OSNABRÜCK UNIVERSITY

Using Genode as Enabler for Research on Modern Operating Systems

Bruxelles, February 5, 2023

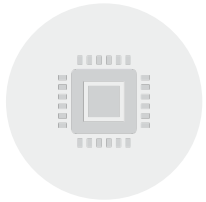
Michael Müller

About me

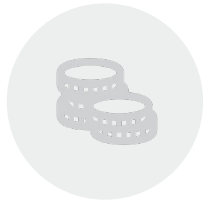
- Studied computer science at TU Dortmund
- Graduated with MSc in 2018
- Since 2018 **PhD student** at ESS Group **at Osnabrück University**
- **Full-time research assistant** in the **MxKernel** project **at** TU Dortmund and **Osnabrück University**
- Focused on **research** for **many-core OSes**

Assumptions from the past … that shaped today's OSes

There is only a single CPU.



Only the CPU computes



Context-switches are cheap

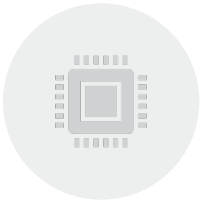
Main memory is scarce

The memory architecture is uniform

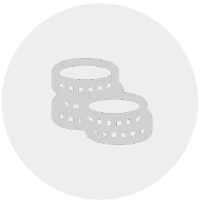
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There is only a **single** CPU.



Only the CPU computes



Context-switches are cheap

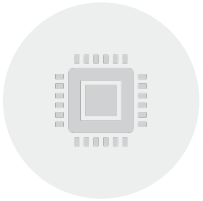
Main memory is scarce

The memory architecture is uniform

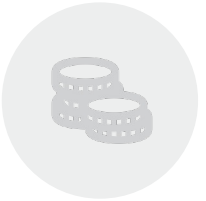
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



Only the CPU computes



Context-switches are cheap

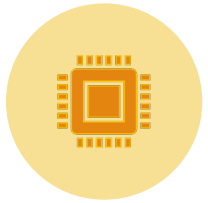
Main memory is scarce

The memory architecture is uniform

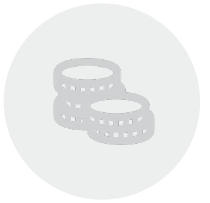
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



Only the CPU computes



Context-switches are cheap

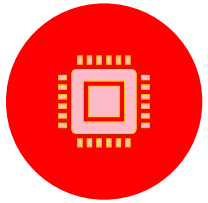
Main memory is scarce

The memory architecture is uniform

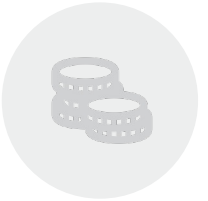
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are cheap

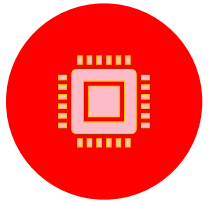
Main memory is scarce

The memory architecture is uniform

I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes

Context-switches are cheap

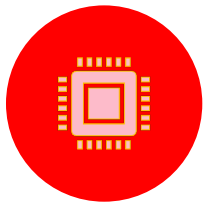
Main memory is scarce

The memory architecture is uniform

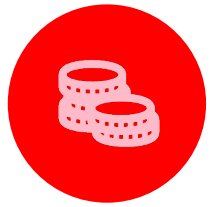
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are **not** cheap **anymore**

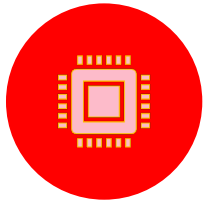
Main memory is scarce

The memory architecture is uniform

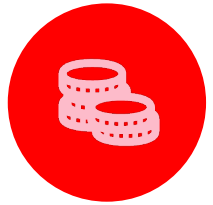
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are **not** cheap **anymore**

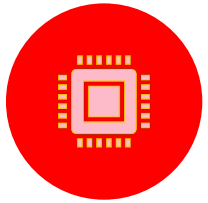
Main memory is scarce

The memory architecture is uniform

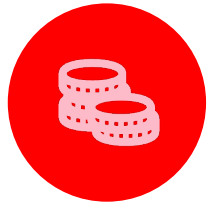
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are **not** cheap **anymore**

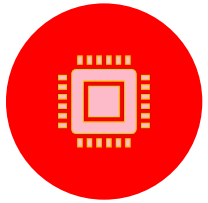
Main memory is ~~scarce~~ **abundant**

The memory architecture is uniform

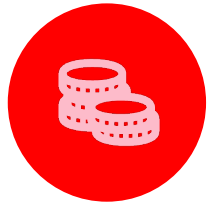
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are **not** cheap **anymore**

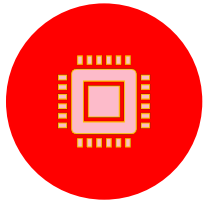
Main memory is ~~scarce~~ **abundant**

The memory architecture is uniform

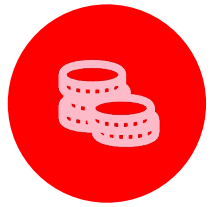
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are **not** cheap **anymore**

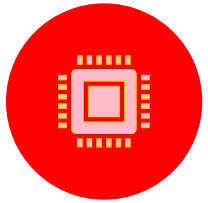
Main memory is ~~scarce~~ **abundant**

The memory architecture is ~~uniform~~ **heterogeneous**

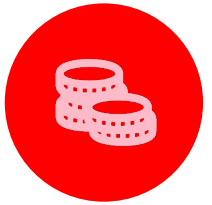
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



Context-switches are **not** cheap **anymore**

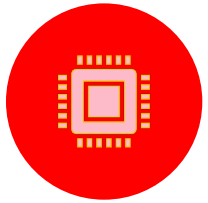
Main memory is ~~scarce~~ **abundant**

The memory architecture is ~~uniform~~ **heterogeneous**

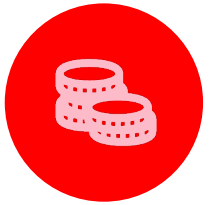
I/O is slower than the CPU

Assumptions from the past … that shaped today's OSes

There **are** ~~is only a single~~ **many** CPUs.



~~Only~~ **Not just** the CPU computes



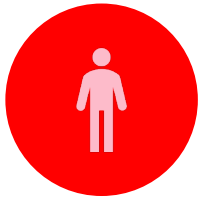
Context-switches are **not** cheap **anymore**

Main memory is ~~scarce~~ **abundant**

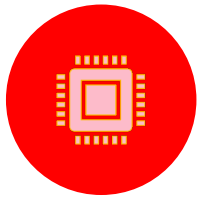
The memory architecture is ~~uniform~~ **heterogeneous**

I/O is ~~slower than~~ **as fast as** the CPU

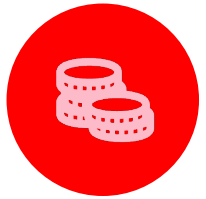
The truth about modern computers



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

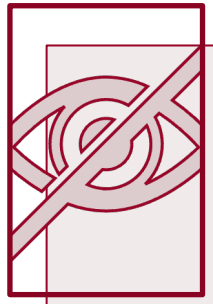
Non-free licensing

- prevents full understanding
- modified system not publishable

Hurdles on the way … what puts OS research at risk

Non-free licensing

- prevents full understanding
- modified system not publishable



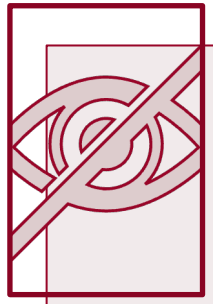
Hardware black boxes

- 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

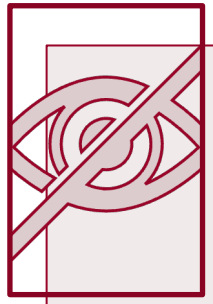
NDAs

- severely restrict publications
- may suppress unfavoured results

Hurdles on the way … what puts OS research at risk

Non-free licensing

- prevents full understanding
- modified system not publishable

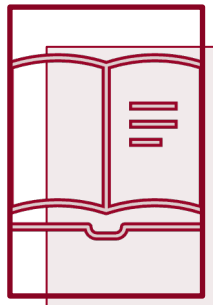


Hardware black boxes

- hinder implementing drivers
- hamper in-depth evaluation

NDAs

- severely restrict publications
- may suppress unfavoured results



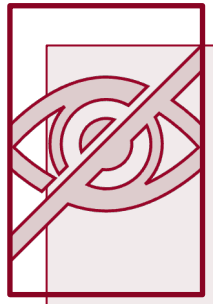
Missing documentation

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

Hurdles on the way … what puts OS research at risk

Non-free licensing

- prevents full understanding
- modified system not publishable

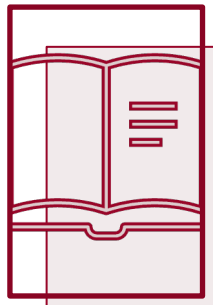


Hardware black boxes

- hinder implementing drivers
- hamper in-depth evaluation

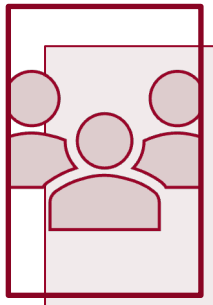
NDAs

- severely restrict publications
- may suppress unfavoured results



Missing documentation

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



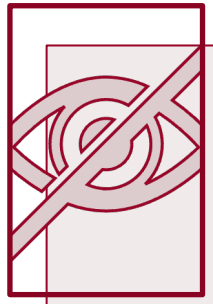
Lack of man-power

- 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

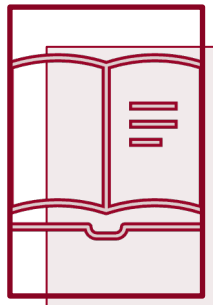


Hardware black boxes

- hinder implementing drivers
- hamper in-depth evaluation

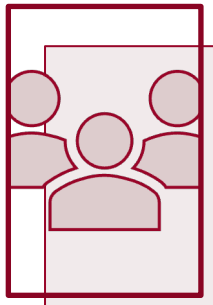
NDAs

- 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

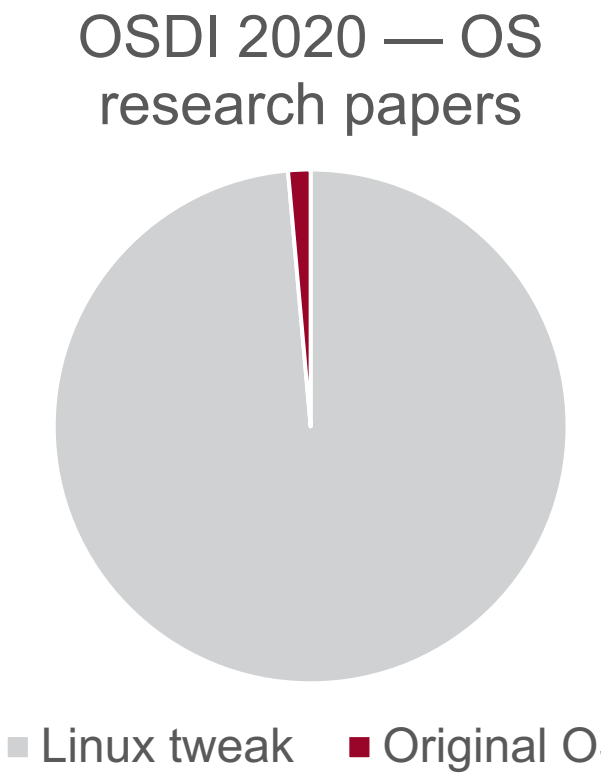
SO, WHAT DO RESEARCHERS DO?

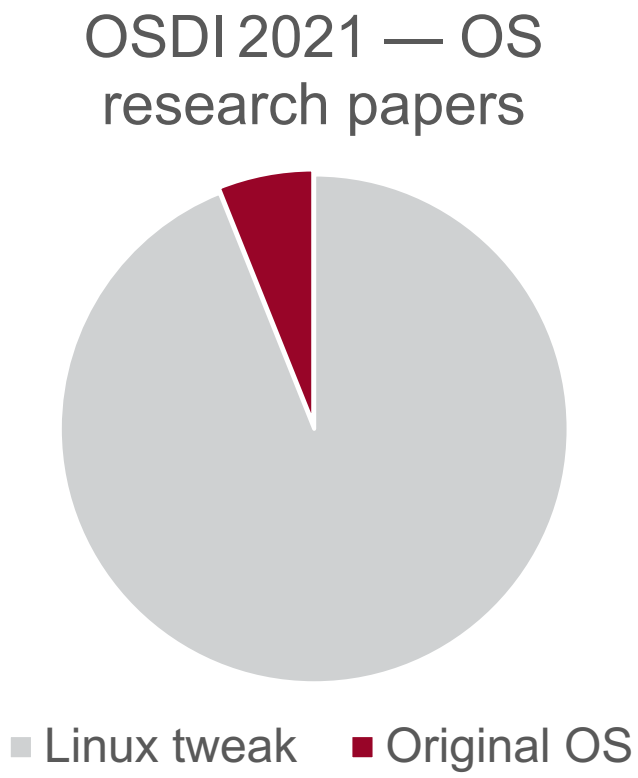
Workarounds and Tweaks

[1] Firestone, D. et al. 2018. Azure Accelerated Networking: SmartNICs in the Public Cloud. *15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018, Renton, WA, USA, April 9-11, 2018* (2018), 51–66.
[2] Fried, J., Ruan, Z., Ousterhout, A. and Belay, A. 2020. Caladan: Mitigating Interference at Microsecond Timescales. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 281–297.
[3] Høiland-Jørgensen, T., Brouer, J.D., Borkmann, D., Fastabend, J., Herbert, T., Ahern, D. and Miller, D. 2018. The eXpress data path: fast programmable packet processing in the operating system kernel. *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies* (New York, NY, USA, Dec. 2018), 54–66.
[4] Hwang, J., Vuppalapati, M., Peter, S. and Agarwal, R. 2021. Rearchitecting Linux Storage Stack for µs Latency and High Throughput. (2021), 113–128.
[5] Kim, H.-J., Lee, Y.-S. and Kim, J.-S. 2016. NVMeDirect: A User-space I/O Framework for Application-specific Optimization on NVMe SSDs. *8th USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2016, Denver, CO, USA, June 20-21, 2016.* (2016).
[6] Koo, J., Im, J., Song, J., Park, J., Lee, E., Kim, B.S. and Lee, S. 2021. Modernizing File System through In-Storage Indexing. *15th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2021, July 14-16, 2021.* (2021), 75–92.
[7] Kwon, D., Boo, J., Kim, D. and Kim, J. 2020. FVM: FPGA-assisted Virtual Device Emulation for Fast, Scalable, and Flexible Storage Virtualization. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 955–971.
[8] Li, J., Sharma, N.Kr., Ports, D.R.K. and Gribble, S.D. 2014. Tales of the Tail: Hardware, OS, and Application-level Sources of Tail Latency. *Proceedings of the ACM Symposium on Cloud Computing* (New York, NY, USA, Nov. 2014), 1–14.
[9] Lin, J., Patel, K., Stephens, B.E., Sivaraman, A. and Akella, A. 2020. PANIC: A High-Performance Programmable NIC for Multi-tenant Networks. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 243–259.
[10] Mogul, J.C. 2003. TCP offload is a dumb idea whose time has come. *Proceedings of the 9th conference on Hot Topics in Operating Systems - Volume 9* (USA, May 2003), 5.
[11] Pirelli, S. and Candea, G. 2020. A Simpler and Faster NIC Driver Model for Network Functions. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 225–241.
[12] Ren, Y., Min, C. and Kannan, S. 2020. CrossFS: A Cross-layered Direct-Access File System. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 137–154.
[13] Rizzo, L. 2012. netmap: A Novel Framework for Fast Packet I/O. *2012 USENIX Annual Technical Conference, Boston, MA, USA, June 13-15, 2012* (2012), 101–112.
[14] Rommel, F., Dietrich, C., Friesel, D., Köppen, M., Borchert, C., Müller, M., Spinczyk, O. and Lohmann, D. 2020. From Global to Local Quiescence: Wait-Free Code Patching of Multi-Threaded Processes. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 651–666.
[15] Ruan, Z., Schwarzkopf, M., Aguilera, M.K. and Belay, A. 2020. AIFM: High-Performance, Application-Integrated Far Memory. *14th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2020, Virtual Event, November 4-6, 2020.* (2020), 315–332.

… wait, there are even more !

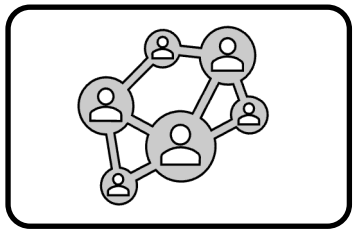
How OS research is done today … mostly





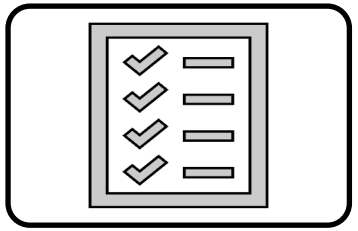
*Source: "**It's Time for Operating Systems to Rediscover Hardware**" -- Timothy Roscoe, USENIX ATC '21/OSDI '21 Joint Keynote*

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?

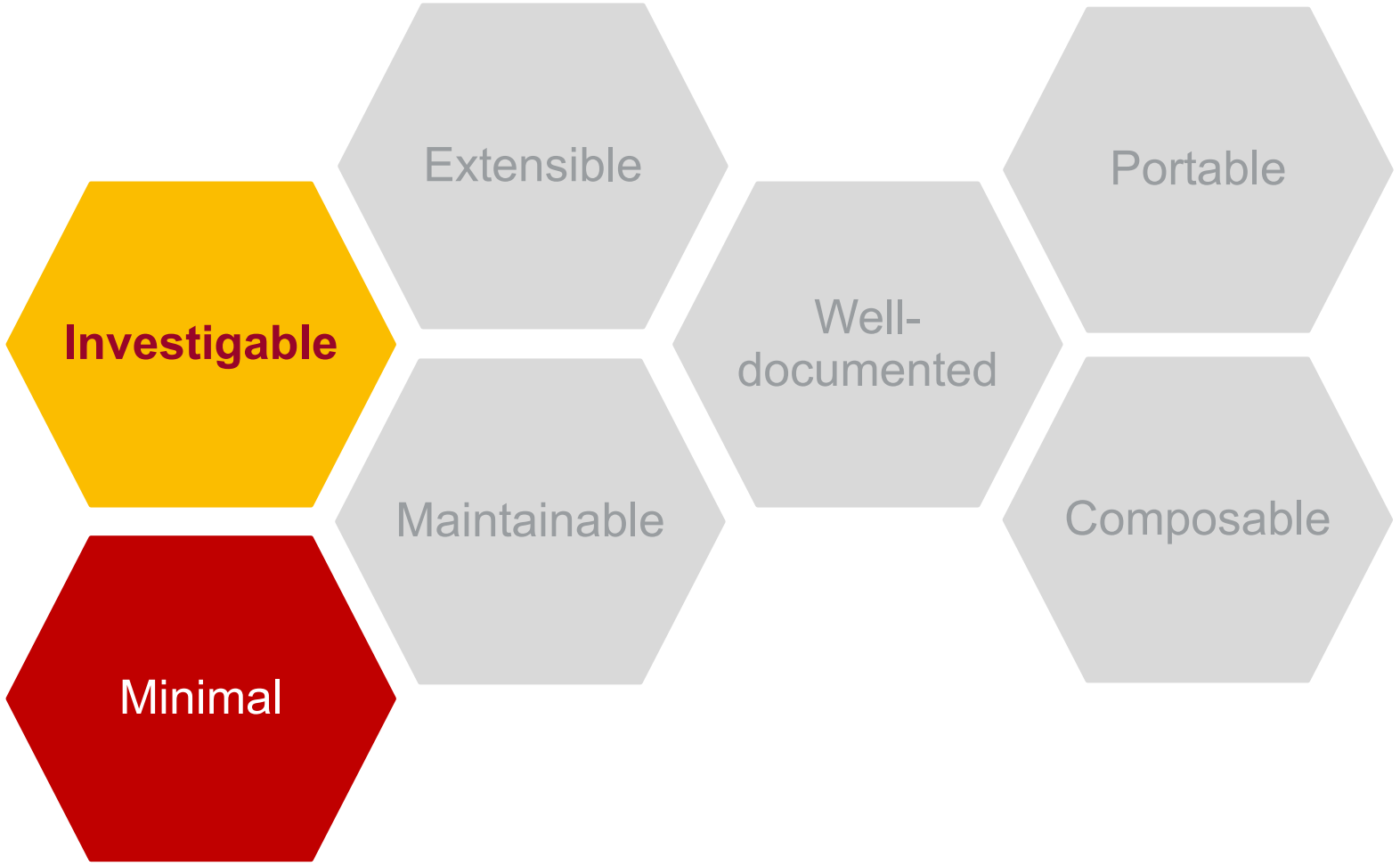
A framework for OS research shall be

A framework for OS research shall be

- Eases **understanding**
- Simplifies **changing** kernel primitives
- Helps **debugging**

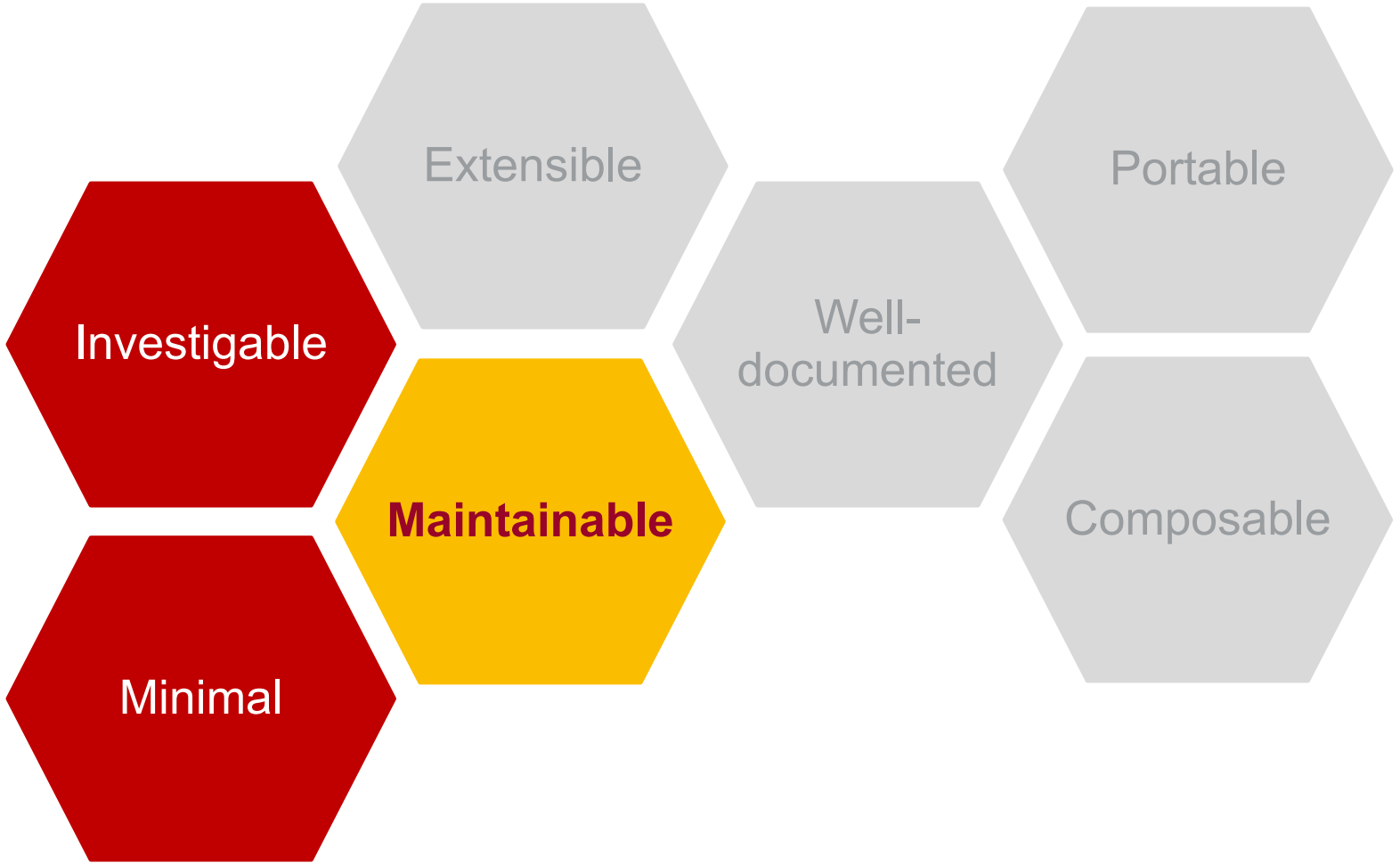
A framework for OS research shall be

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



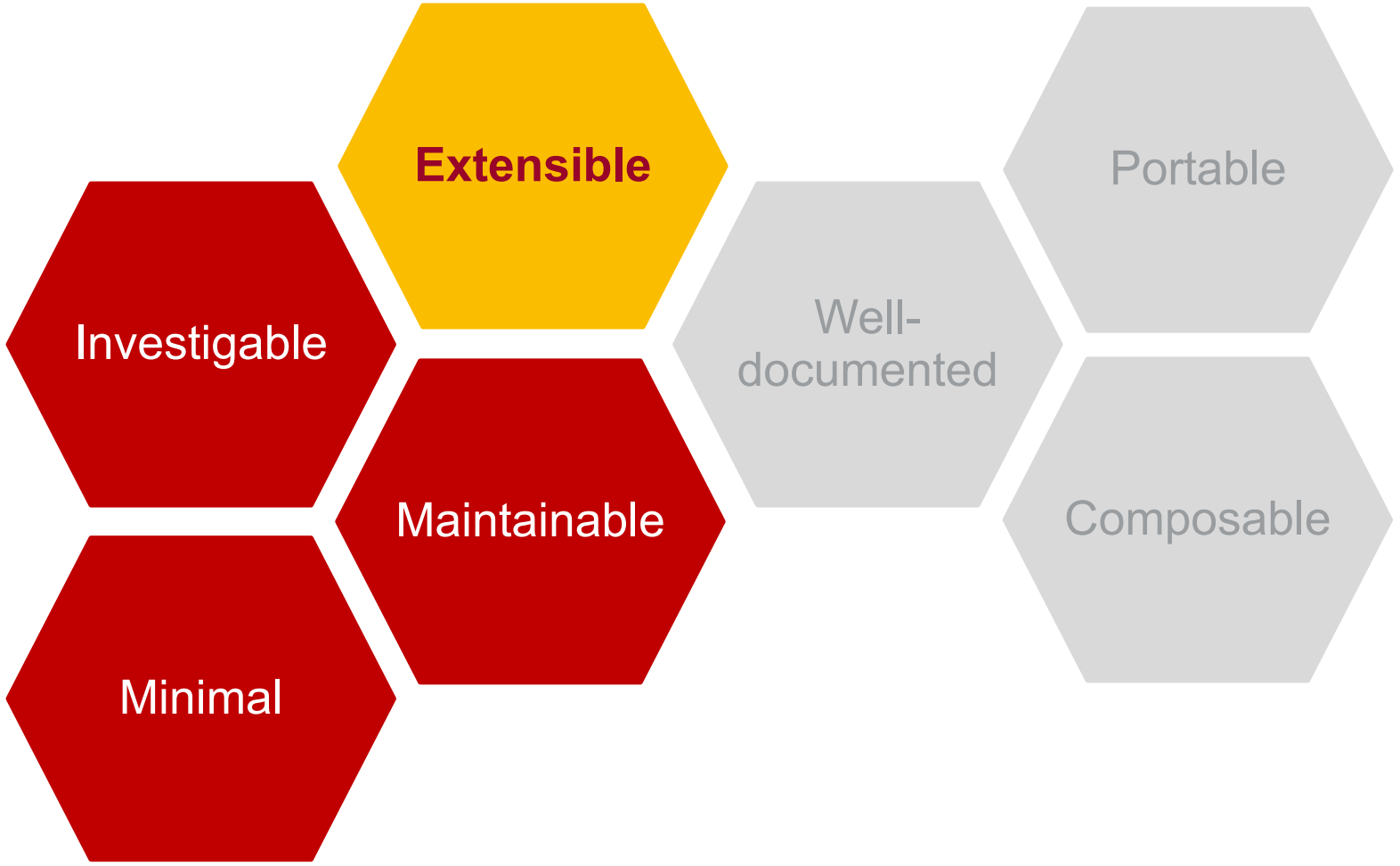
A framework for OS research shall be

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

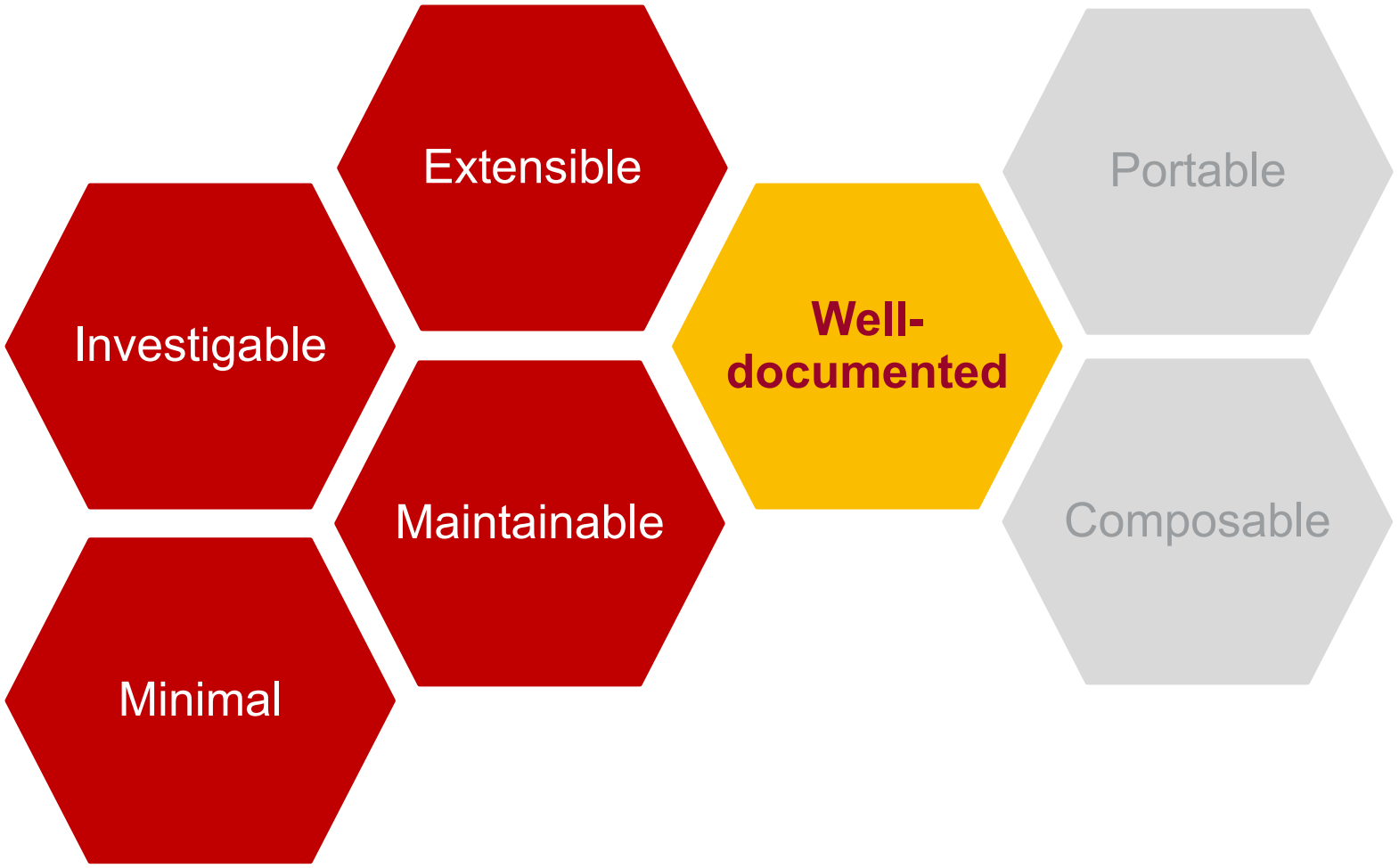


A framework for OS research shall be

- **Easy** to **write** new extensions
- **Separation** of concerns
- Well-defined **components**
- Well-defined **interface** between components

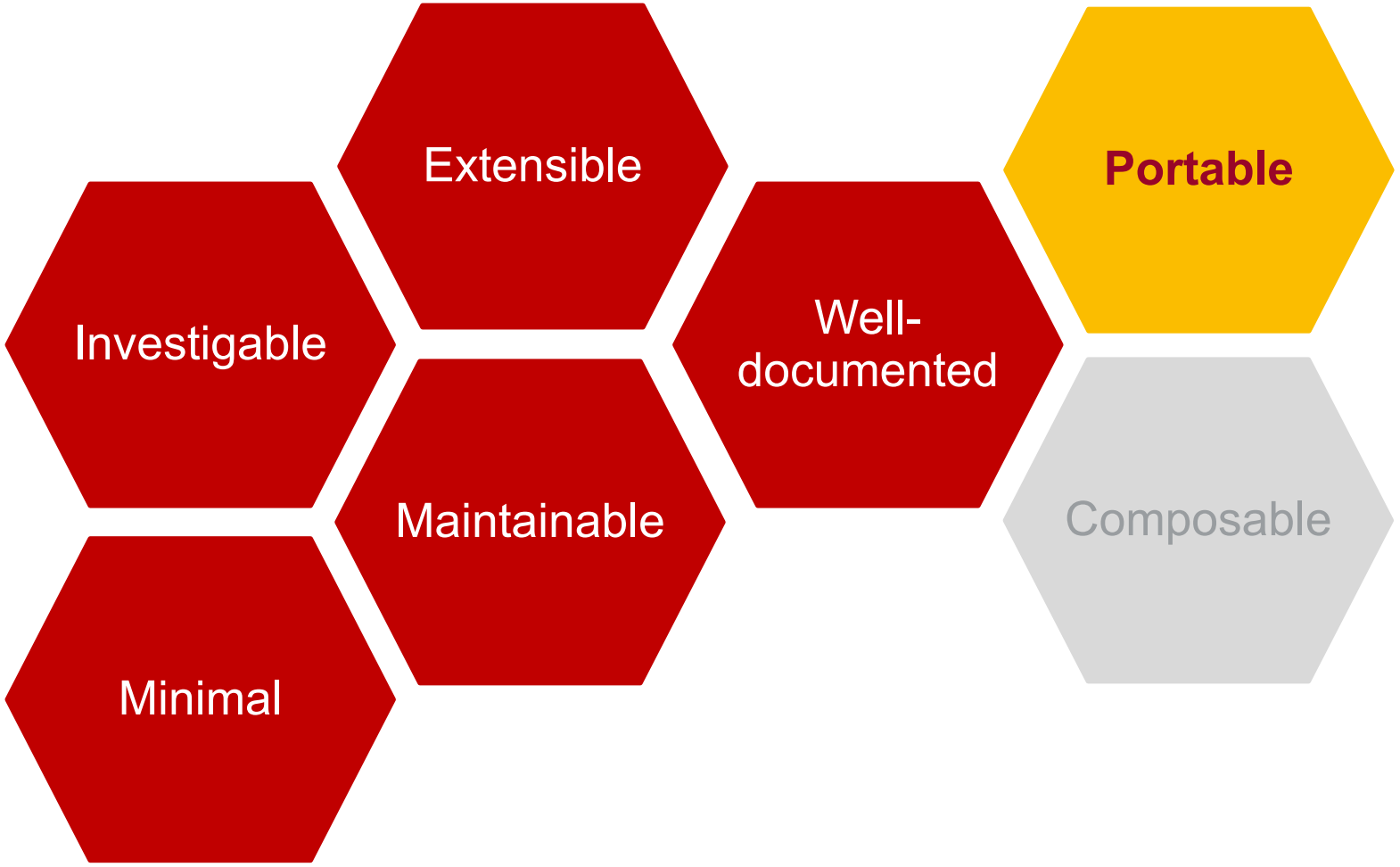


A framework for OS research shall be



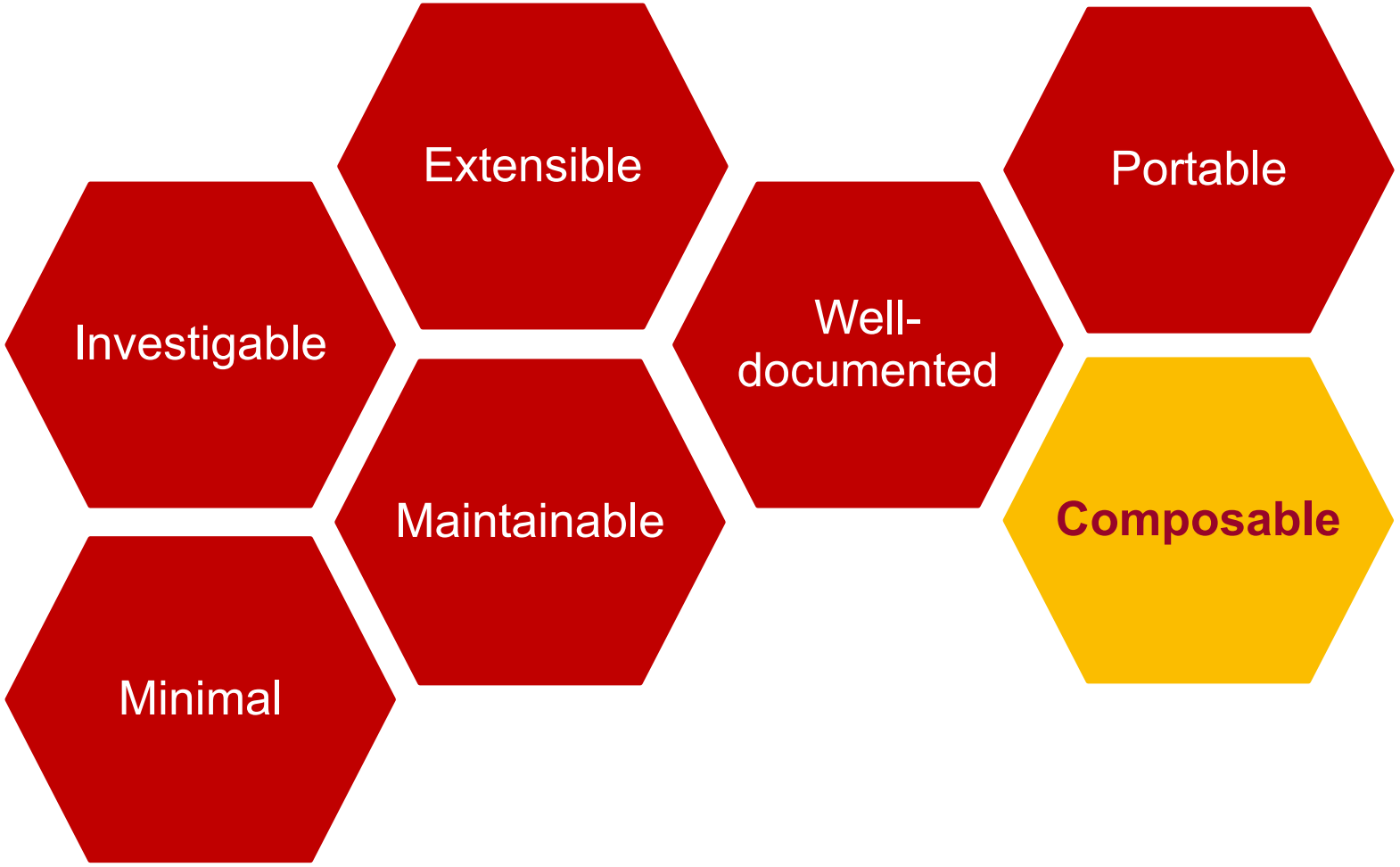
- **Ease** getting started
- **Provides** description of **important** components and **interfaces**
- **Ideally** a book
- **And** documented **code**

A framework for OS research shall be



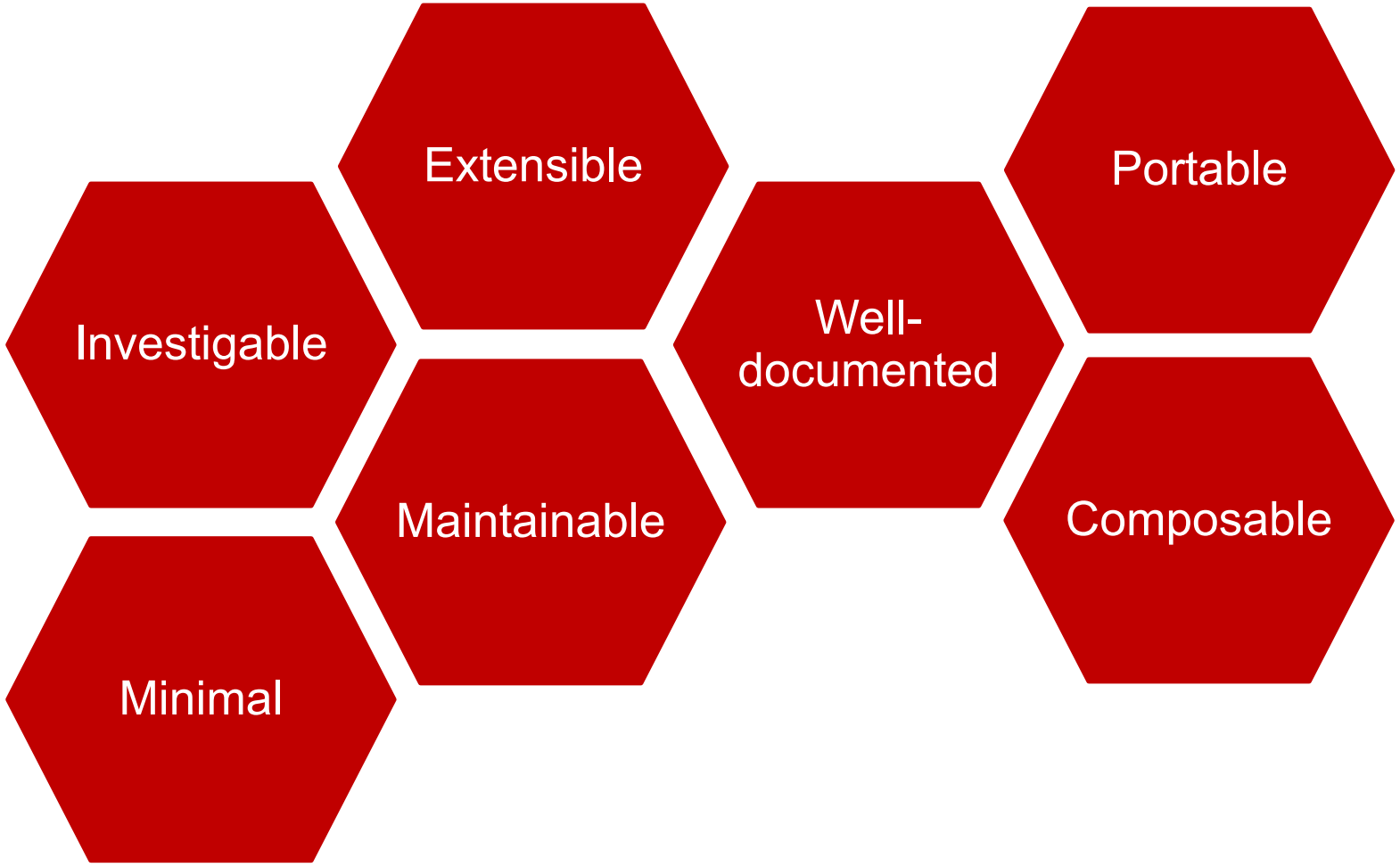
- **Future-proof**
- Enable **experimental** hardware
- **Support** Hardware/OS co-design

A framework for OS research shall be



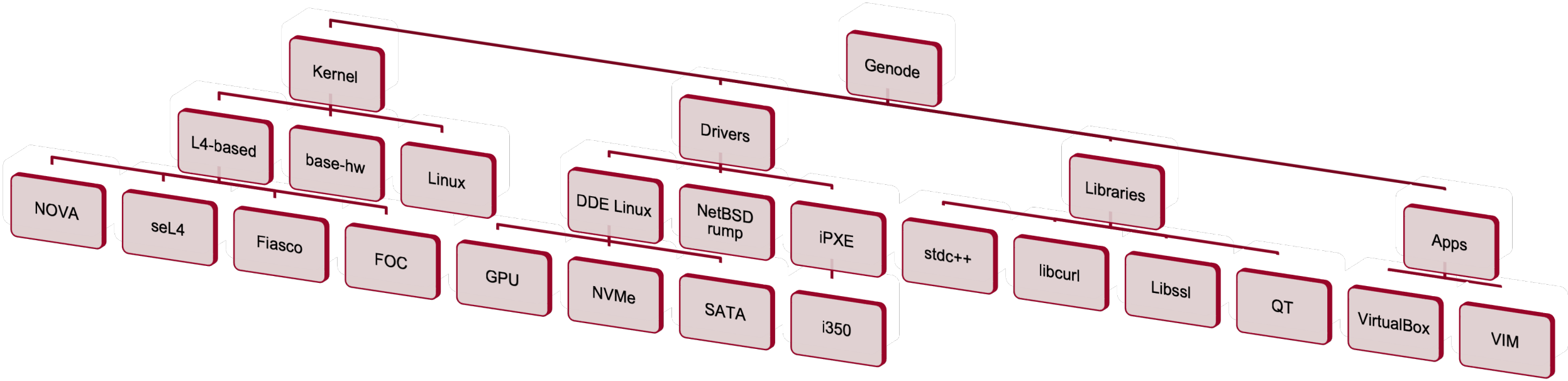
- **Replaceable** OS components
- Allow **different** OS interfaces **simultaneously**
- **Reusability** of drivers etc.

A framework for OS research shall be



A CANDIDATE FOR AN OS FRAMEWORK

The Genode OS Framework

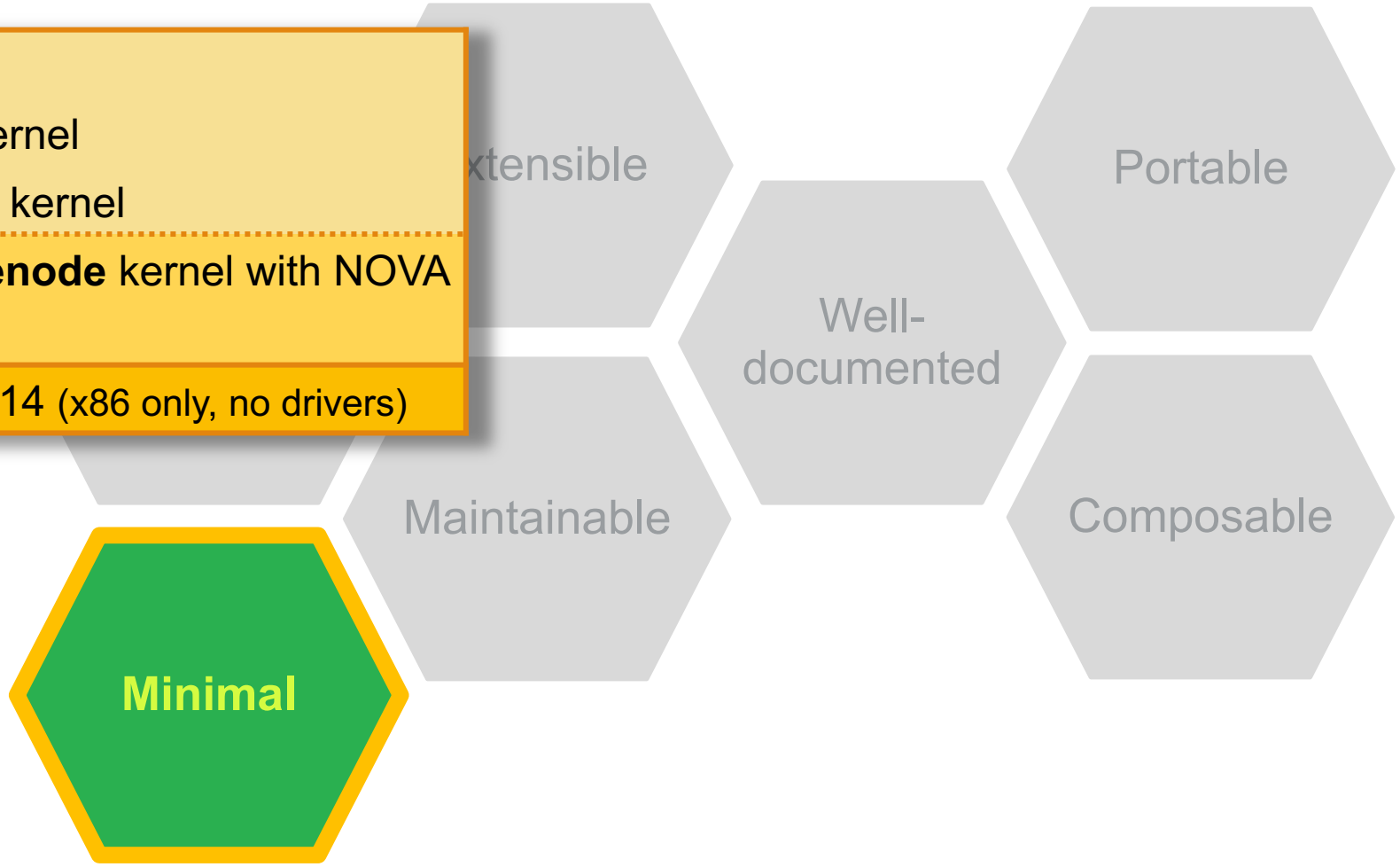


Does Genode fit the bill?

- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

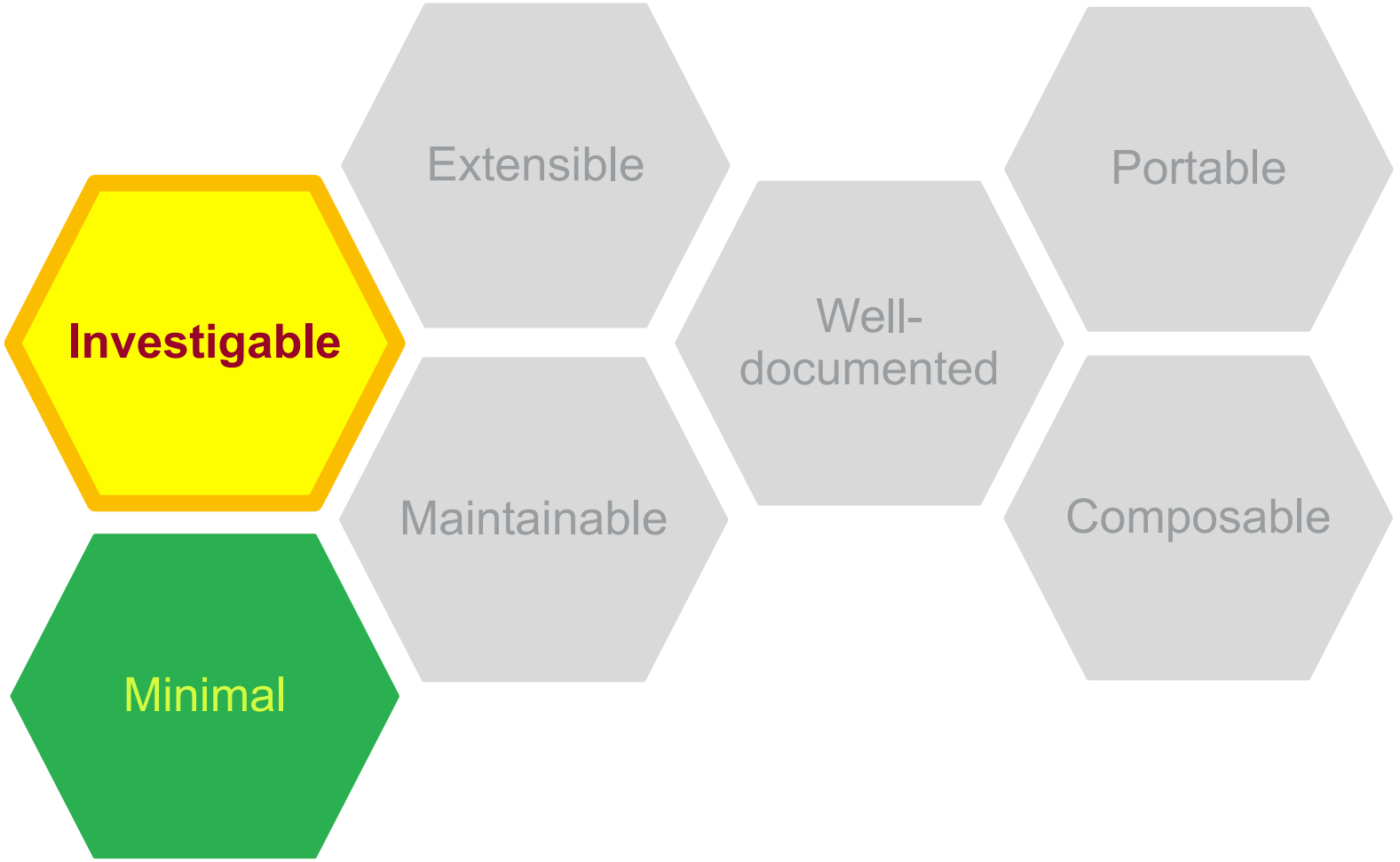
31,918 LoC	for base
21,039 LoC	for NOVA kernel
18,211 LoC	for base-hw kernel
52,957 LoC	for basic **Genode** kernel with NOVA
50,129 LoC	base-hw
911,059 LoC	for **Linux** 4.14 (x86 only, no drivers)



- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

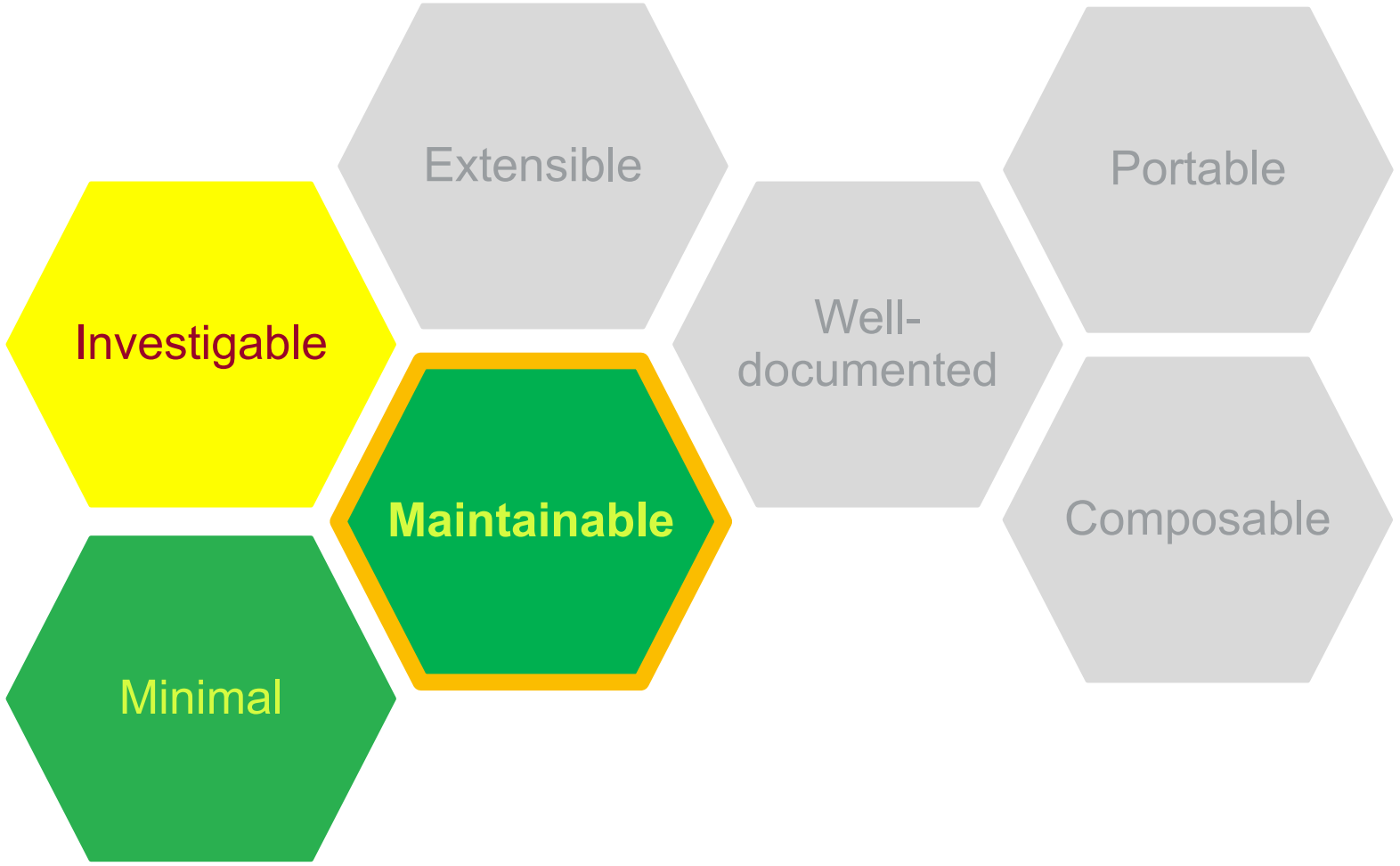
- Code Under **AGPL**
- Only **basic** tracing yet
- Only rudimentary **manual** profiling



- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

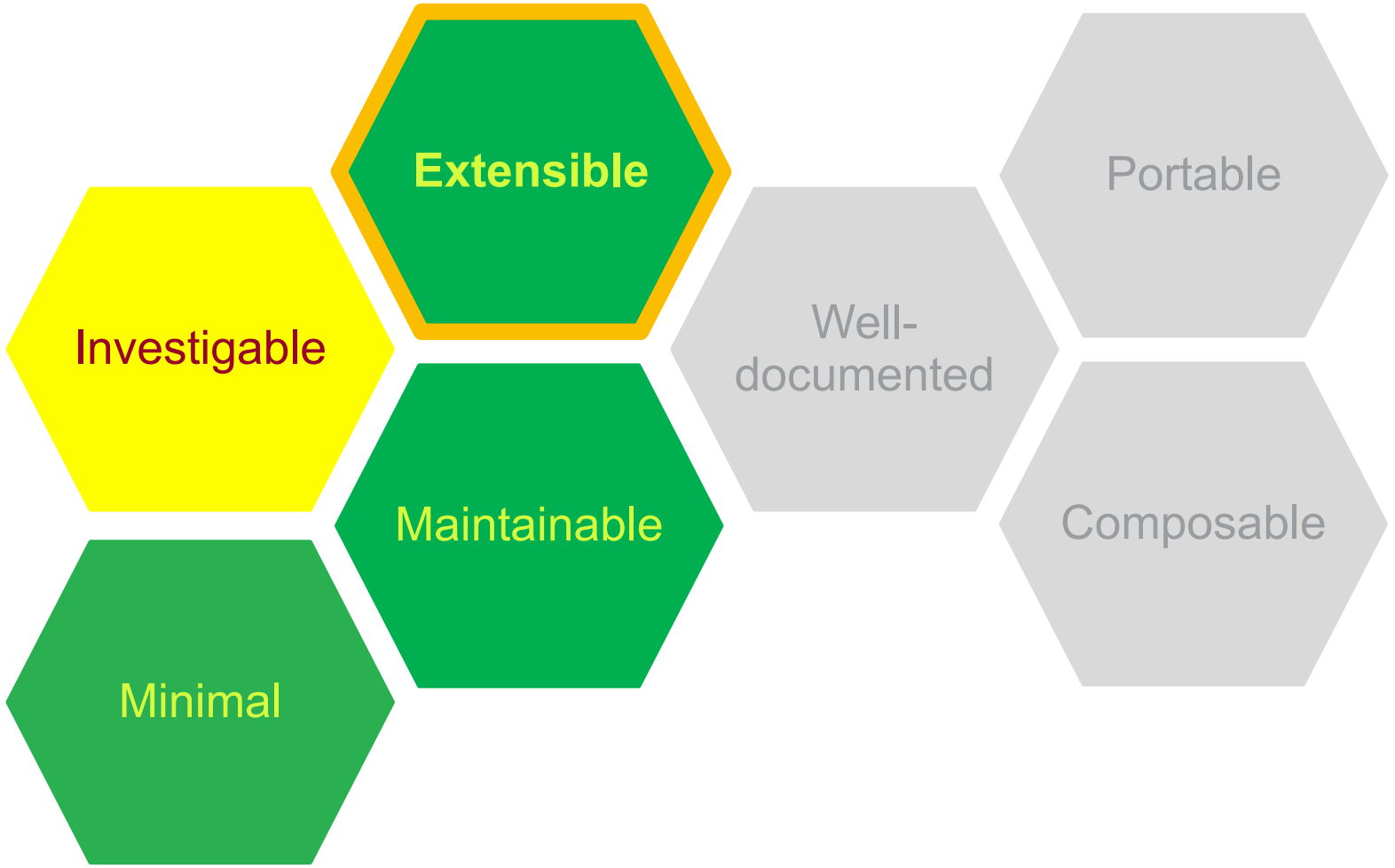
- **Quarterly** updates
- Mostly **minimal** changes to kernel API



- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

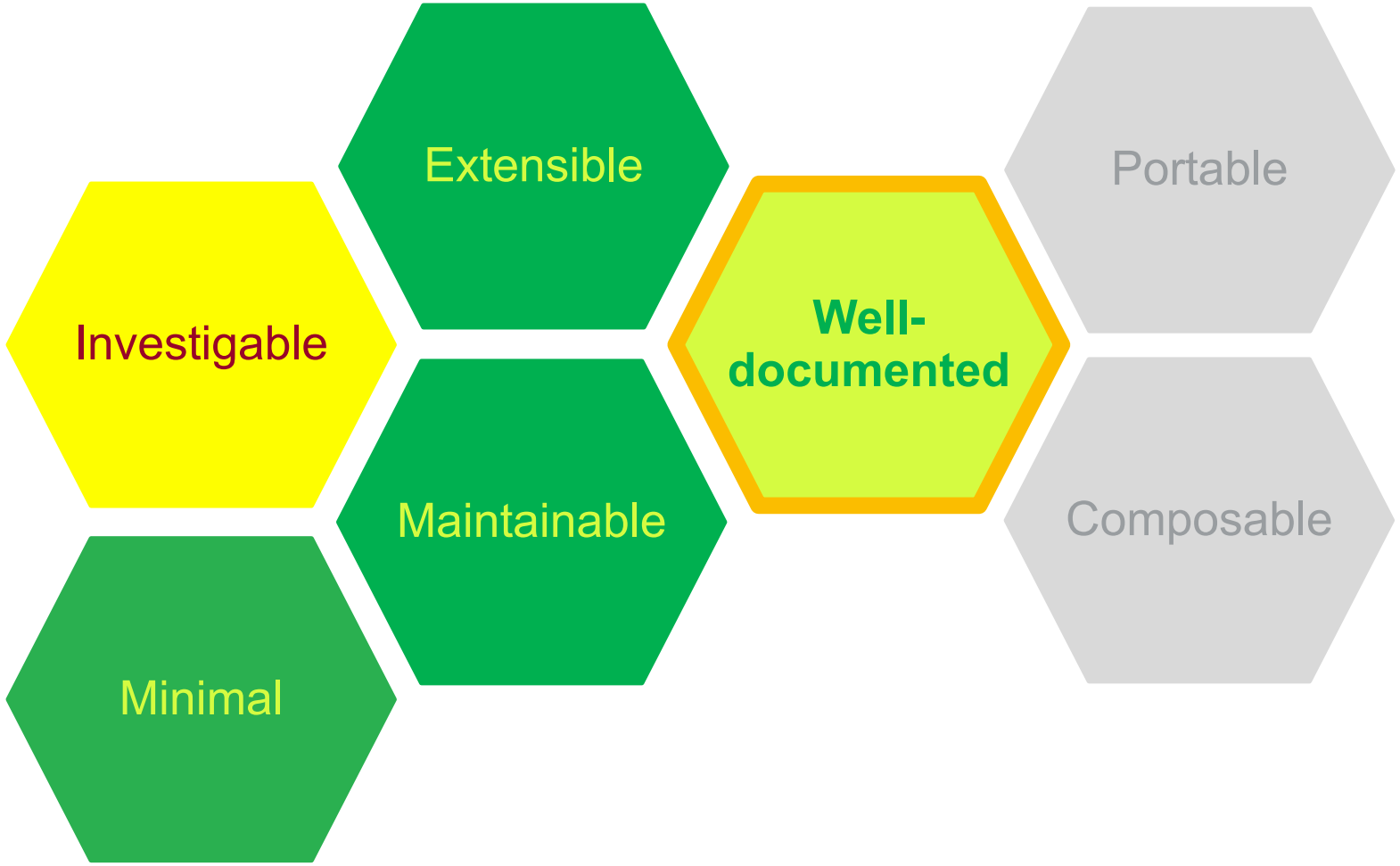
- **Clearly seperated** components
- **Well-defined** RPC interface
- **Minimal** requirements for new components



- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

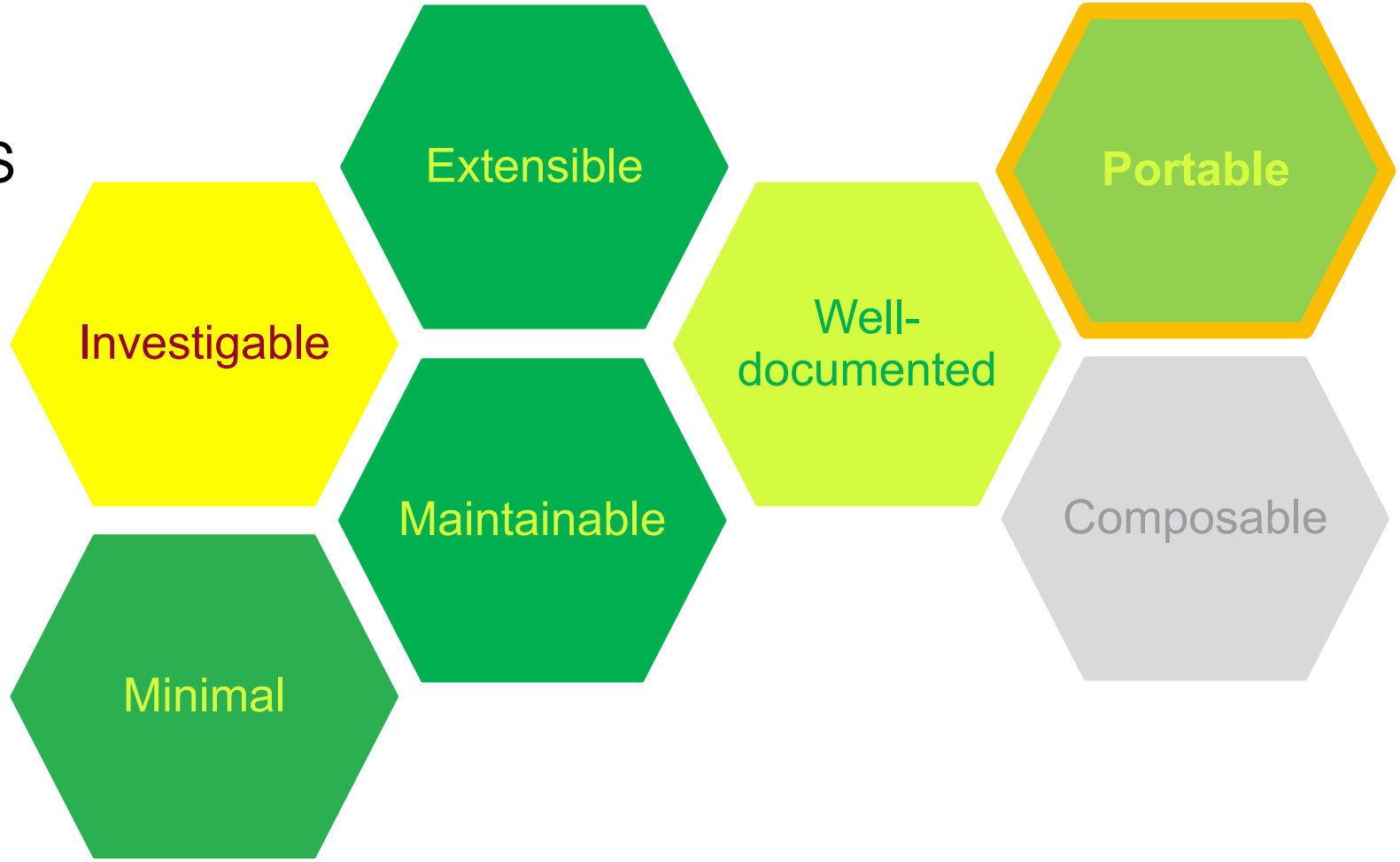
- **Book** “Genode Foundations”
- Extensive **changelog** for each release
- **FOSDEM** talks



- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

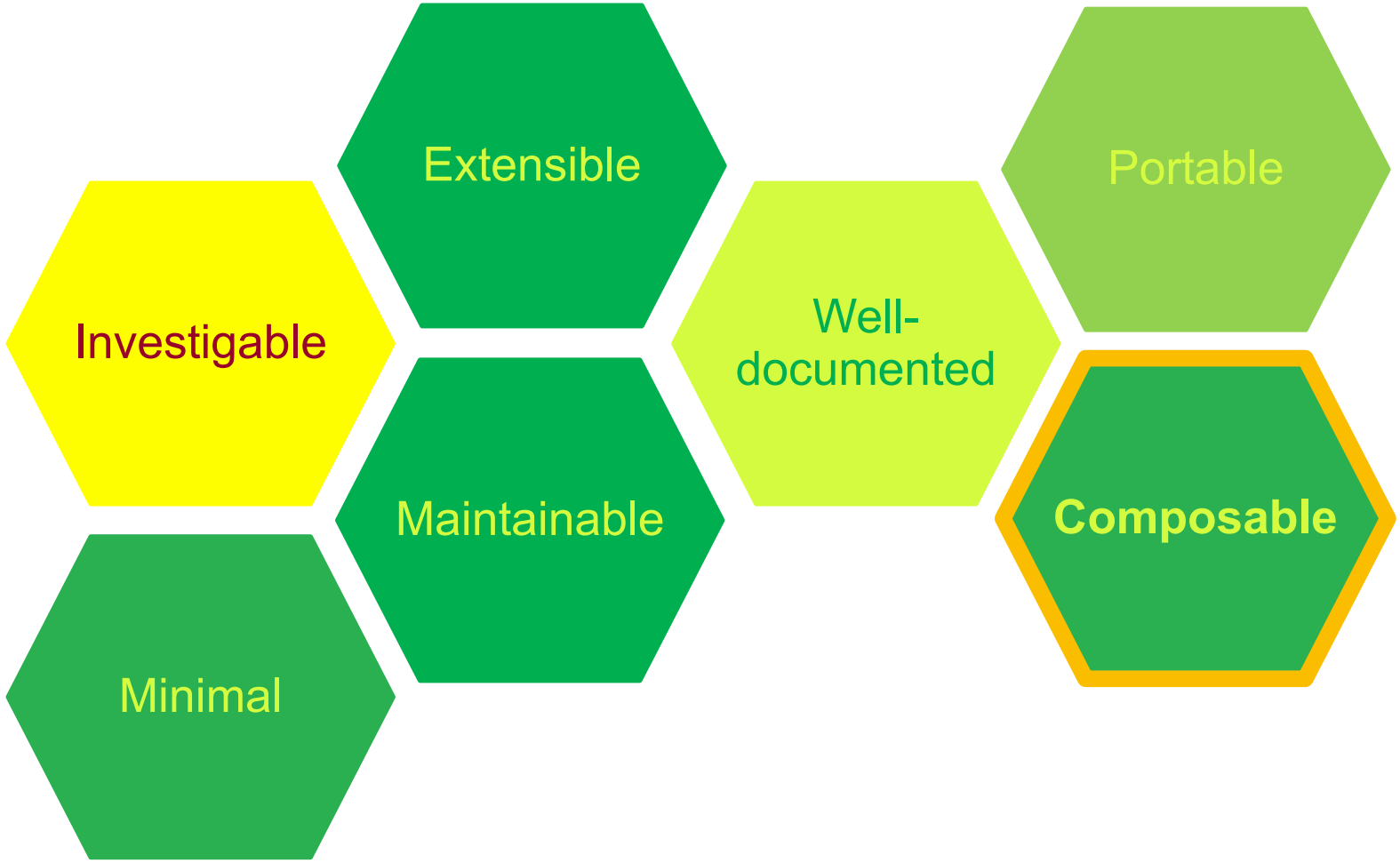
- **Strict seperation** of hardware platform and OS code
- **Generalized** concepts for hardware features



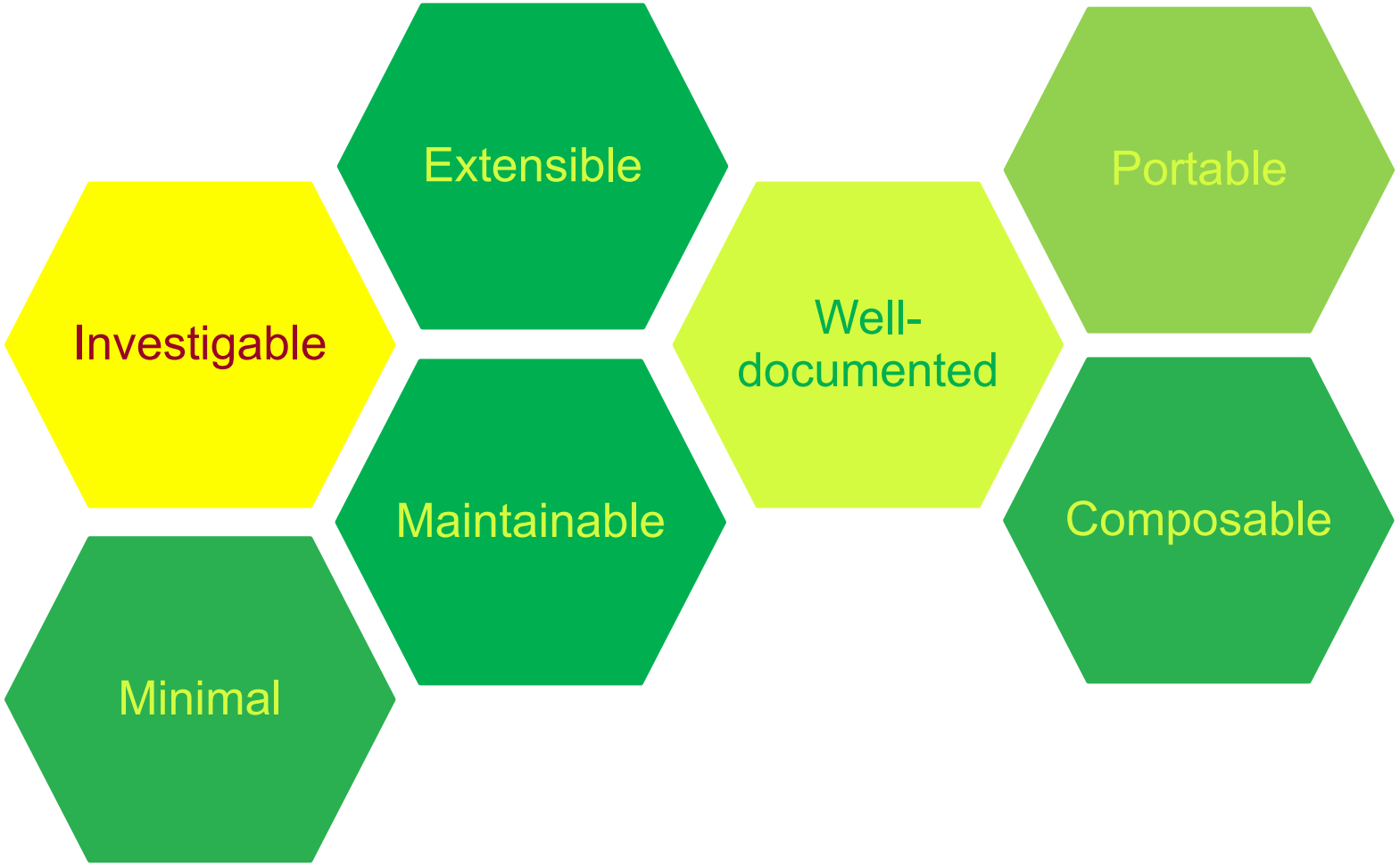
- complete
- almost complete
- satisfactory
- partial
- not yet
- not at all

Does Genode fit the bill?

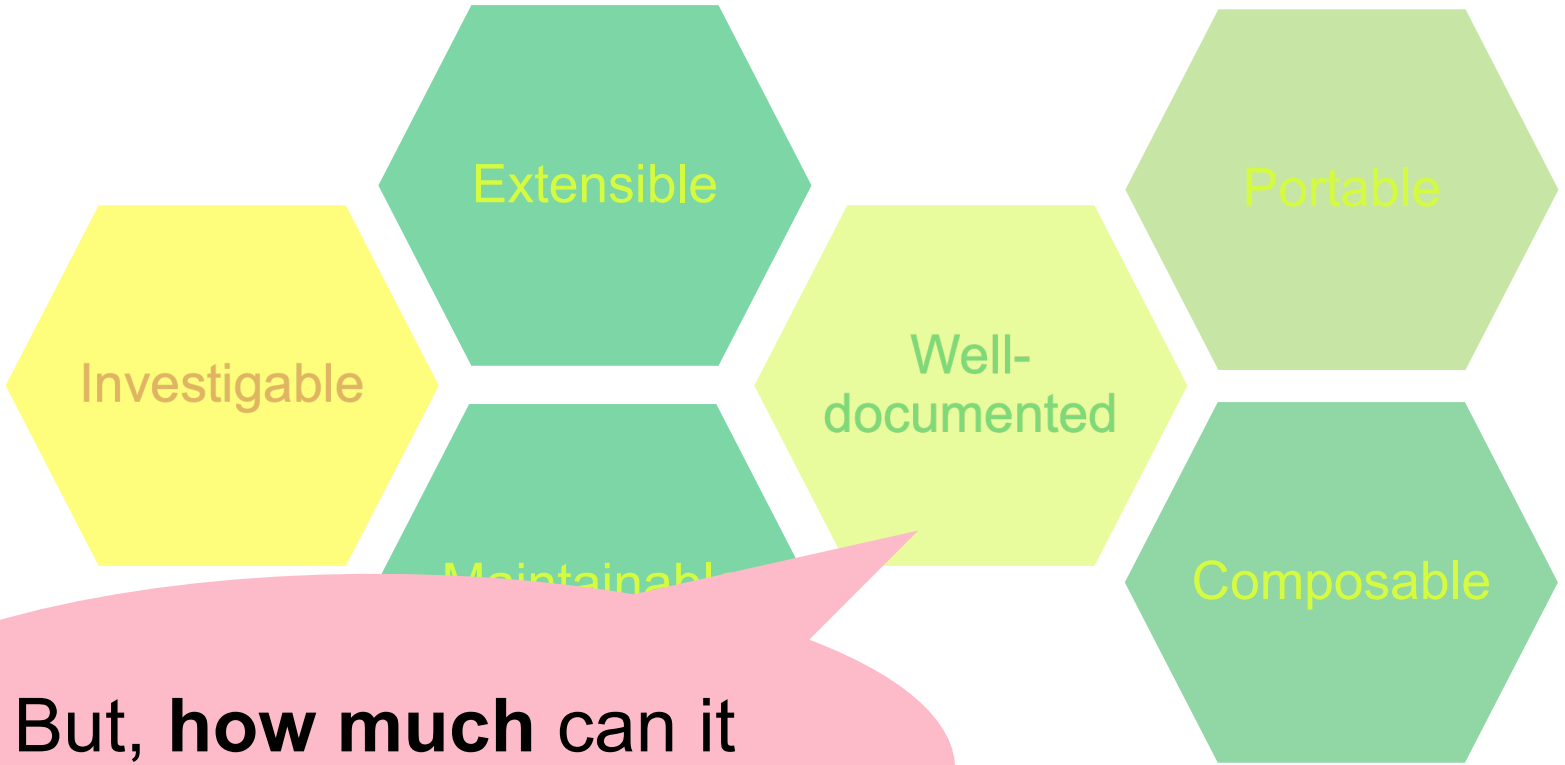
- **Component-based** architecture
- Own OS can be “**sculptured**”
- **Multiple** instances of a service possible



Does Genode fit the bill?



Does Genode fit the bill?

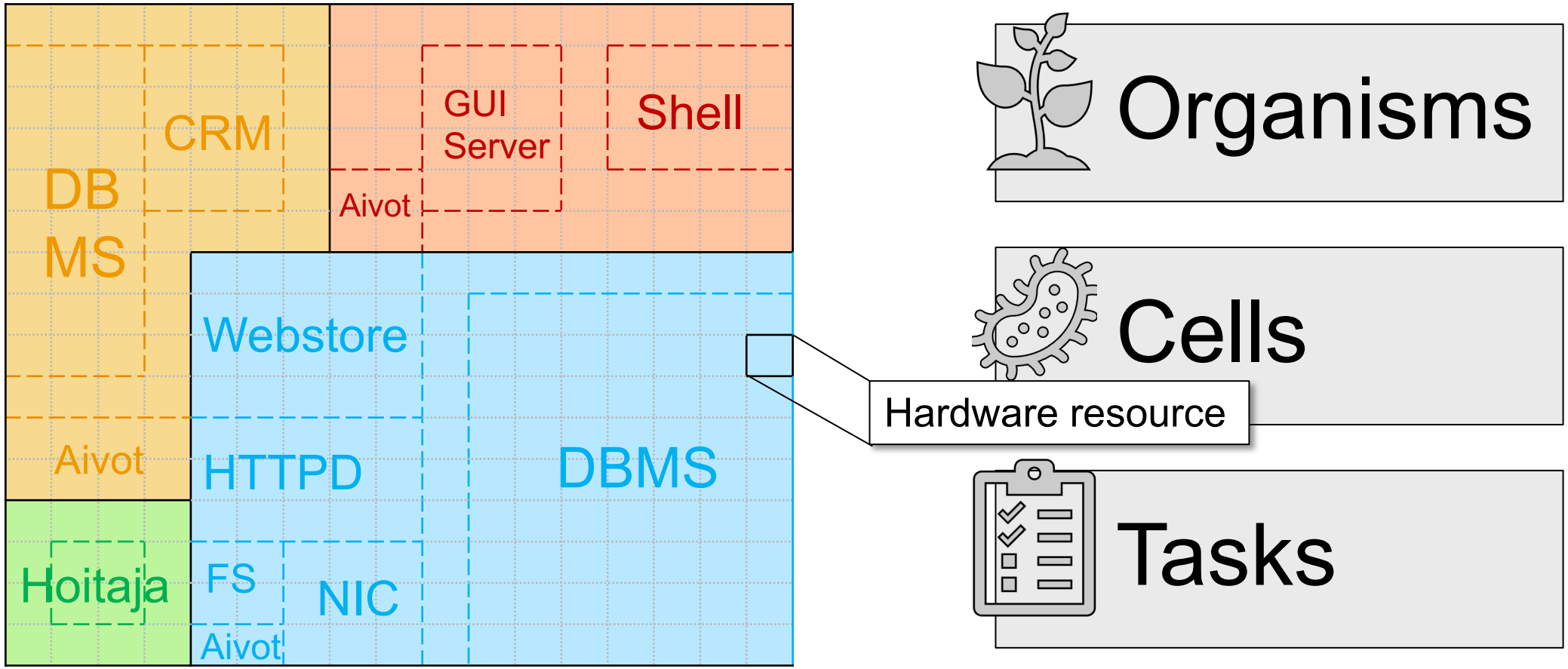


Meet

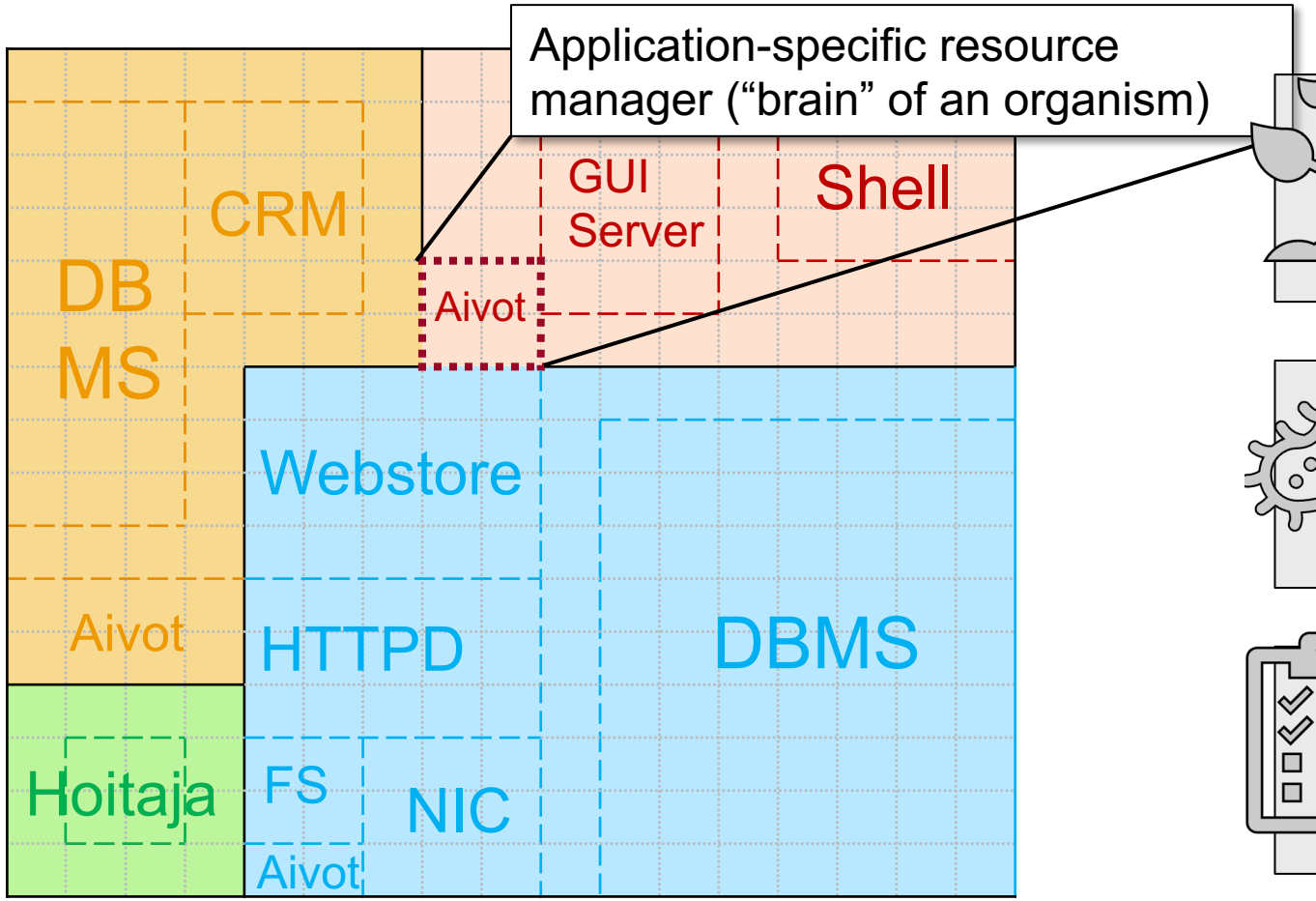
EalánOS

… an experimental implementation of the MxKernel architecture using the Genode OS Framework

EalánOS — Concepts



EalánOS — Concepts



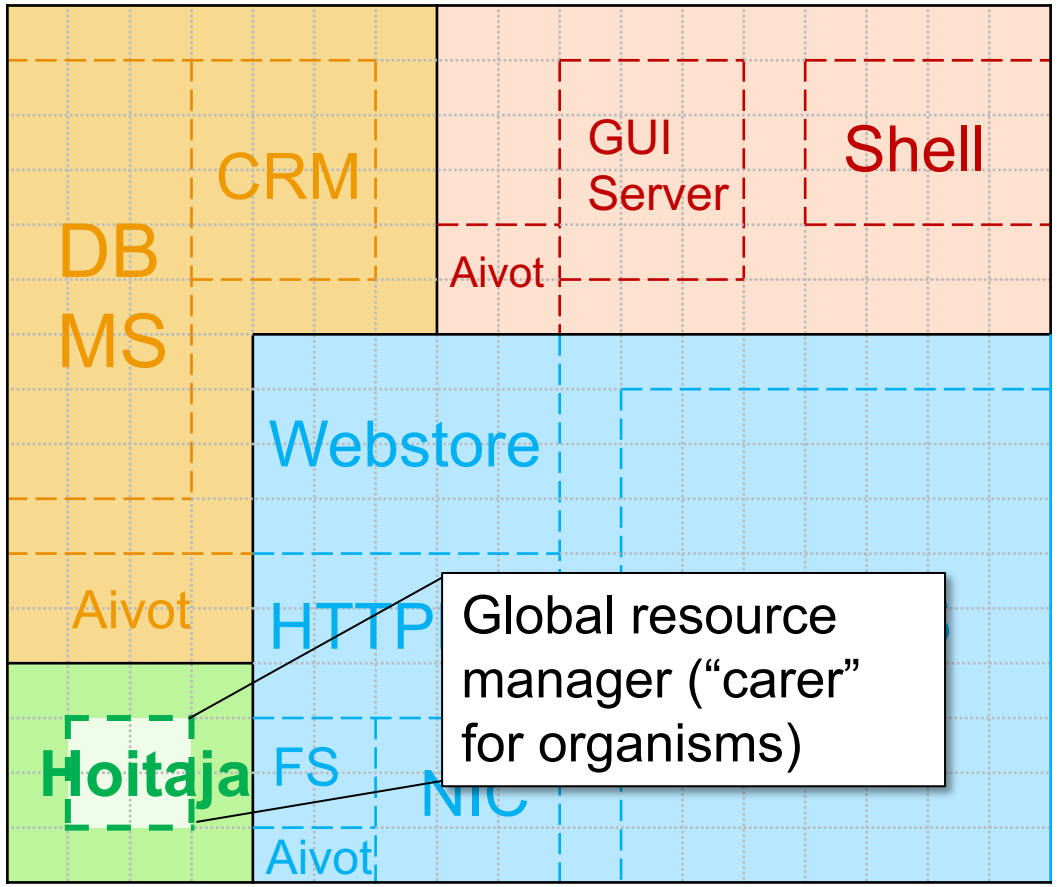
Organisms

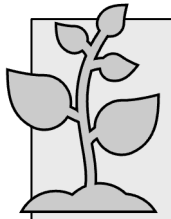
- Resource container for a **user-session**
- **Custom** OS services

Cells

Tasks

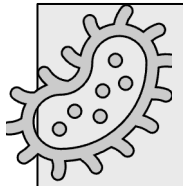
EalánOS — Concepts





Organisms

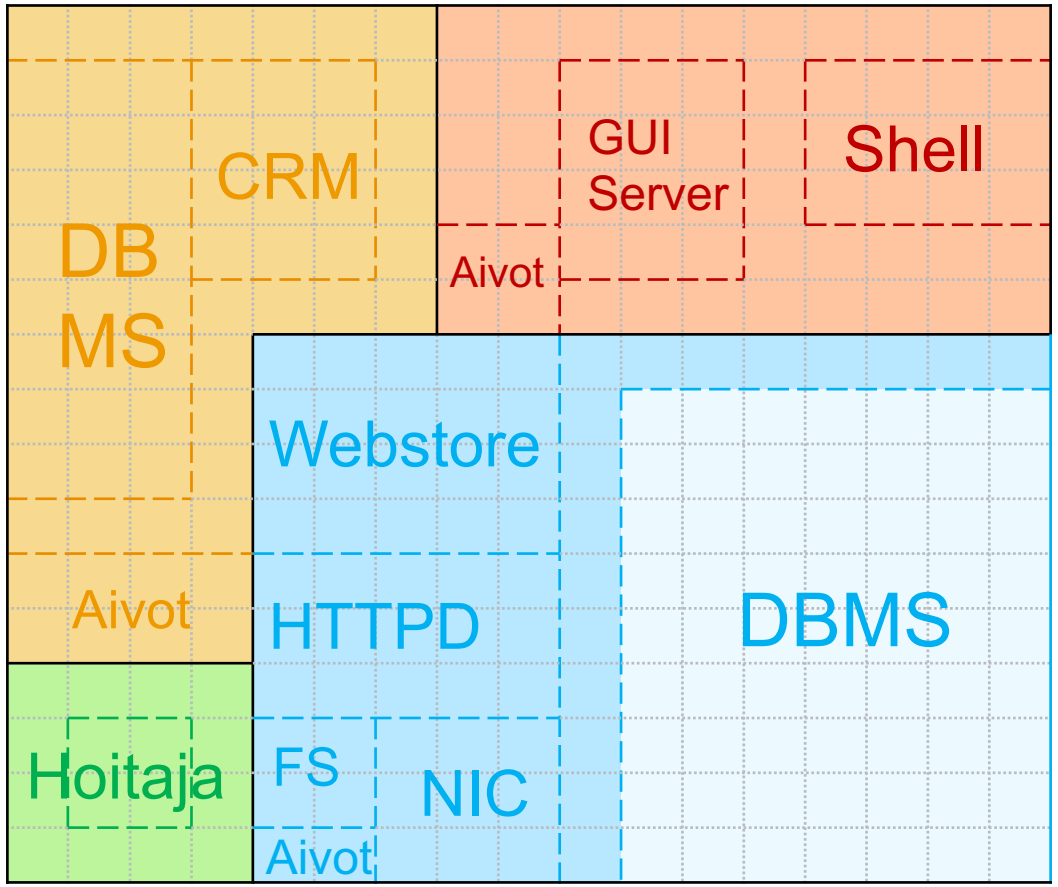
- Resource container for a **user-session**
- **Custom** OS services

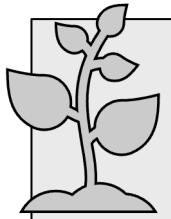


Cells

Tasks

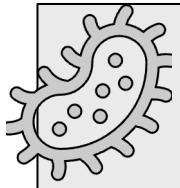
EalánOS — Concepts





Organisms

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

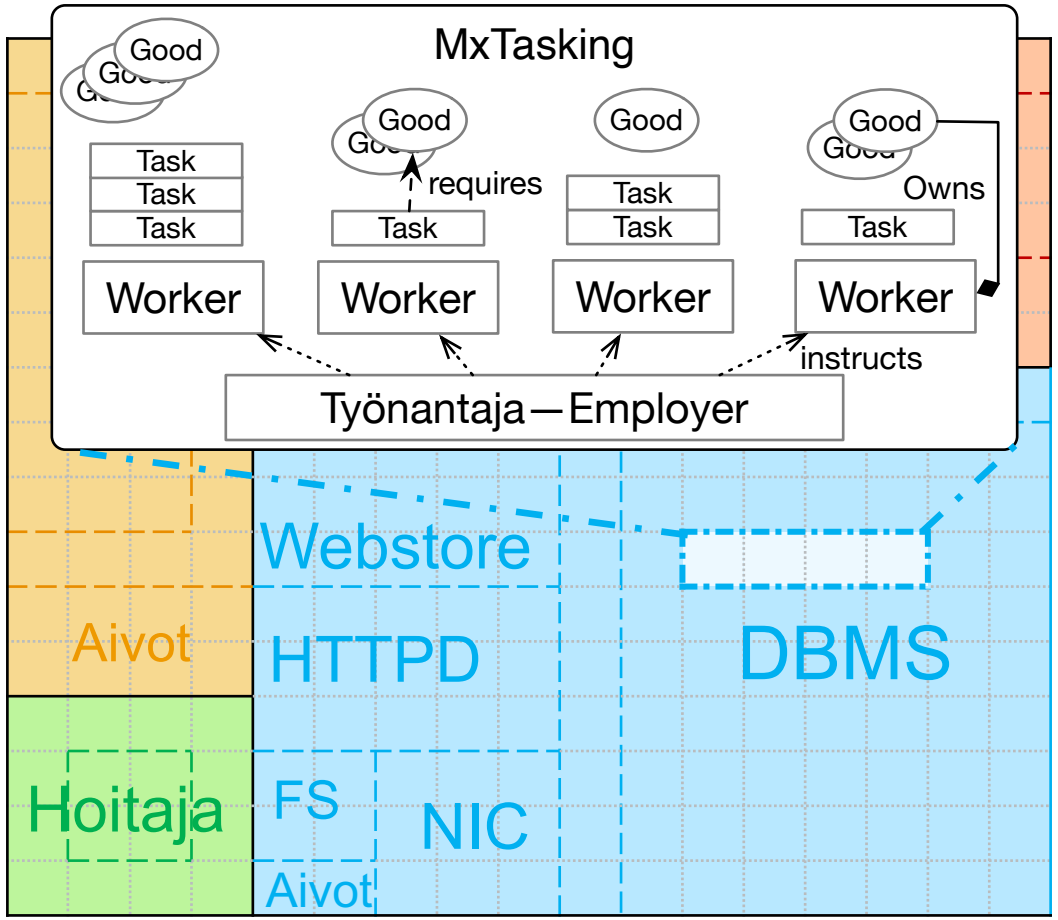


Cells

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

Tasks

EalánOS — Concepts



Organisms

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

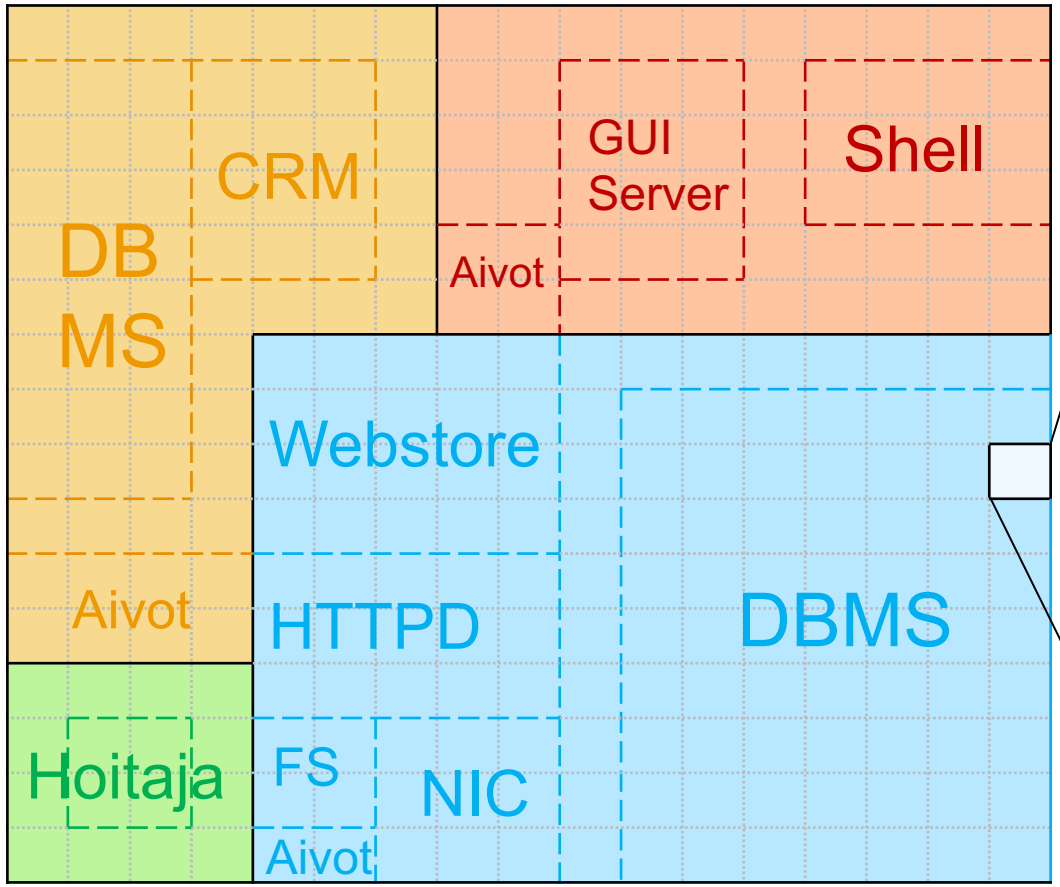
Cells

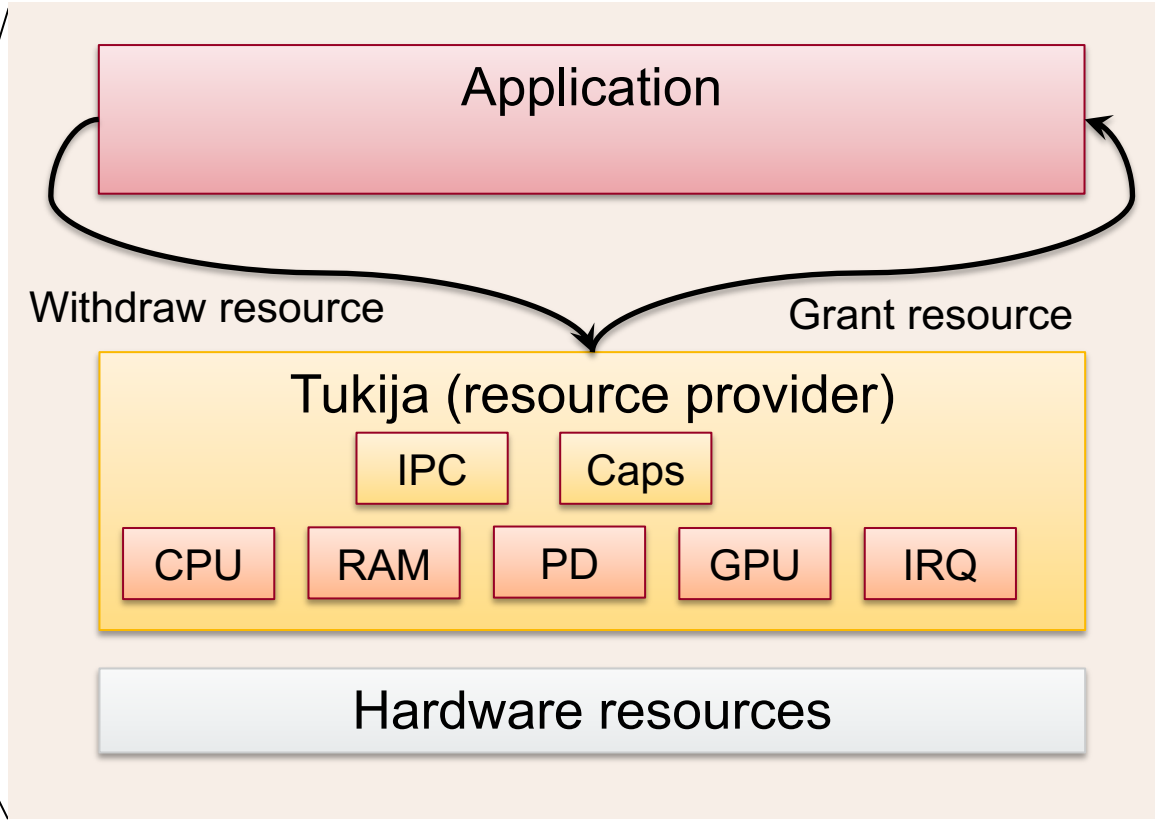
- ***Elastic*** resource container
- Contains a **single** process

Tasks

- closed units of work
- **not** preemptable

EalánOS — Architecture

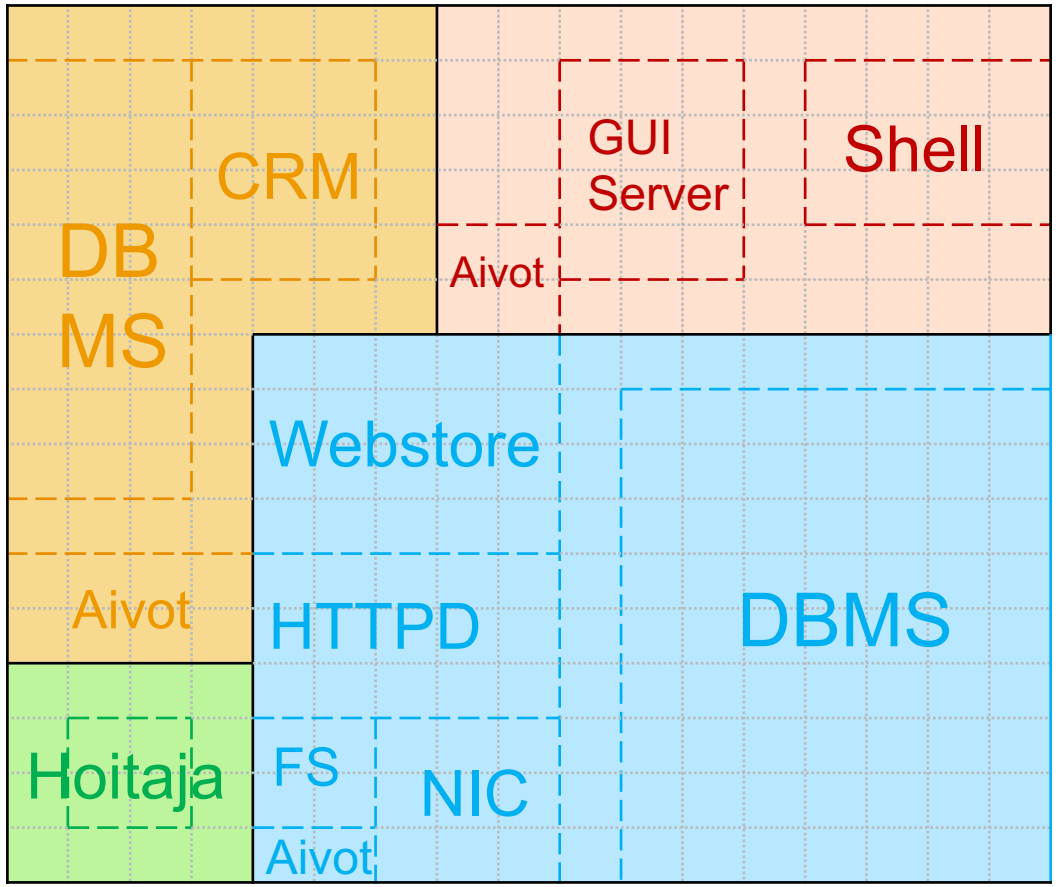


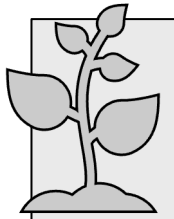


How Genode assists in creating a research OS

THE MAKING-OF EALÀNOS

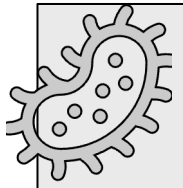
EalánOS — What do we need? What do we have?





Organisms

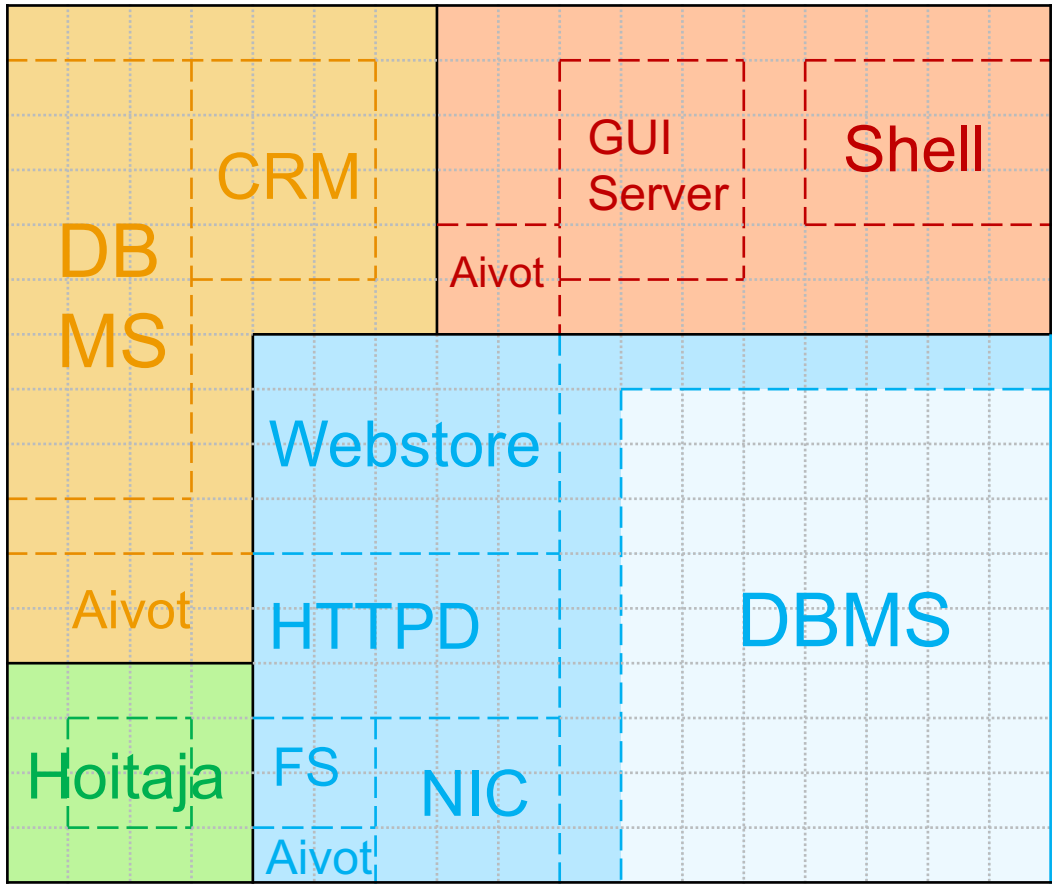
- **Service interception**

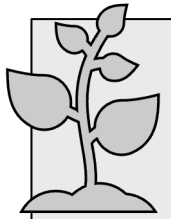


Cells

Tasks

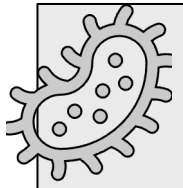
EalánOS — What do we need? What do we have?





Organisms

- Service interception

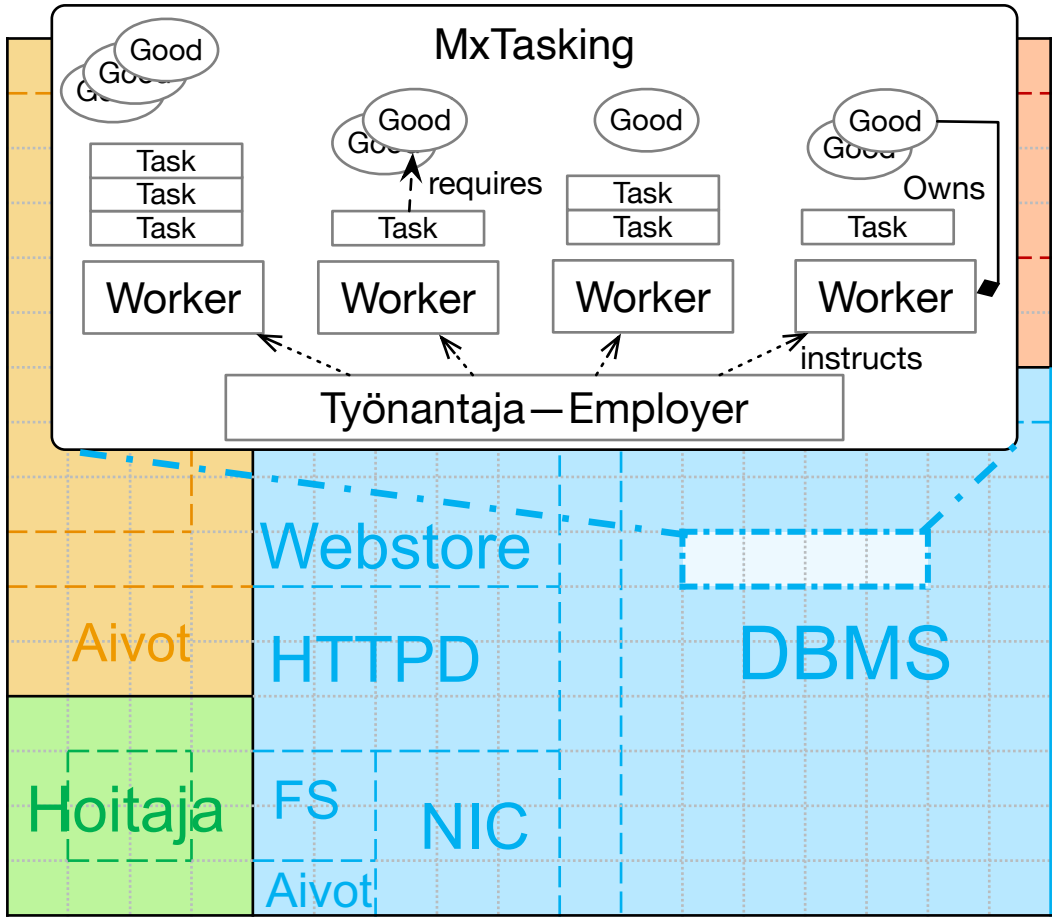


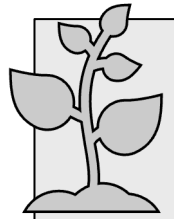
Cells

- **Genode components**

Tasks

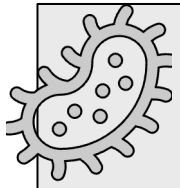
EalánOS — What do we need? What do we have?





Organisms

- Service interception



Cells

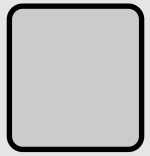
- Genode components

Tasks

- **Port MxTasking[1]**

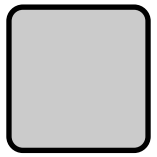
[1] MxTasking: Task-based framework with built-in prefetching and synchronization — github.com/jmuehlig/mxtasking

MxTasking — What does it need?



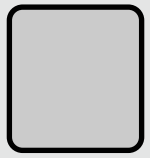
Standard C++ library

- For internal data structures
- Portability



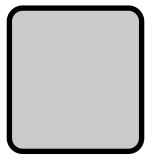
Filesystem

- For benchmarks
- For writing profiling results



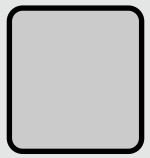
Timer support

- For garbage collection
- And profiling



Multicore support

- Needed for task parallelism



NUMA support

- For NUMA-aware task scheduling
- And data placement

MxTasking — What does it need?

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

365 LoC

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

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

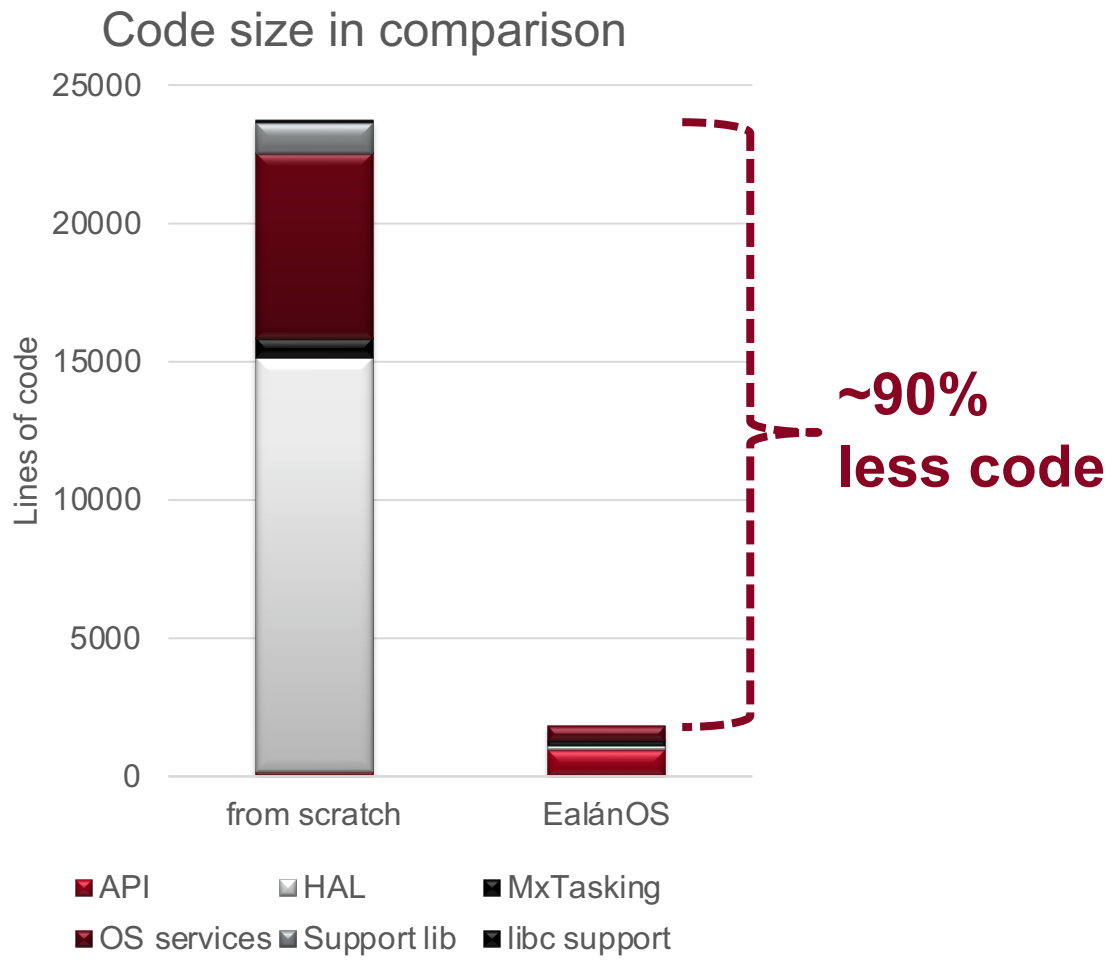
Regional heaps
- for NUMA-aware memory allocation

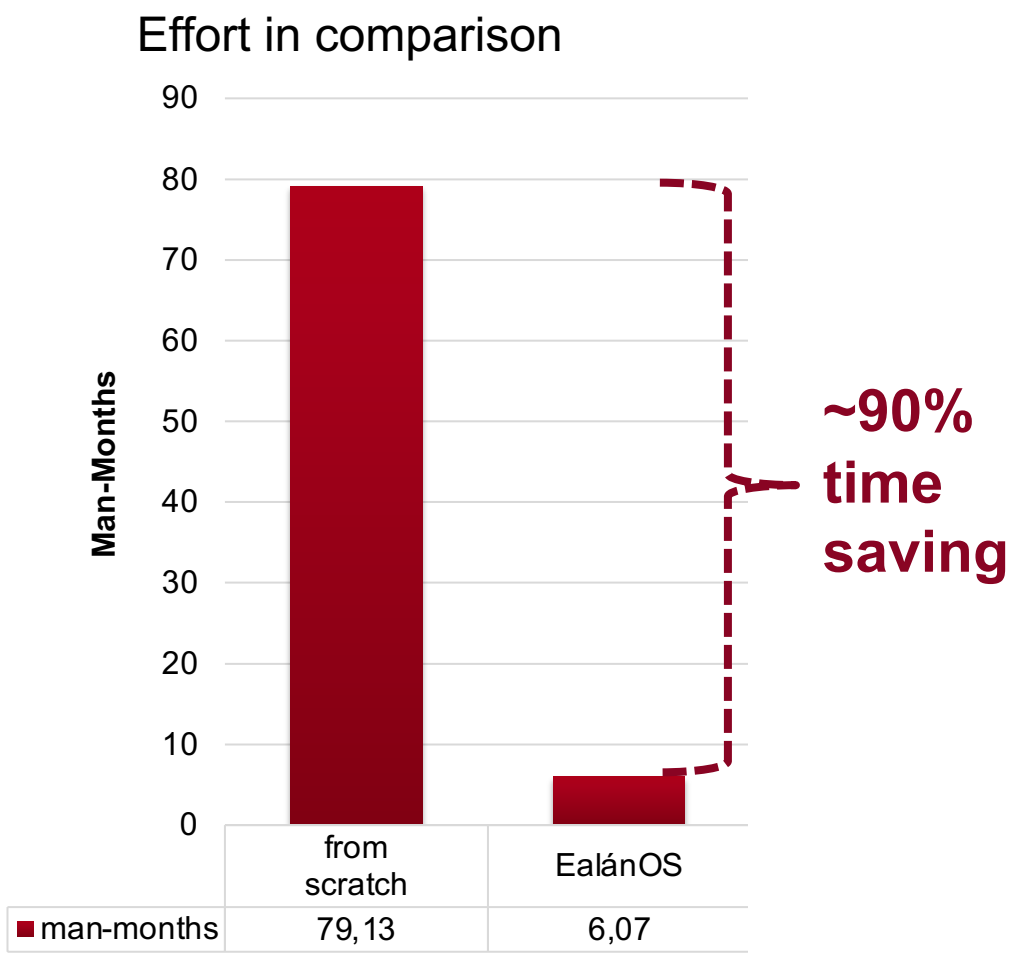
683 LoC

MxTasking glue-code

186 LoC

What you can win with Genode





Automated Experiments with Genode

TO THE LAB!

Sculpting a Genode system

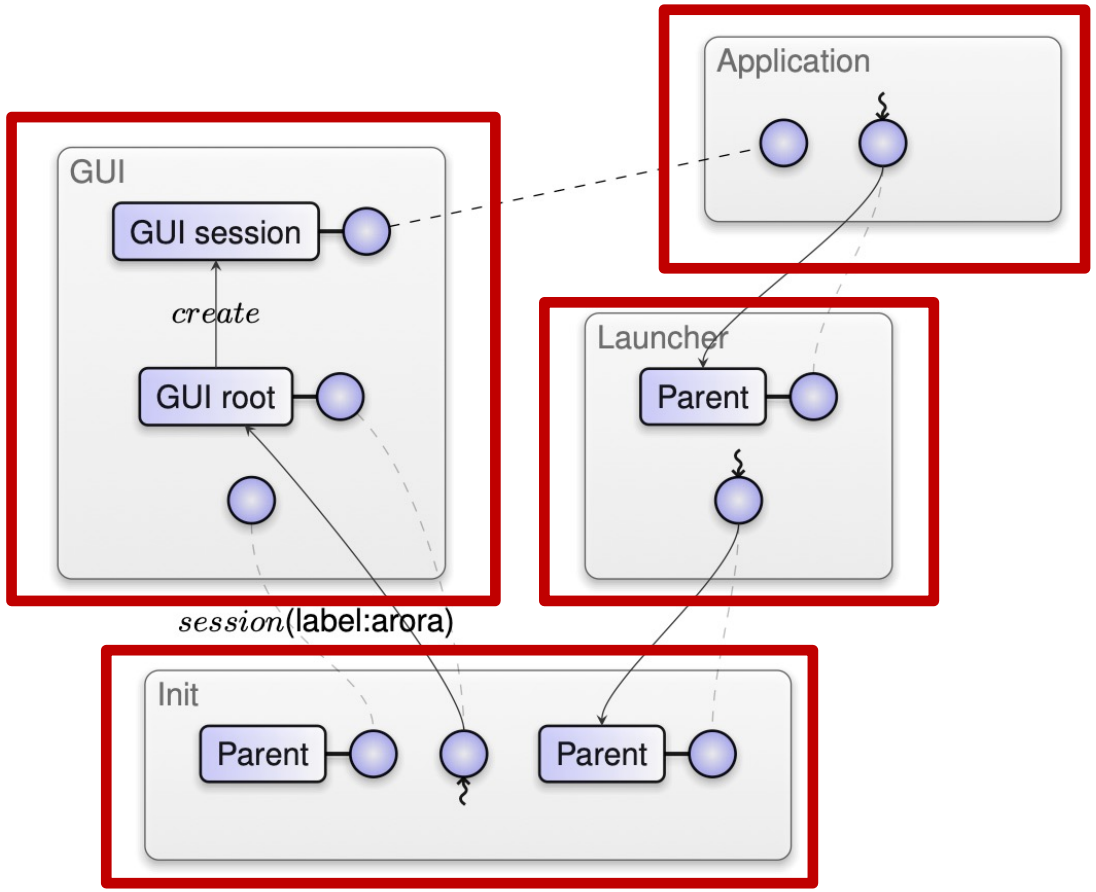


Figure 11: Session creation at the server.

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

*Source: "**Genode Foundations 22.05**" -- Norman Feske, Genode Labs GmbH, Dresden, 2022*

Sculpting a Genode system

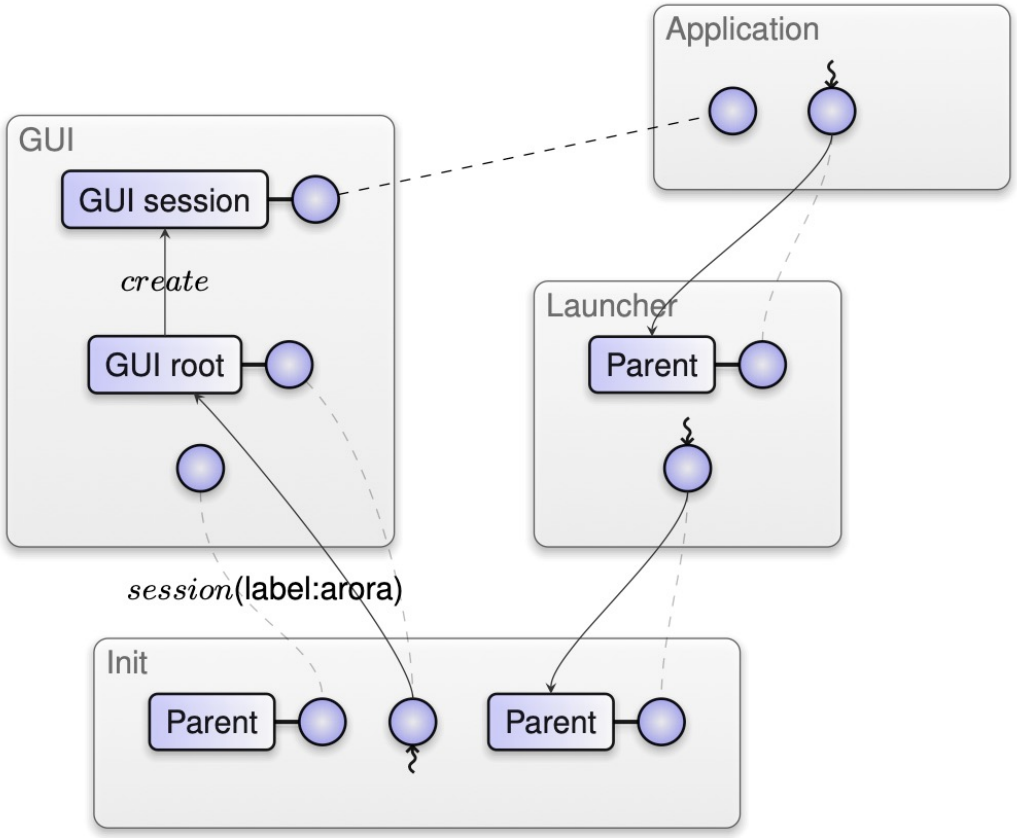


Figure 11: Session creation at the server.

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

*Source: "**Genode Foundations 22.05**" -- Norman Feske, Genode Labs GmbH, Dresden, 2022*

Sculpting a Genode system

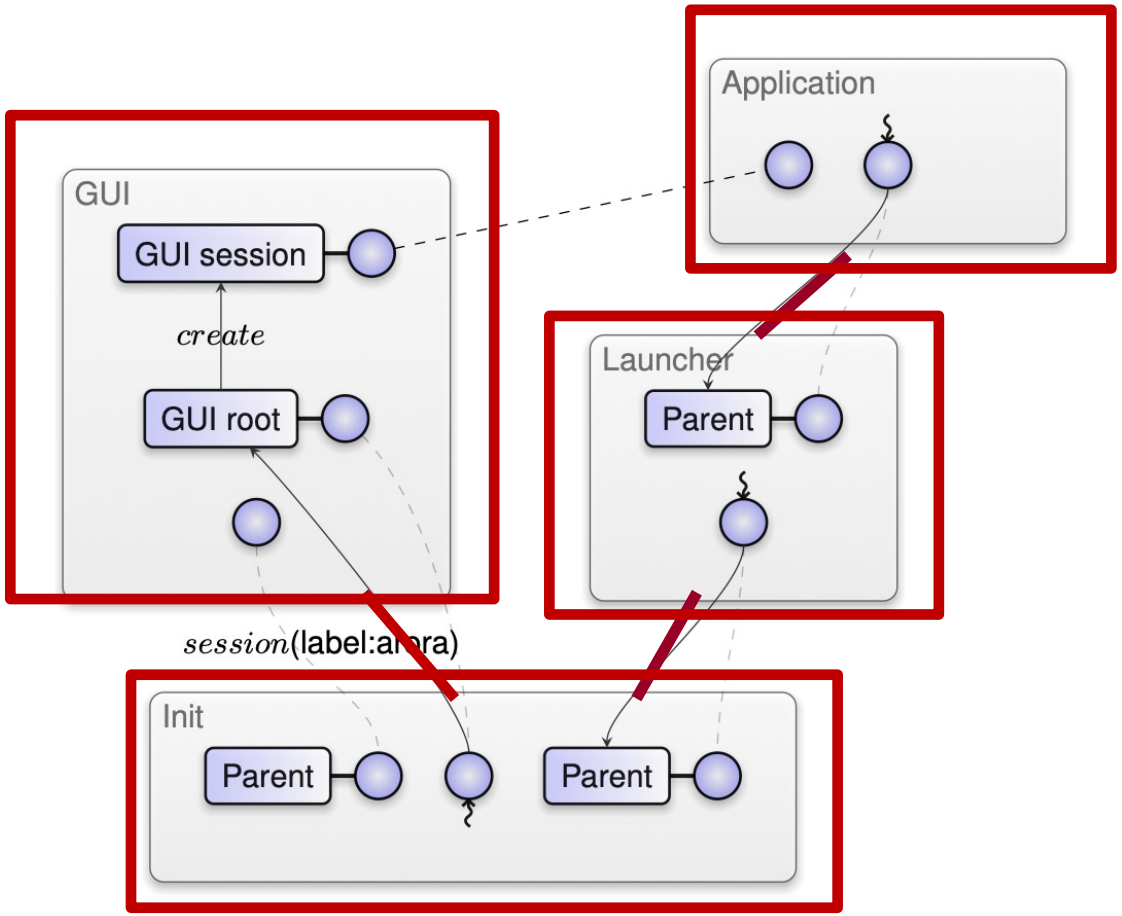


Figure 11: Session creation at the server.

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

*Source: "**Genode Foundations 22.05**" -- Norman Feske, Genode Labs GmbH, Dresden, 2022*

Sculpting a Genode system

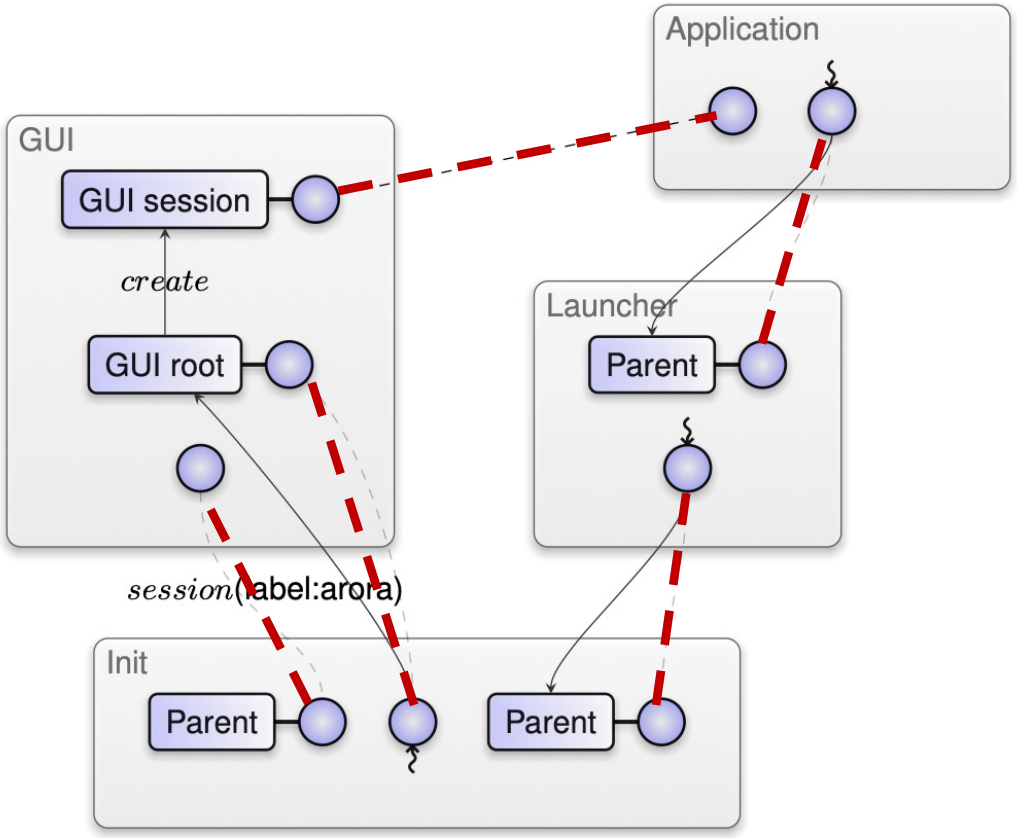


Figure 11: Session creation at the server.

- 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

*Source: "**Genode Foundations 22.05**" -- Norman Feske, Genode Labs GmbH, Dresden, 2022*

Sculpting a Genode system

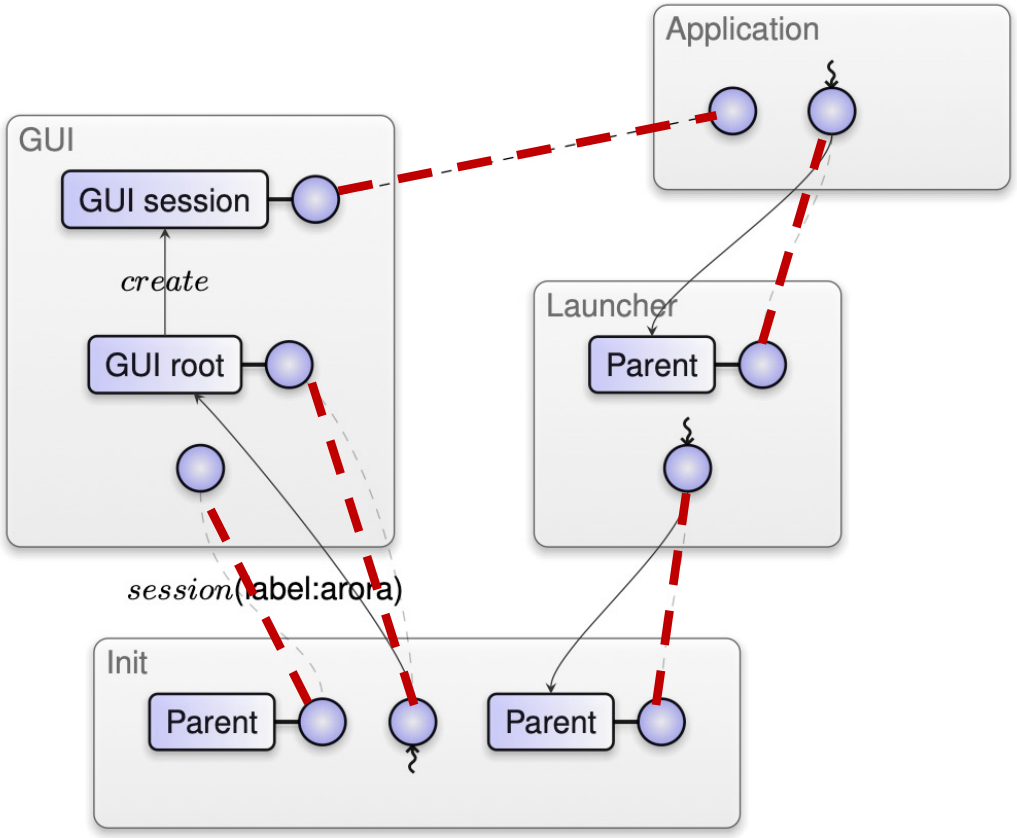


Figure 11: Session creation at the server.

Now, let's try an example experiment!

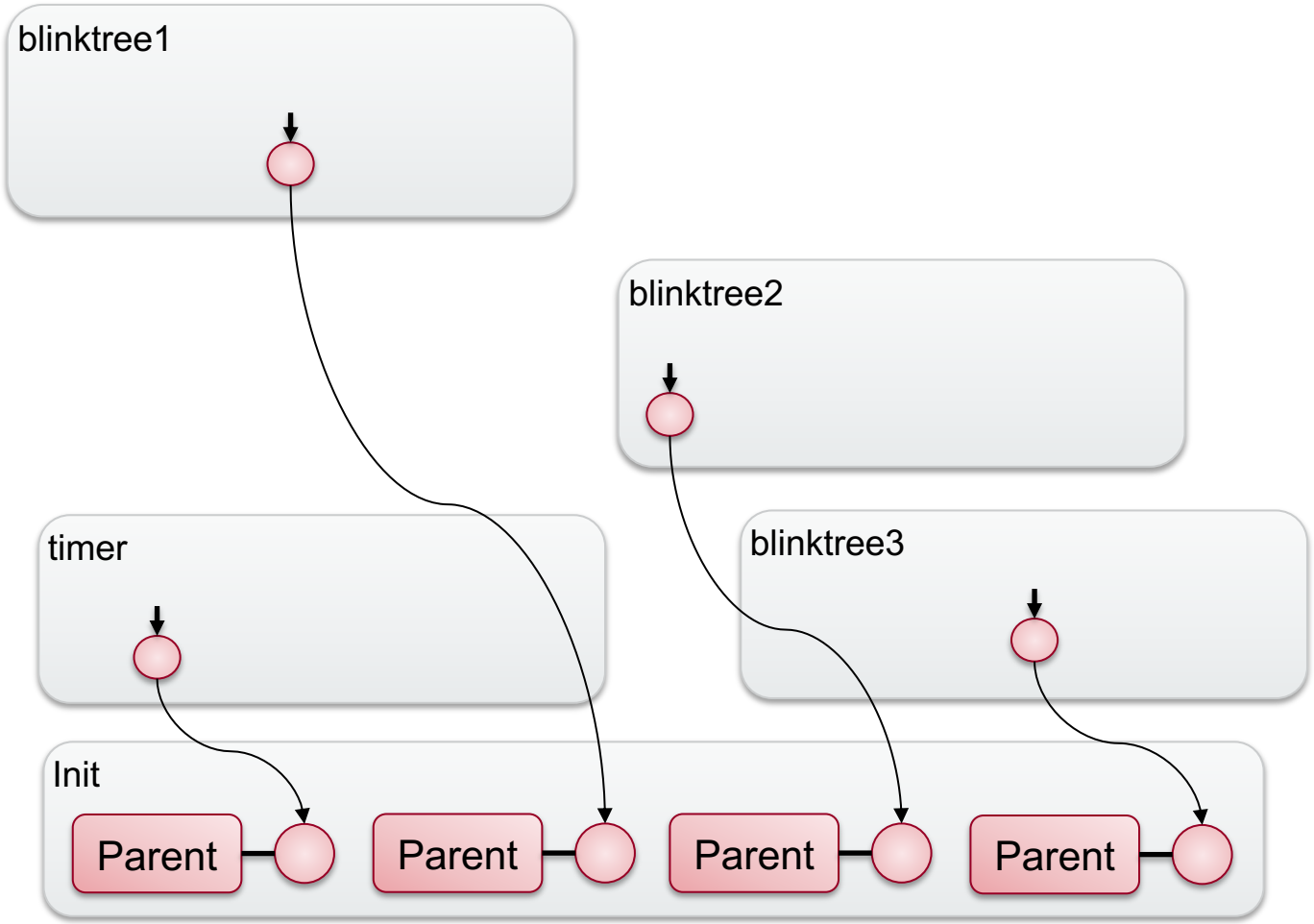
*Source: "**Genode Foundations 22.05**" -- Norman Feske, Genode Labs GmbH, Dresden, 2022*

Example: B-link tree benchmark

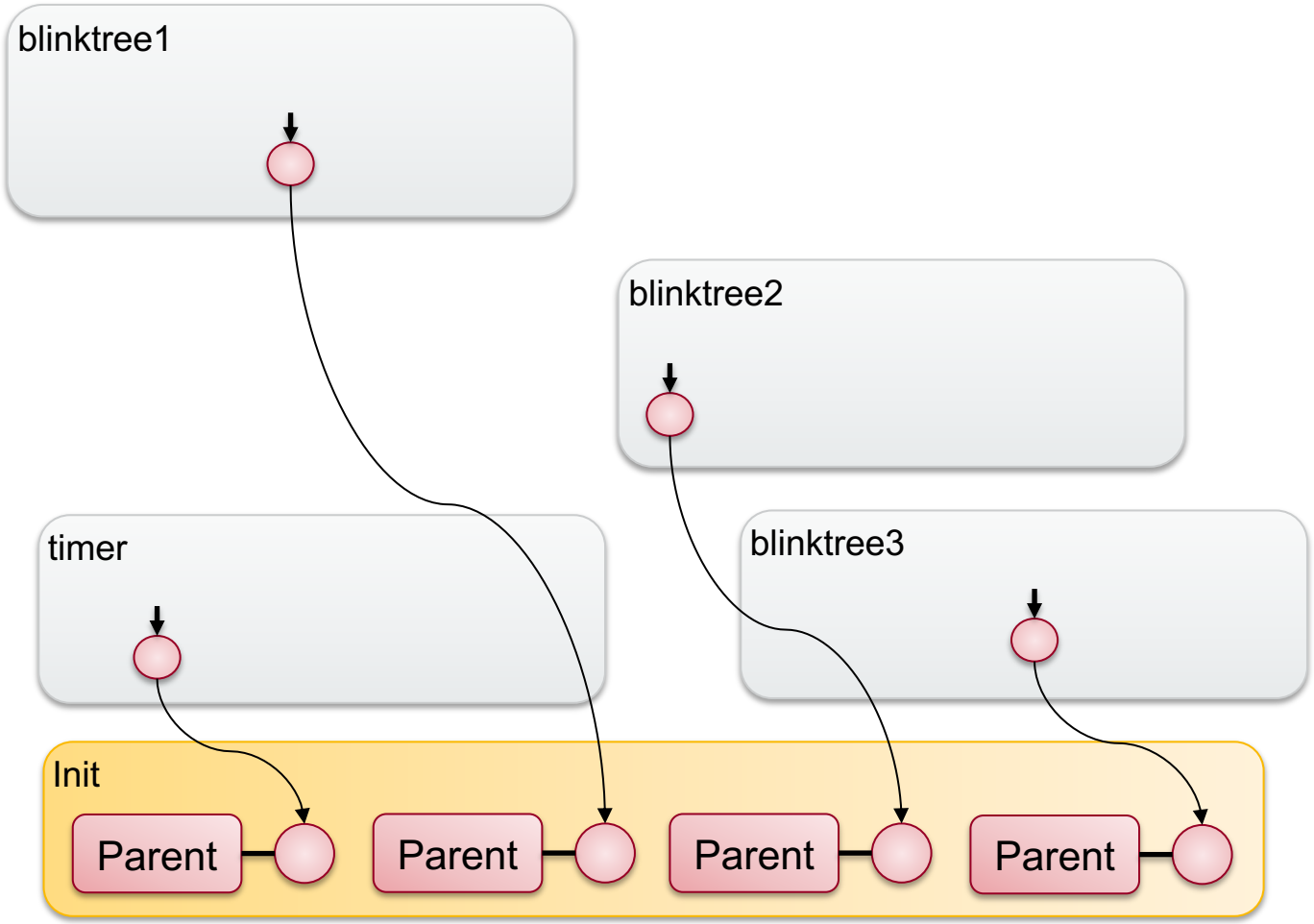
- Investigate how **throughput** of benchmark is **affected** when we run **multiple instances**
  - On the **same** set of CPU cores
  - On a **disjunct** set of CPU cores
- **Which** scenario will yield the **higher** throughput at the respective **maximum** of cores?

- 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

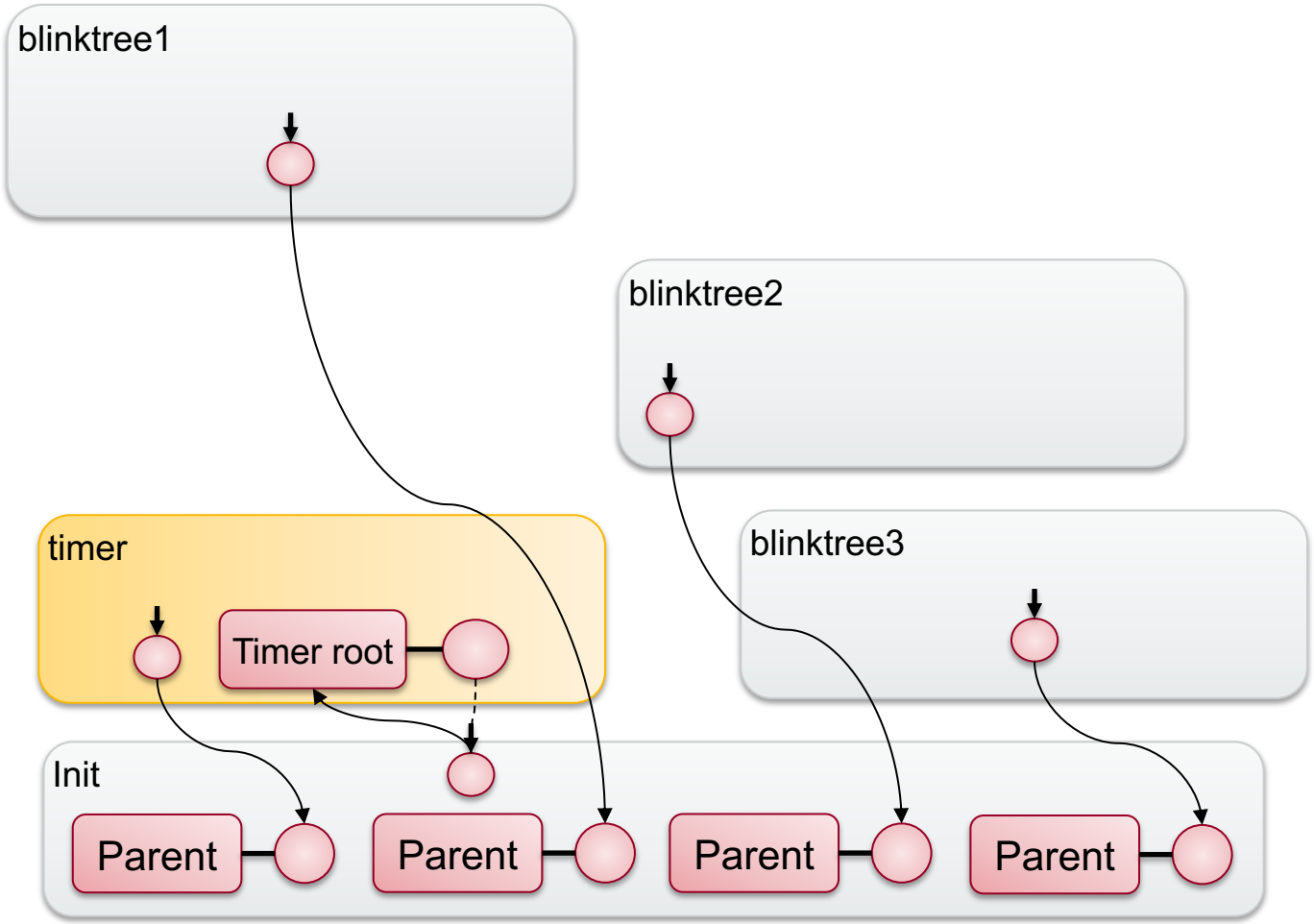


Example: B-link tree benchmark



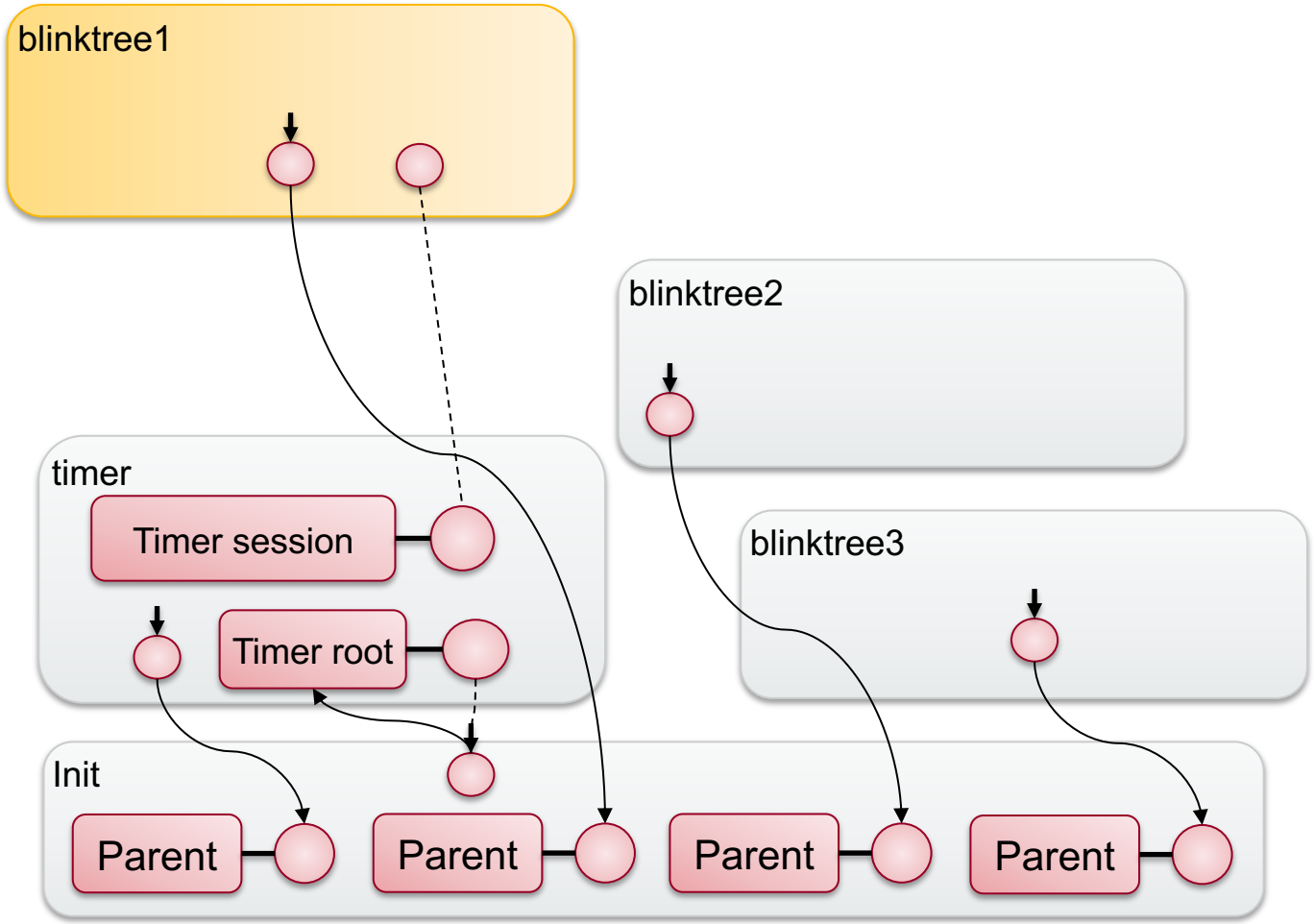

```
<config>
    <start name="timer">
    </start>
    <start name="blinktree1">
        <binary name="blinktree"/>
    </start>
    <start name="blinktree2">
        <binary name="blinktree"/>
    </start>
    <start name="blinktree3">
        <binary name="blinktree"/>
    </start>
</config>
```

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

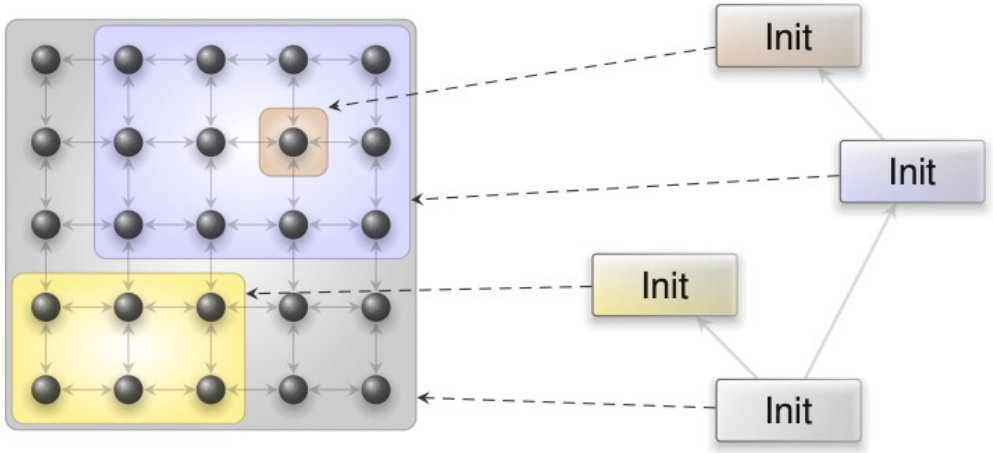
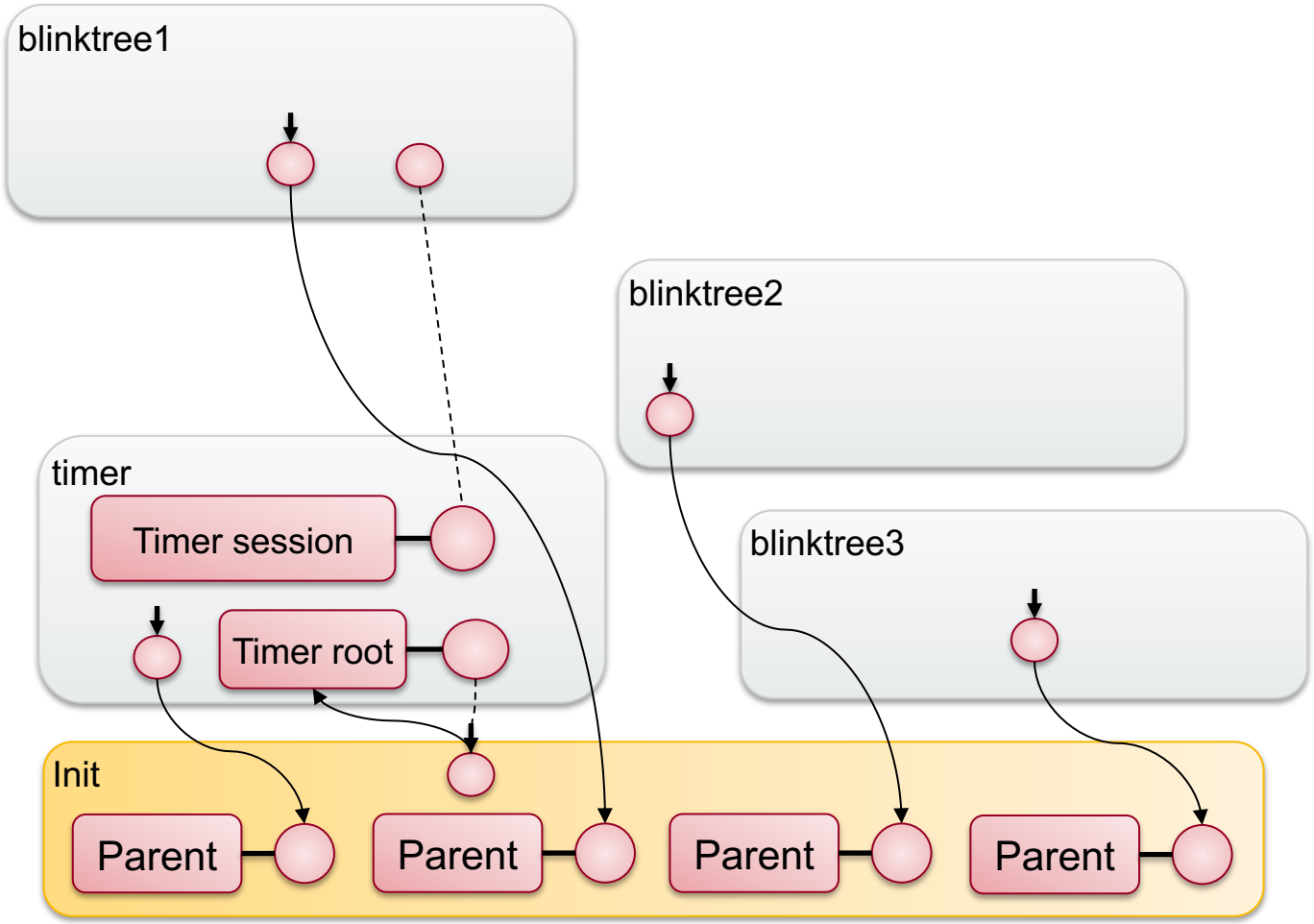


Figure 48: Successive virtualization of CPU affinity spaces by nested instances of init

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

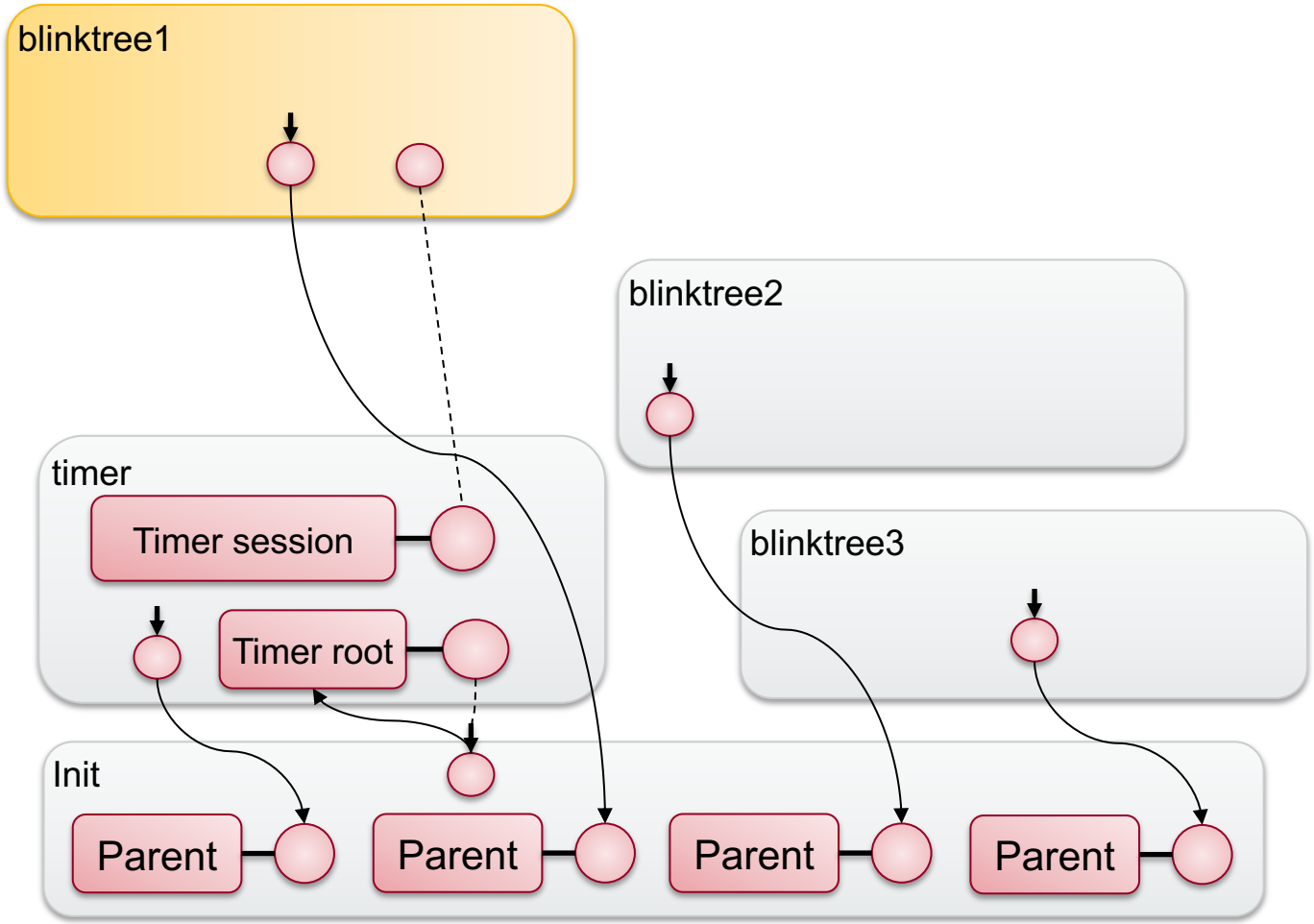
*Source: "**Genode Foundations 22.05**" -- Norman Feske, Genode Labs GmbH, Dresden, 2022*

Example: B-link tree benchmark



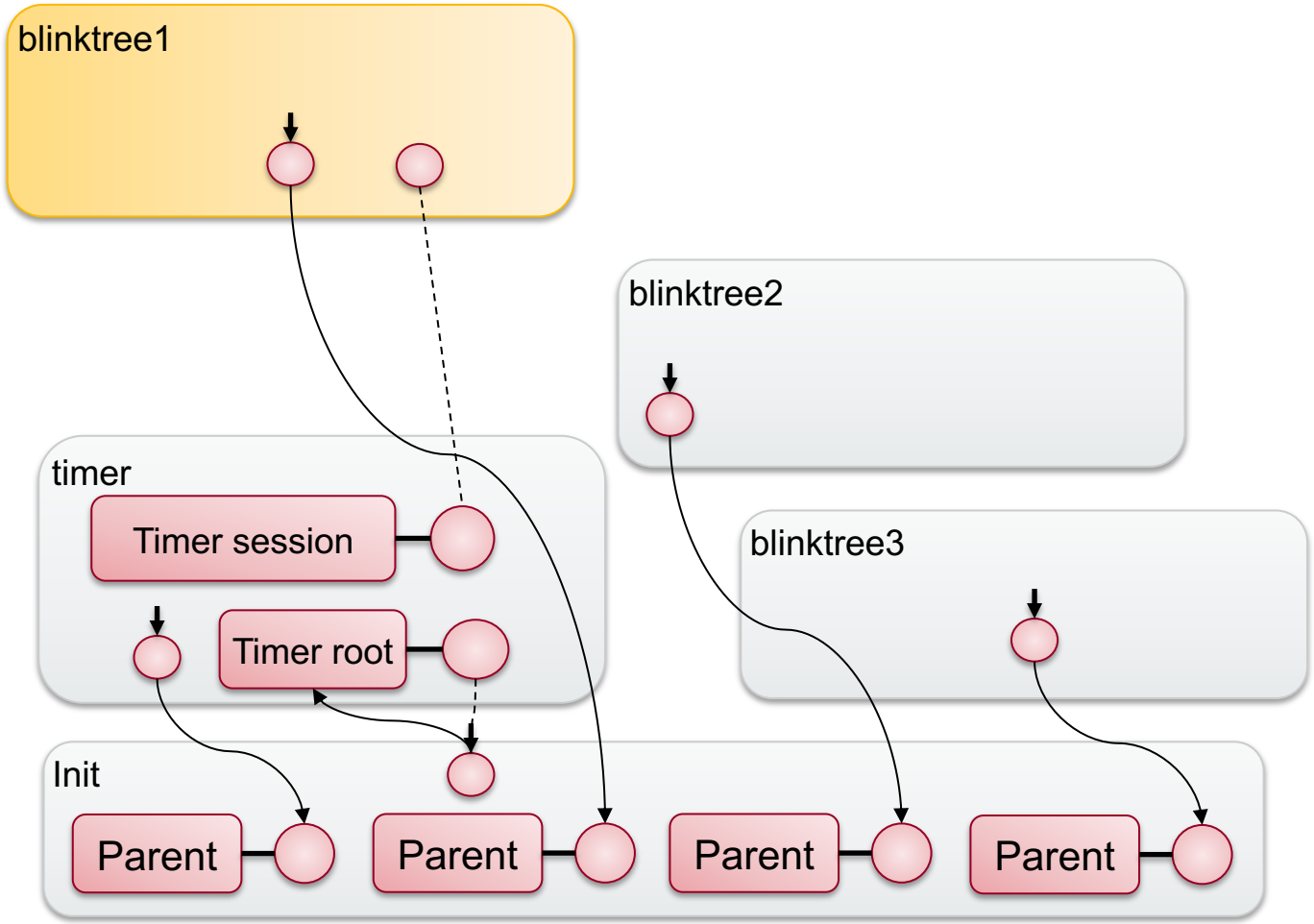

```
<config>
  <affinity-space width=”64” height="1"/>
  <start name=“timer”>
        <provides><service name=“Timer”/></provides>
    <route>
          <any-service><parent/><any-child/></any-service>
    </route>
  </start>
  <start name=“blinktree1”>
    <binary name=“blinktree”/>
    <route>
      <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



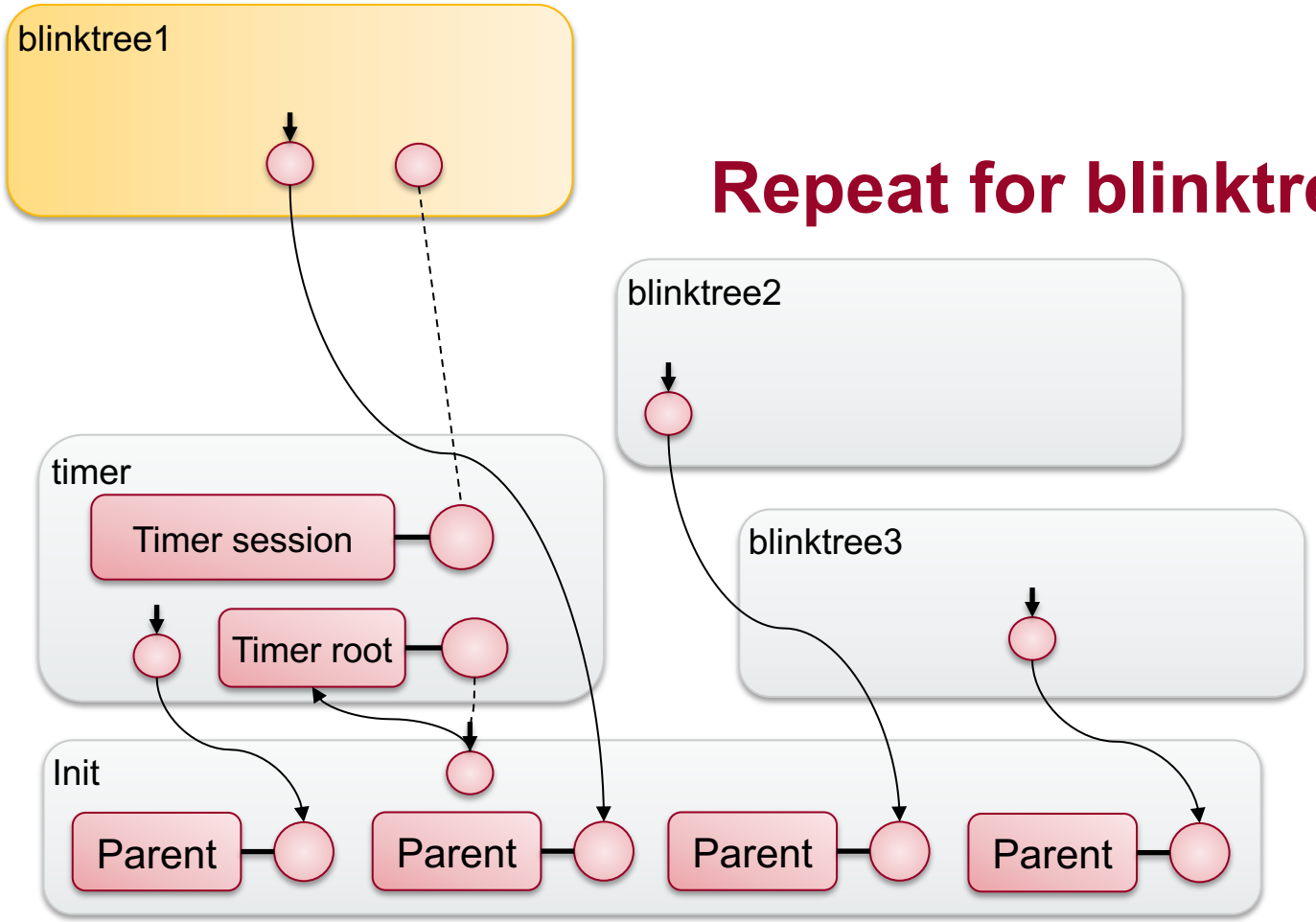

```
<config>
   <affinity-space width=”64” height="1"/>
   <start name=“timer”>
          <provides><service name=“Timer”/></provides>
       <route>
             <any-service><parent/><any-child/></any-
service>
          </route>
   </start>
   <start name=“blinktree1”>
          <affinity xpos="1" ypos="0"
width=”63" height="1"/>
       <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




```
<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"/></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>
```

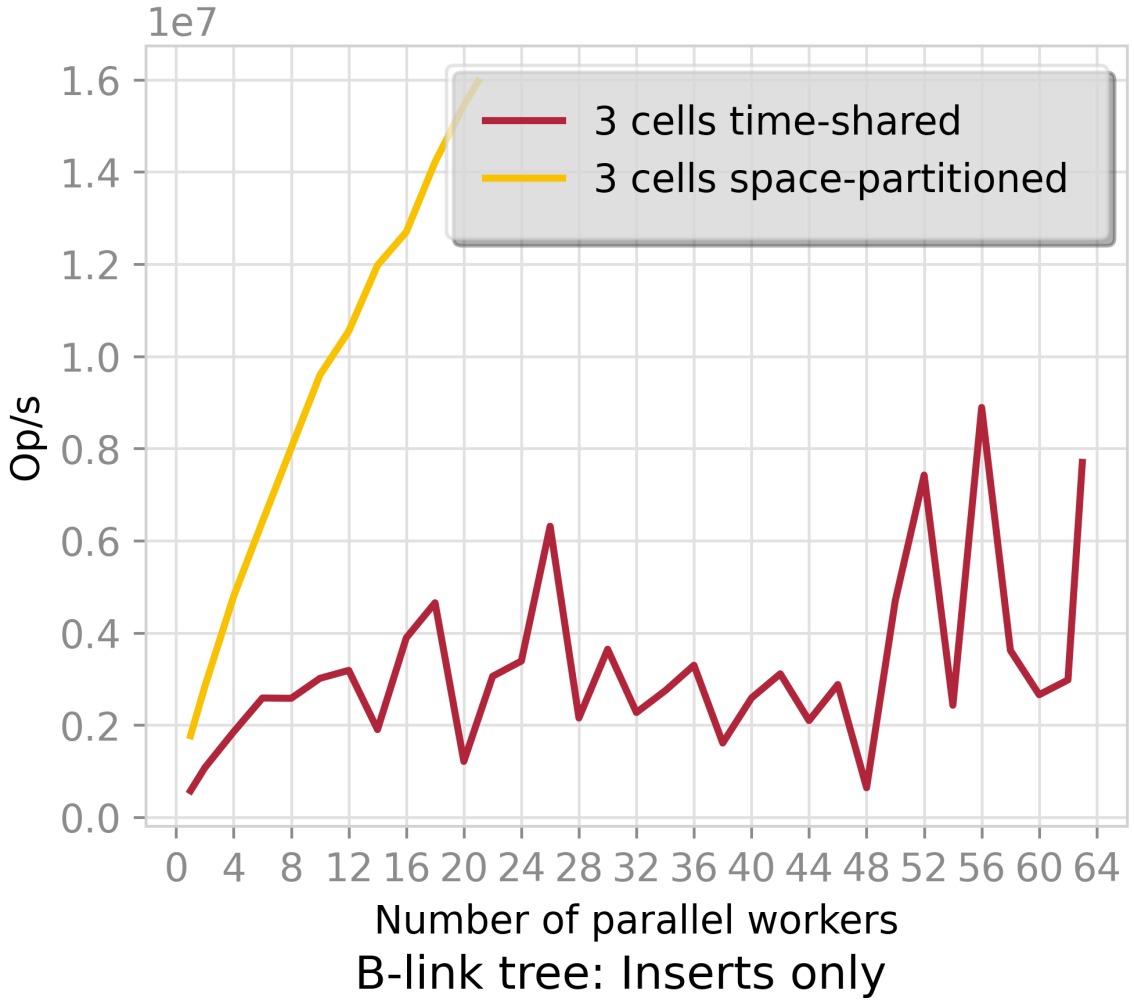
Example: B-link tree benchmark



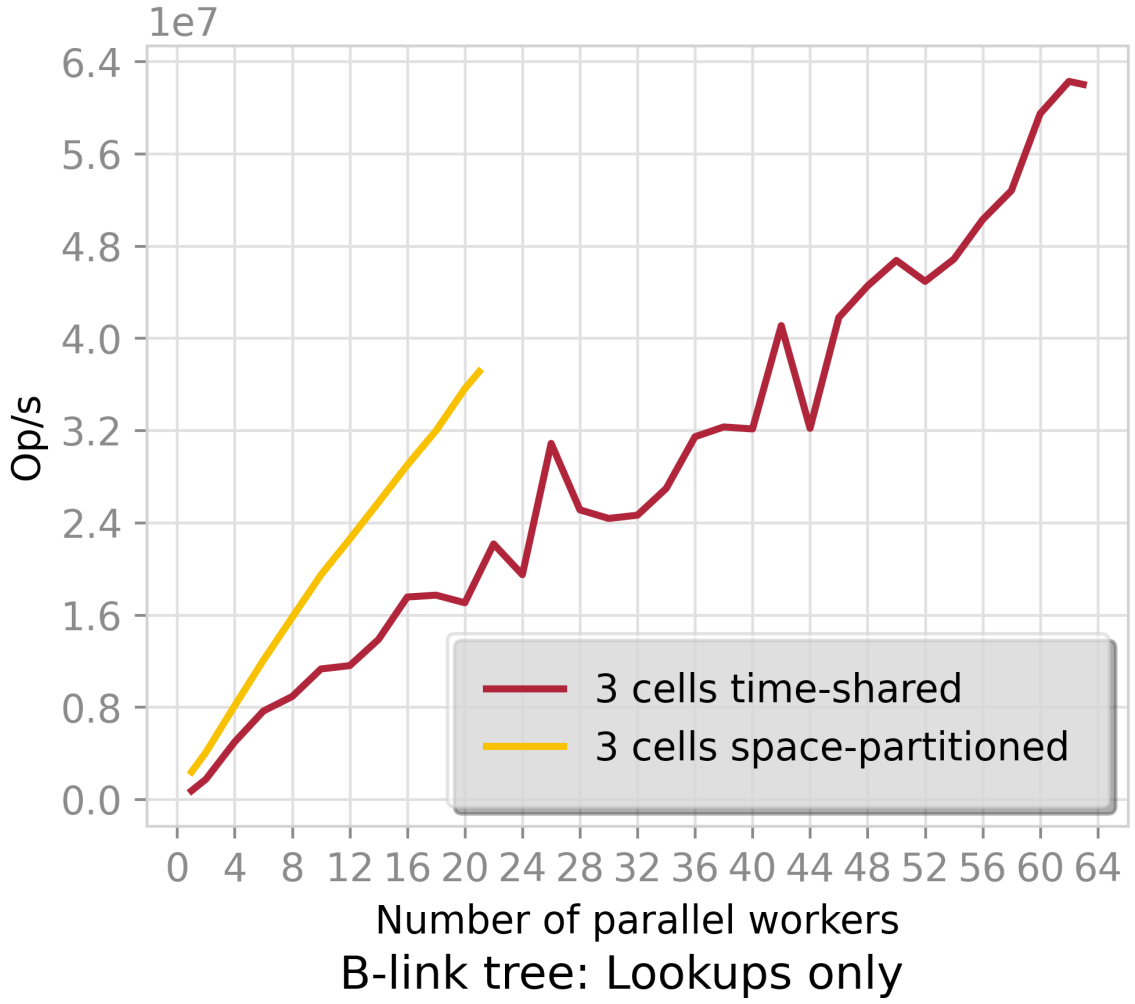
Repeat for blinktree2 and blinktree3

```
<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"/></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: Results



B-link tree: Inserts only

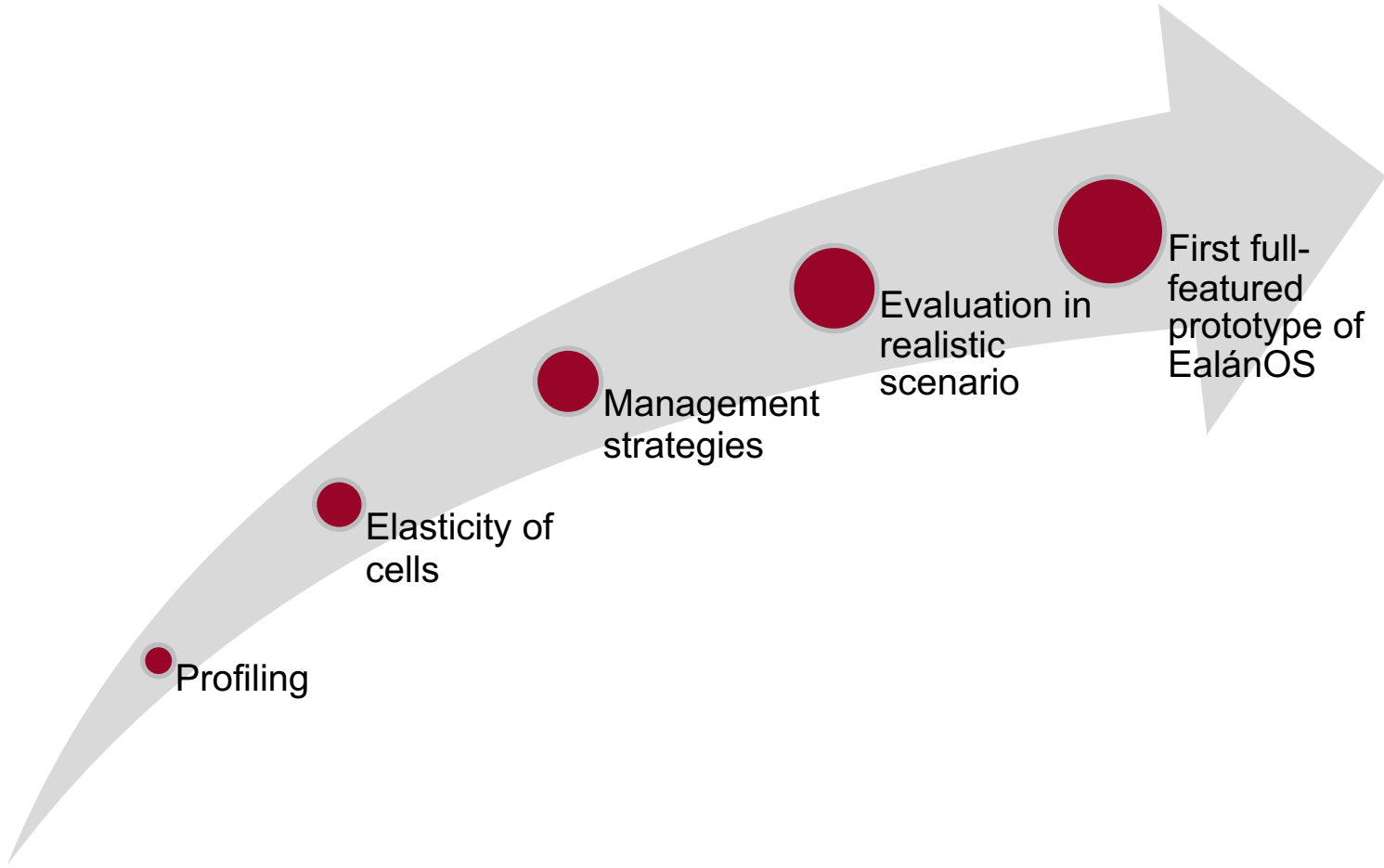


B-link tree: Lookups only

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



- github.com/mmueller41/genode
- github.com/mmueller41/NOVA

EalánOS

- github.com/jmuehlig/mxtasking
- dbis.cs.tu-dortmund.de

MxTasking

Get in touch

Embedded Software Systems

- ess.cs.uos.de
- www.uni-osnabrueck.de

SPP2037

SCALABLE DATA MANAGEMENT FOR FUTURE HARDWARE

- mxkernel.org
- dfg-spp2037.org