prevents full understanding
  - Modified system not publishable

- **Hardware black boxes**
  - Hinder implementing drivers
  - Hamper in-depth evaluation

- **NDAs**
  - Severely restrict publications
  - May suppress unfavoured results

- **Missing documentation**
  - Increases evaluation and implementation effort
  - Leads to reverse-engineering

- **Lack of man-power**
  - Puts tight limit on what can be done
  - May endanger project success
Hurdles on the way ... what puts OS research at risk

- **Non-free licensing**
  - prevents full understanding
  - modified system not publishable

- **Hardware black boxes**
  - hinder implementing drivers
  - hamper in-depth evaluation

- **NDAs**
  - severely restrict publications
  - may suppress unfavoured results

- **Missing documentation**
  - increases evaluation and implementation effort
  - leads to reverse-engineering

- **Lack of man-power**
  - puts tight limit on what can be done
  - may endanger project success

- **Complexity of modern hardware**
  - increases effort needed for OS engineering and implementation
  - impede comprehension
SO, WHAT DO RESEARCHERS DO?
Workarounds and Tweaks


… wait, there are even more !
How OS research is done today … mostly

OSDI 2020 — OS research papers

- Linux tweak
- Original OS

OSDI 2021 — OS research papers

- Linux tweak
- Original OS

Source: “It’s Time for Operating Systems to Rediscover Hardware” – Timothy Roscoe, USENIX ATC ’21/OSDI ’21 Joint Keynote
Why hacking Linux isn’t a good idea

- **Huge and complex code-base**
  - Increases effort needed

- **POSIX-compliance**
  - Limits OS abstractions and interfaces
  - Significant changes may break user space

- **Moving target**
  - Constant maintenance required
  - No maintenance ⇒ Extensions will break
Isn’t there something better?
Isn’t there something better?

... maybe some OS framework?
A framework for OS research shall be

- Minimal
- Portable
- Extensible
- Well-documented
- Composable
- Investigable
- Maintainable
A framework for OS research shall be

- Eases understanding
- Simplifies changing kernel primitives
- Helps debugging
A framework for OS research shall be

- Necessary to understand measurements
- Open Source code base
- Provide profiling tools

---

Investigable

Minimal

Extensible

Well-documented

Portable

Composable

Maintainable
A framework for OS research shall be

- Regular updates
- But should **not break** fundamental interfaces
- If so, **only small** changes needed

- Investigable
- Extensible
- Portable
- Well-documented
- Composable
- **Maintainable**
- Minimal
A framework for OS research shall be

- **Easy to write** new extensions
- **Separation of** concerns
- **Well-defined components**
- **Well-defined interface** between components
A framework for OS research shall be

- **Minimal**
- **Portable**
- **Extensible**
- **Well-documented**
- **Investigable**
- **Maintainable**
- **Composable**

- **Ease** getting started
- **Provides** description of **important** components and interfaces
- **Ideally** a book
- **And** documented **code**
A framework for OS research shall be

- Minimal
- Extensible
- Well-documented
- Maintainable
- Composable
- Portable

- Future-proof
- Enable experimental hardware
- Support Hardware/OS co-design
A framework for OS research shall be

- Extensible
- Portable
- Investigable
- Well-documented
- Maintainable
- Composable

- Replaceable OS components
- Allow different OS interfaces simultaneously
- Reusability of drivers etc.
A framework for OS research shall be

- Minimal
- Portable
- Extensible
- Well-documented
- Investigable
- Composable
- Maintainable
A CANDIDATE FOR AN OS FRAMEWORK
The Genode OS Framework
Does Genode fit the bill?

- Minimal
- Investigable
- Extensible
- Well-documented
- Portable
- Composable
- Maintainable

- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all
Does Genode fit the bill?

- **Minimal**
- **Portable**
- **Extensible**
- **Well-documented**
- **Maintainable**
- **Composable**

<table>
<thead>
<tr>
<th>Description</th>
<th>Linux 4.14 (x86 only, no drivers)</th>
<th>Genode kernel with NOVA</th>
<th>NOVA kernel</th>
<th>base-hw kernel</th>
<th>base</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoC</td>
<td>911,059</td>
<td>52,957</td>
<td>21,039</td>
<td>31,918</td>
<td></td>
</tr>
</tbody>
</table>

- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all
Does Genode fit the bill?

- Code Under **AGPL**
- Only **basic** tracing yet
- Only rudimentary **manual** profiling

**Complete**
- Investigable
- Minimal

**Almost Complete**
- Extensible
- Maintainable

**Satisfactory**
- Portable
- Well-documented

**Partial**
- Composable

**Not Yet**
- Not yet
- Not at all

**Not at all**
- Only basic tracing yet
- Only rudimentary manual profiling
Does Genode fit the bill?

- **Quarterly** updates
- Mostly **minimal** changes to kernel API

### Degree of Satisfaction

- **Complete**
- Almost complete
- Satisfactory
- Partial
- Not yet
- Not at all

- **Quarterly updates**
- **Mostly minimal** changes to kernel API
Does Genode fit the bill?

- Clearly separated components
- **Well-defined** RPC interface
- **Minimal** requirements for new components

<table>
<thead>
<tr>
<th></th>
<th>complete</th>
<th>almost complete</th>
<th>satisfactory</th>
<th>partial</th>
<th>not yet</th>
<th>not at all</th>
</tr>
</thead>
</table>

- Extensible
- Investigable
- Maintainable
- Minimal
- Portable
- Well-documented
- Composable
Does Genode fit the bill?

- **Book** “Genode Foundations”
- **Extensive changelog** for each release
- **FOSDEM** talks

<table>
<thead>
<tr>
<th>Category</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Book</td>
<td>complete</td>
</tr>
<tr>
<td>Changelog</td>
<td>complete</td>
</tr>
<tr>
<td>FOSDEM</td>
<td>complete</td>
</tr>
<tr>
<td>Investigable</td>
<td>partial</td>
</tr>
<tr>
<td>Maintenable</td>
<td>almost complete</td>
</tr>
<tr>
<td>Minimal</td>
<td>not complete</td>
</tr>
<tr>
<td>Well-Documented</td>
<td>complete</td>
</tr>
<tr>
<td>Portable</td>
<td>not complete</td>
</tr>
<tr>
<td>Composable</td>
<td>not complete</td>
</tr>
</tbody>
</table>

Using Genode as Enabler for Research on Modern Operating Systems / Michael Müller / 47
Does Genode fit the bill?

- **Strict separation** of hardware platform and OS code
- **Generalized** concepts for hardware features

- **Complete**
- **Almost complete**
- **Satisfactory**
- **Partial**
- **Not yet**
- **Not at all**
Does Genode fit the bill?

- **Component-based architecture**
- **Own OS can be “sculptured”**
- **Multiple** instances of a service possible

<table>
<thead>
<tr>
<th>Feature</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Investigable</td>
<td>complete</td>
</tr>
<tr>
<td>Minimal</td>
<td>almost complete</td>
</tr>
<tr>
<td>Extensible</td>
<td>satisfactory</td>
</tr>
<tr>
<td>Maintainable</td>
<td>partial</td>
</tr>
<tr>
<td>Portable</td>
<td>not yet</td>
</tr>
<tr>
<td>Composable</td>
<td>not at all</td>
</tr>
<tr>
<td>Well-documented</td>
<td></td>
</tr>
</tbody>
</table>
Does Genode fit the bill?

- Investigable
- Extensible
- Portable
- Well-documented
- Composable
- Minimal
- Maintainable
Does Genode fit the bill?

- Investigable
- Extensible
- Well-documented
- Portable
- Composable
- Maintainable

But, how much can it facilitate OS research?
Meet EalánOS

... an experimental implementation of the MxKernel architecture using the Genode OS Framework
EalánOS — Concepts

- CRM
- GUI Server
- Shell
- DBMS
- Webstore
- HTTPD
- FS
- NIC
- Aivot
- Hoitaja

Organisms

Cells

Tasks

Hardware resource
EalánOS — Concepts

Application-specific resource manager ("brain" of an organism)

Organisms
- Resource container for a **user-session**
- **Custom** OS services

Cells

Tasks

Using Genode as Enabler for Research on Modern Operating Systems / Michael Müller / 54
EalánOS — Concepts

Organisms
- Resource container for a user-session
- Custom OS services

Cells

Tasks

Global resource manager ("carer" for organisms)
EalánOS — Concepts

Organisms
- Resource container for a user-session
- Custom OS services

Cells
- Elastic resource container
- Contains a single process

Tasks
EalánOS — Concepts

- **Organisms**
  - Resource container for a *user-session*
  - Custom OS services

- **Cells**
  - *Elastic* resource container
  - Contains a *single* process

- **Tasks**
  - closed units of work
  - *not* preemptable
EalánOS — Architecture

Hardware resources

Tukija (resource provider)
- IPC
- Caps
- CPU
- RAM
- PD
- GPU
- IRQ

Application

Withdraw resource

Grant resource

Using Genode as Enabler for Research on Modern Operating Systems / Michael Müller / 58
How Genode assists in creating a research OS

THE MAKING-OF EALÀNOS
EalánOS — What do we need? What do we have?

- DBMS
- CRM
- Webstore
- HTTPD
- FS
- NIC
- Shell
- GUI Server
- Aivot
- Hoitaja

Organisms
- Service interception

Cells

Tasks
EalánOS — What do we need? What do we have?

- Organisms
  - Service interception

- Cells
  - Genode components

- Tasks
EalánOS — What do we need? What do we have?

Organisms
- Service interception

Cells
- Genode components

Tasks
- Port MxTasking

1 MxTasking: Task-based framework with built-in prefetching and synchronization — github.com/jmuehlig/mxtasking
### MxTasking — What does it need?

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard C++ library</td>
<td>• For internal data structures</td>
</tr>
<tr>
<td></td>
<td>• Portability</td>
</tr>
<tr>
<td>Filesystem</td>
<td>• For benchmarks</td>
</tr>
<tr>
<td></td>
<td>• For writing profiling results</td>
</tr>
<tr>
<td>Timer support</td>
<td>• For garbage collection</td>
</tr>
<tr>
<td></td>
<td>• And profiling</td>
</tr>
<tr>
<td>Multicore support</td>
<td>• Needed for task parallelism</td>
</tr>
<tr>
<td>NUMA support</td>
<td>• For NUMA-aware task scheduling</td>
</tr>
<tr>
<td></td>
<td>• And data placement</td>
</tr>
</tbody>
</table>
MxTasking — What does it need?

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Details</th>
</tr>
</thead>
</table>
| Standard C++ library         | • For internal data structures  
                              | • Portability |
| Filesystem                   | • For benchmarks  
                              | • For writing profiling results |
| Timer support                 | • For garbage collection  
                              | • And profiling |
| Multicore support            | • Needed for task parallelism |
| NUMA support                  | • For NUMA-aware task scheduling  
                              | • And data placement |
Filling the gaps … NUMA support for Genode

- NOVA extension for NUMA
  - for accessing SRAT entries

- Topology service
  - for querying topology from user-space

- Regional heaps
  - for NUMA-aware memory allocation

- MxTasking glue-code
Filling the gaps … NUMA support for Genode

- NOVA extension for NUMA
  - for accessing SRAT entries

- Topology service
  - for querying topology from user-space

- Regional heaps
  - for NUMA-aware memory allocation

- MxTasking glue-code

365 LoC
Filling the gaps … NUMA support for Genode

- NOVA extension for NUMA
  - for accessing SRAT entries
  - 365 LoC

- Topology service
  - for querying topology from user-space
  - 531 LoC

- Regional heaps
  - for NUMA-aware memory allocation

- MxTasking glue-code
Filling the gaps ... NUMA support for Genode

- NOVA extension for NUMA
  - for accessing SRAT entries
  
  365 LoC

- Topology service
  - for querying topology from user-space
  
  531 LoC

- Regional heaps
  - for NUMA-aware memory allocation
  
  683 LoC

- MxTasking glue-code
Filling the gaps ... NUMA support for Genode

- **NOVA extension for NUMA**
  - for accessing SRAT entries
  - 365 LoC

- **Topology service**
  - for querying topology from user-space
  - 531 LoC

- **Regional heaps**
  - for NUMA-aware memory allocation
  - 683 LoC

- **MxTasking glue-code**
  - 186 LoC
What you can win with Genode

**Code size in comparison**

- **from scratch**: ~25000 lines of code
- **EalánOS**: ~1000 lines of code

- **API**
- **HAL**
- **MxTasking**
- **OS services**
- **Support lib**
- **libc support**

**Effort in comparison**

- **from scratch**: ~90% less code
- **EalánOS**: ~90% time saving

<table>
<thead>
<tr>
<th></th>
<th>from scratch</th>
<th>EalánOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>man-months</td>
<td>79.13</td>
<td>6.07</td>
</tr>
</tbody>
</table>

Using Genode as Enabler for Research on Modern Operating Systems / Michael Müller / 70
Automated Experiments with Genode

TO THE LAB!
Sculpting a Genode system

- A **Genode** system **consists** of components

---

**Figure 11:** Session creation at the server.

Source: “**Genode Foundations 22.05**” – Norman Feske, Genode Labs GmbH, Dresden, 2022
A Genode system consists of components

Components are applications, OS servers, drivers etc.
A **Genode** system **consists** of components

- Components are **applications**, **OS servers**, **drivers** etc.
- Running system is a **tree of components**

---

**Sculpting a Genode system**

*Figure 11: Session creation at the server.*

Source: "**Genode Foundations 22.05**" – Norman Feske, Genode Labs GmbH, Dresden, 2022
A Genode system consists of components

Components are applications, OS servers, drivers etc.

Running system is a tree of components

Scenarios specify the tree of components and the relation among its nodes as XML configurations
Sculpting a Genode system

Now, let’s try an example experiment!

Figure 11: Session creation at the server.

Source: “Genode Foundations 22.05” – Norman Feske, Genode Labs GmbH, Dresden, 2022
Example: B-link tree benchmark

- Investigate how **throughput** of benchmark is **affected** when we run **multiple instances**
  - On the **same** set of CPU cores
  - On a **disjunct** set of CPU cores
- **Which** scenario will yield the **higher** throughput at the respective **maximum** of cores?

- B-link trees are a wide-spread **datastructure** for **indexing** in **database** systems.
- The B-link tree benchmark is based on **YCSB**, a **common benchmark** for **key-value stores**.
Example: B-link tree benchmark
Example: B-link tree benchmark

```
<config>
  <start name="timer">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree1">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree2">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree3">
    <binary name="blinktree"/>
  </start>
</config>
```
Example: B-link tree benchmark

```
<config>
  <start name="timer">
    <provides><service name="Timer"/></provides>
    <route>
      <any-service><parent/><any-child/></any-service>
    </route>
  </start>
  <start name="blinktree1">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree2">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree3">
    <binary name="blinktree"/>
  </start>
</config>
```
Example: B-link tree benchmark

```
<config>
  <start name="timer">
    <provides><service name="Timer"/></provides>
    <route>
      <any-service><parent/><any-child/></any-service>
    </route>
  </start>
  <start name="blinktree1">
    <binary name="blinktree"/>
    <route>
      <service name="Timer"><child name="timer"/></service>
      <any-service><parent/><any-child/></any-service>
    </route>
  </start>
  <start name="blinktree2">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree3">
    <binary name="blinktree"/>
  </start>
</config>
```
Example: B-link tree benchmark

- Affinity spaces map components to a set of CPU cores
- And they can be nested

*Source: “Genode Foundations 22.05” – Norman Feske, Genode Labs GmbH, Dresden, 2022*
Example: B-link tree benchmark

```xml
<config>
  <affinity-space width="64" height="1"/>
  <start name="timer">
    <provides><service name="Timer"></provides>
    <route>
      <any-service><parent/></any-service>
    </route>
  </start>
  <start name="blinktree1">
    <binary name="blinktree"/>
    <route>
      <service name="Timer"><child name="timer"></child>
      <any-service><parent/></any-service>
    </route>
  </start>
  <start name="blinktree2">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree3">
    <binary name="blinktree"/>
  </start>
</config>
```
Example: B-link tree benchmark

```xml
<config>
  <affinity-space width="64" height="1"/>
  <start name="timer">
    <provides><service name="Timer"/></provides>
    <route>
      <any-service><parent/>
      <any-service><parent/> <child service="Timer"/>
      <any-service><parent/>
    </route>
  </start>
  <start name="blinktree1">
    <affinity xpos="1" ypos="0" width="63" height="1"/>
    <binary name="blinktree"/>
    <route>
      <service name="Timer"><child name="timer"/></service>
      <any-service><parent/></any-service>
    </route>
  </start>
  <start name="blinktree2">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree3">
    <binary name="blinktree"/>
  </start>
</config>
```
Example: B-link tree benchmark

```
<config>
  <affinity-space width="64" height="1"/>
  <start name="timer">
    <provides><service name="Timer"></provides>
    <route>
      <any-service><parent/><any-child/></any-service>
    </route>
  </start>
  <start name="blinktree1">
    <affinity xpos="1" ypos="0" width="63" height="1"/>
    <resource name="RAM" quantum="80G"/>
    <binary name="blinktree"/>
    <route>
      <service name="Timer"><child name="timer"></child></service>
      <any-service><parent/><any-child/></any-service>
    </route>
  </start>
  <start name="blinktree2">
    <binary name="blinktree"/>
  </start>
  <start name="blinktree3">
    <binary name="blinktree"/>
  </start>
</config>
```
Example: B-link tree benchmark

Repeat for blinktree2 and blinktree3
Example: Results

- **B-link tree: Inserts only**
  - 3 cells time-shared
  - 3 cells space-partitioned

- **B-link tree: Lookups only**
  - 3 cells time-shared
  - 3 cells space-partitioned
Conclusion

- Hardware has changed **tremendously**
- More OS research needed, but **high** entrance hurdle
- An OS framework can **lower** this barrier
- Genode can save up to **92% of development time**
- EalánOS **contributes to** Genode by offering
  - State-of-the art **task-parallel** programming
  - **NUMA** support
  - Support for **many-core** systems
- With a **focus** on research and the **datacenter**
The road to the future

- Profiling
- Elasticity of cells
- Management strategies
- Evaluation in realistic scenario
- First full-featured prototype of EalánOS