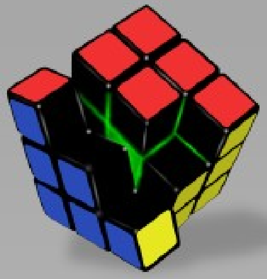


technische universität
dortmund

# *Increasing the Predictability of Modern COTS Hardware through Cache-Aware OS-Design*

11th Workshop on Operating Systems Platforms for
Embedded Real-Time Applications

**Hendrik Borghorst**

hendrik.borghorst@udo.edu
https://ess.cs.tu-dortmund.de/~hb

Embedded System Software Group
Computer Science 12, TU Dortmund

# Cyber-physical systems

Situation today:

- Specialized systems are widespread
- Tight time bounds critical

Future:

- Multiple small real-time task-sets on one system preferable

# Cyber-physical systems

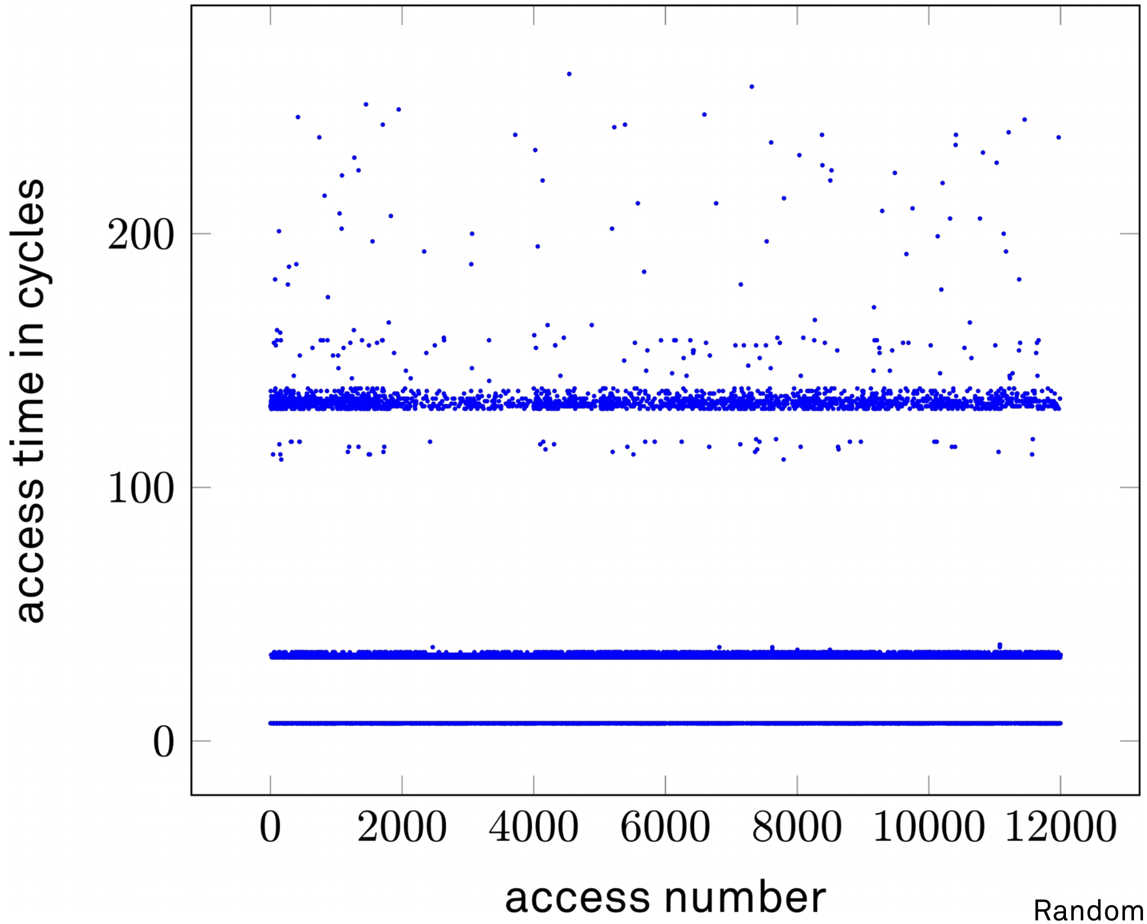
Situation today:

- Specialized systems are widespread
- Tight time bounds critical

Future:

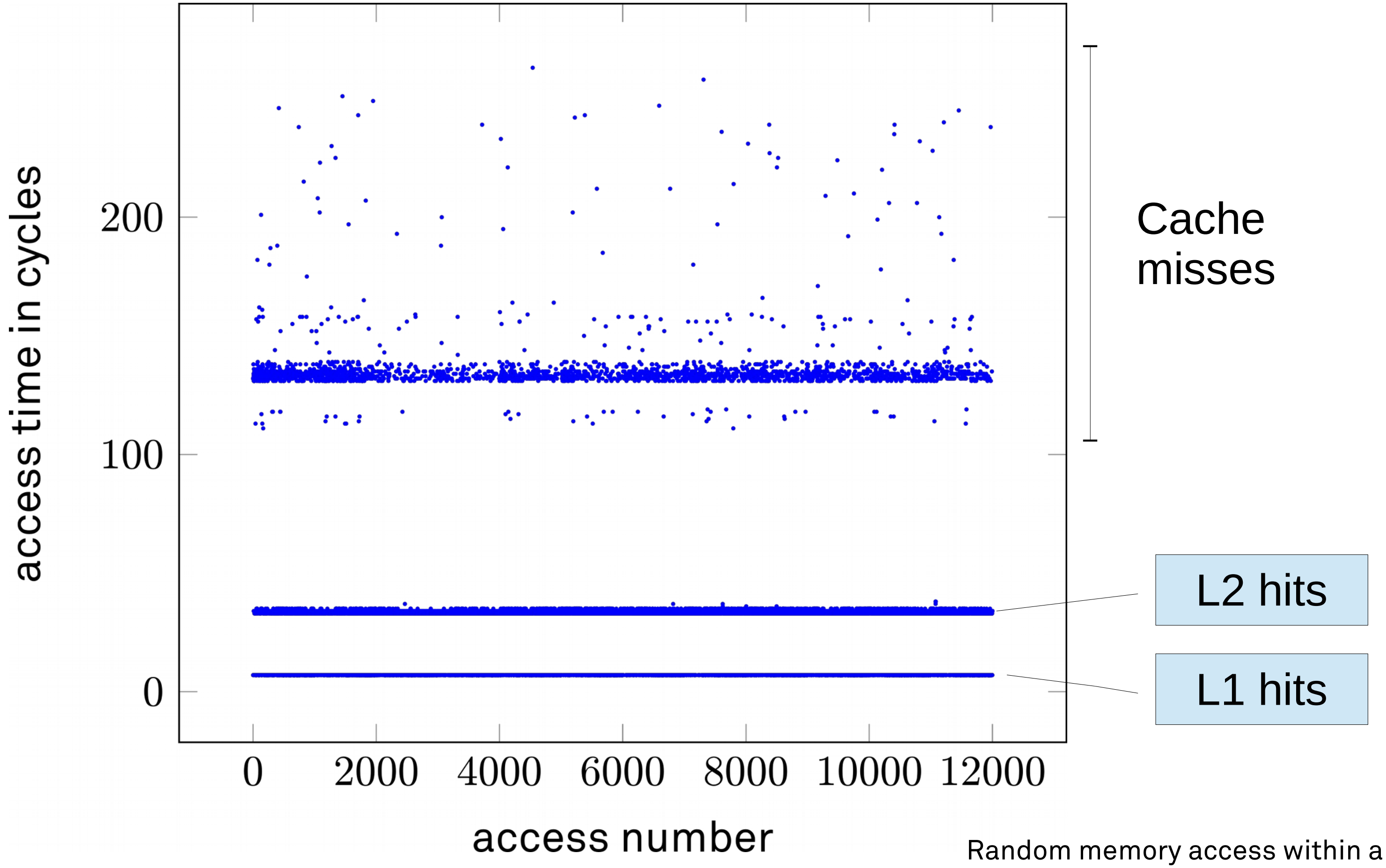
- Multiple small real-time task-sets on one system preferable
- Why is off-the-shelf hardware not used for such systems?

# Random data access



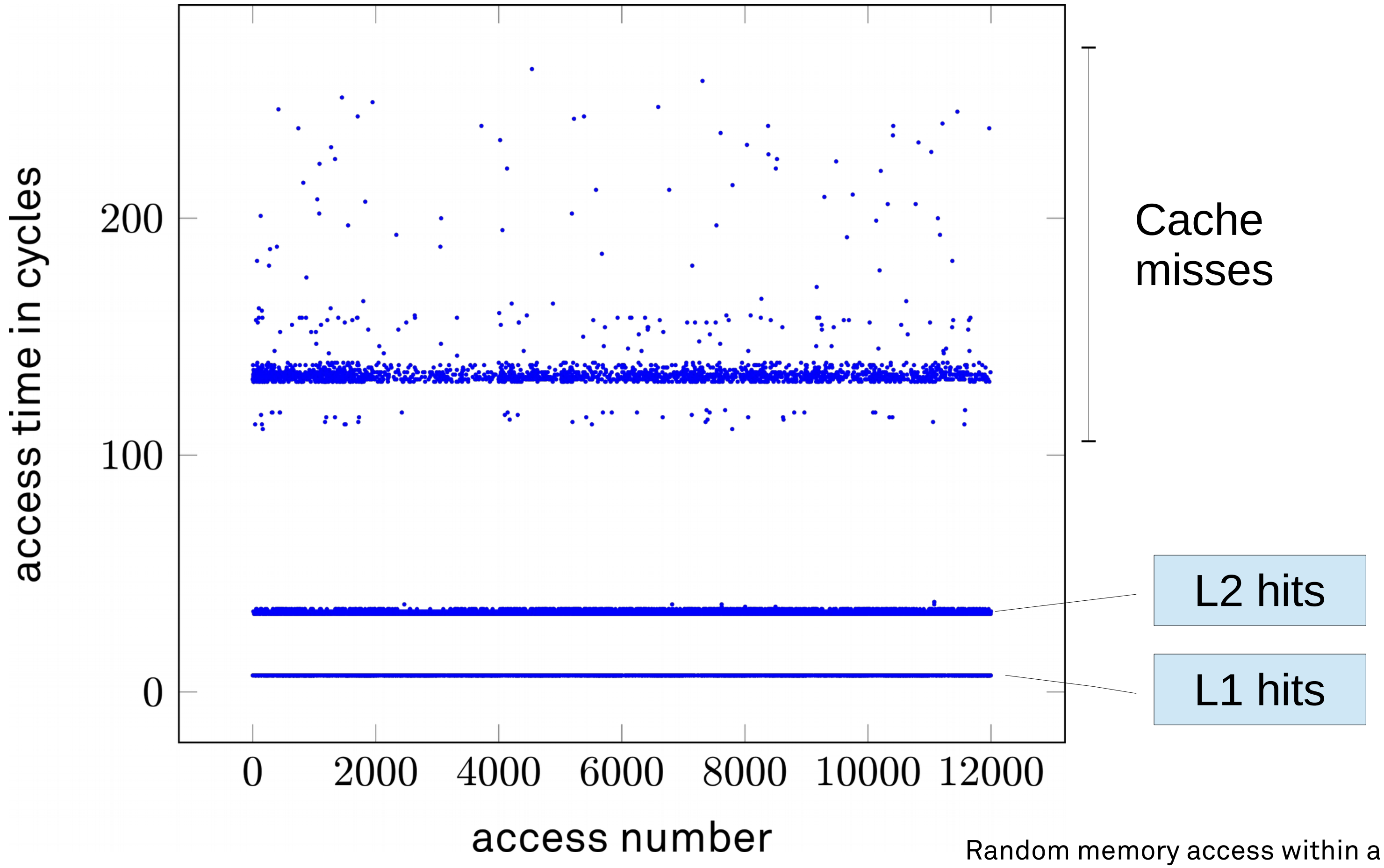
Random memory access within a 64kB range

# Random data access



Random memory access within a 64kB range

# Random data access



Random memory access within a 64kB range

Unstable execution times

# Causes for unpredictability

- DRAM[1]
  - Unstable access latency
- Shared buses between multiple cores

→ Overall system response time unstable

[1] Dasari, D.; Akesson, B.; Nelis, V.; Awan, M.A.; Petters, S.M., "Identifying the sources of unpredictability in COTS-based multicore systems," Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on , vol., no., pp.39,48, 19-21 June 2013

# Causes for unpredictability

- DRAM[1]
  - Unstable access latency
- Shared buses between multiple cores

→ Overall system response time unstable

Approach:

[1] Dasari, D.; Akesson, B.; Nelis, V.; Awan, M.A.; Petters, S.M., "Identifying the sources of unpredictability in COTS-based multicore systems," Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on , vol., no., pp.39,48, 19-21 June 2013

# Causes for unpredictability

- DRAM[1]
  - Unstable access latency
- Shared buses between multiple cores

→ Overall system response time unstable

Approach:

- Caches can reduce unpredictability

[1] Dasari, D.; Akesson, B.; Nelis, V.; Awan, M.A.; Petters, S.M., "Identifying the sources of unpredictability in COTS-based multicore systems," Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on , vol., no., pp.39,48, 19-21 June 2013

# Causes for unpredictability

- DRAM[1]
  - Unstable access latency
- Shared buses between multiple cores

→ Overall system response time unstable

Approach:

- Caches can reduce unpredictability
- Can the OS control which data stays in the cache?

[1] Dasari, D.; Akesson, B.; Nelis, V.; Awan, M.A.; Petters, S.M., "Identifying the sources of unpredictability in COTS-based multicore systems," Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on , vol., no., pp.39,48, 19-21 June 2013

# Causes for unpredictability

- DRAM[1]
  - Unstable access latency
- Shared buses between multiple cores

→ Overall system response time unstable

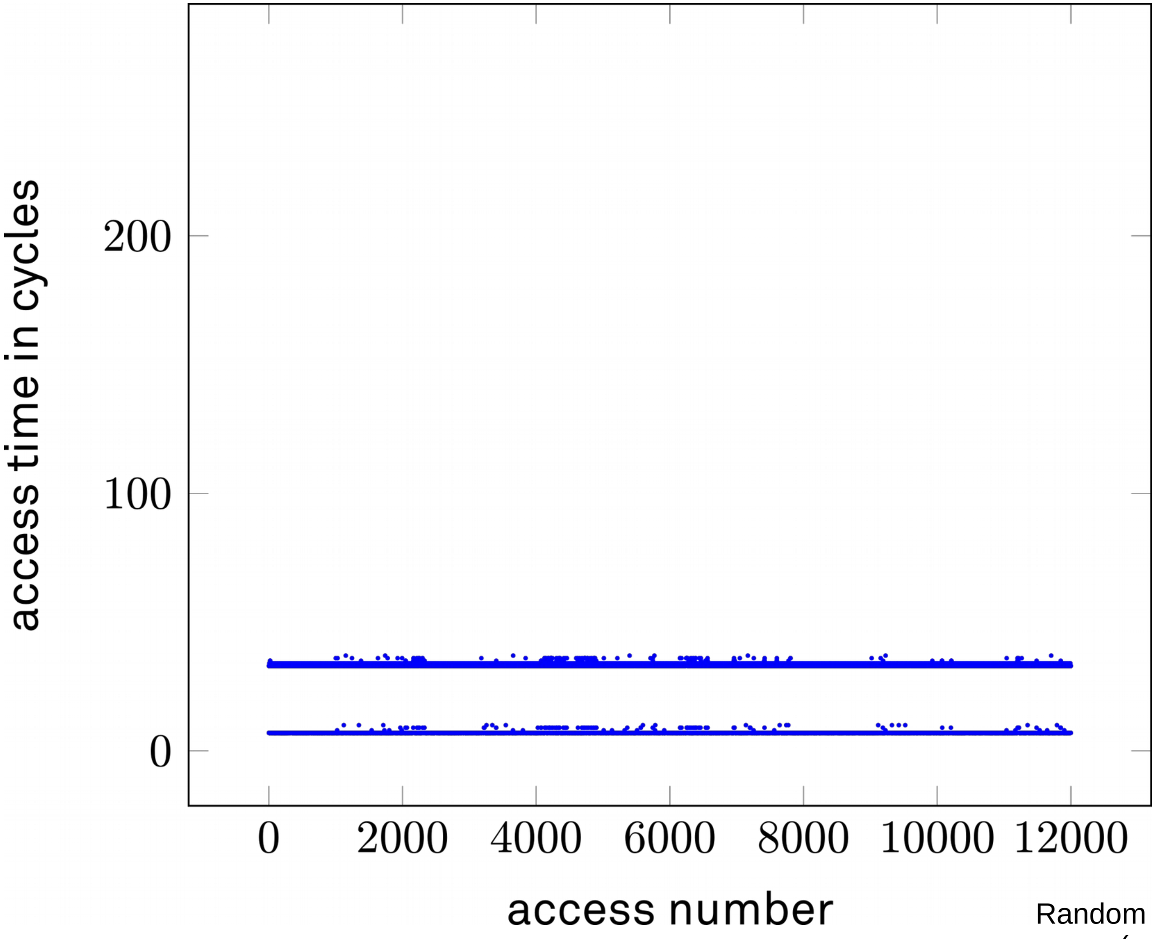
Approach:

- Caches can reduce unpredictability
- Can the OS control which data stays in the cache?

➡ No unexpected cache misses

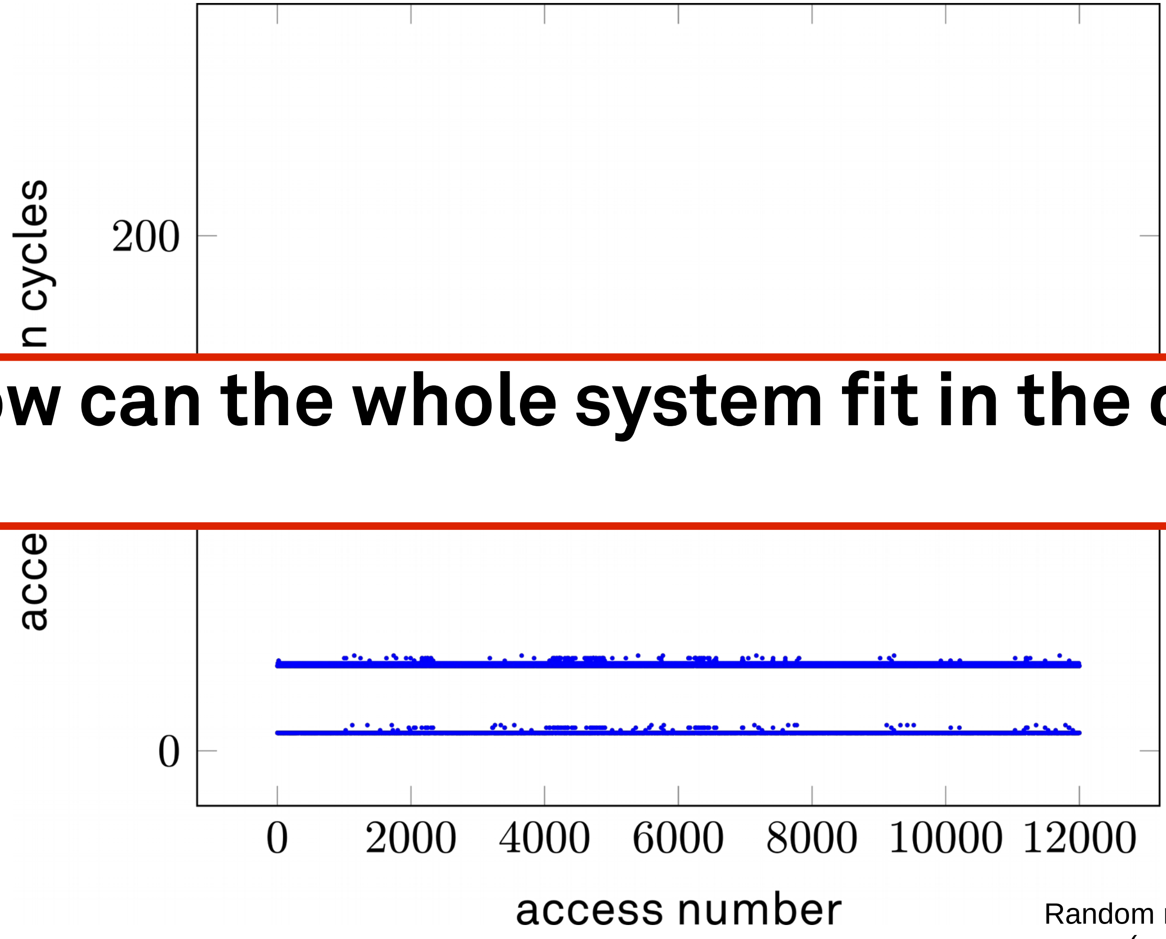
[1] Dasari, D.; Akesson, B.; Nelis, V.; Awan, M.A.; Petters, S.M., "Identifying the sources of unpredictability in COTS-based multicore systems," Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on , vol., no., pp.39,48, 19-21 June 2013

# OS-controlled cache



Random memory access within a 64kB range (prefetched/locked)

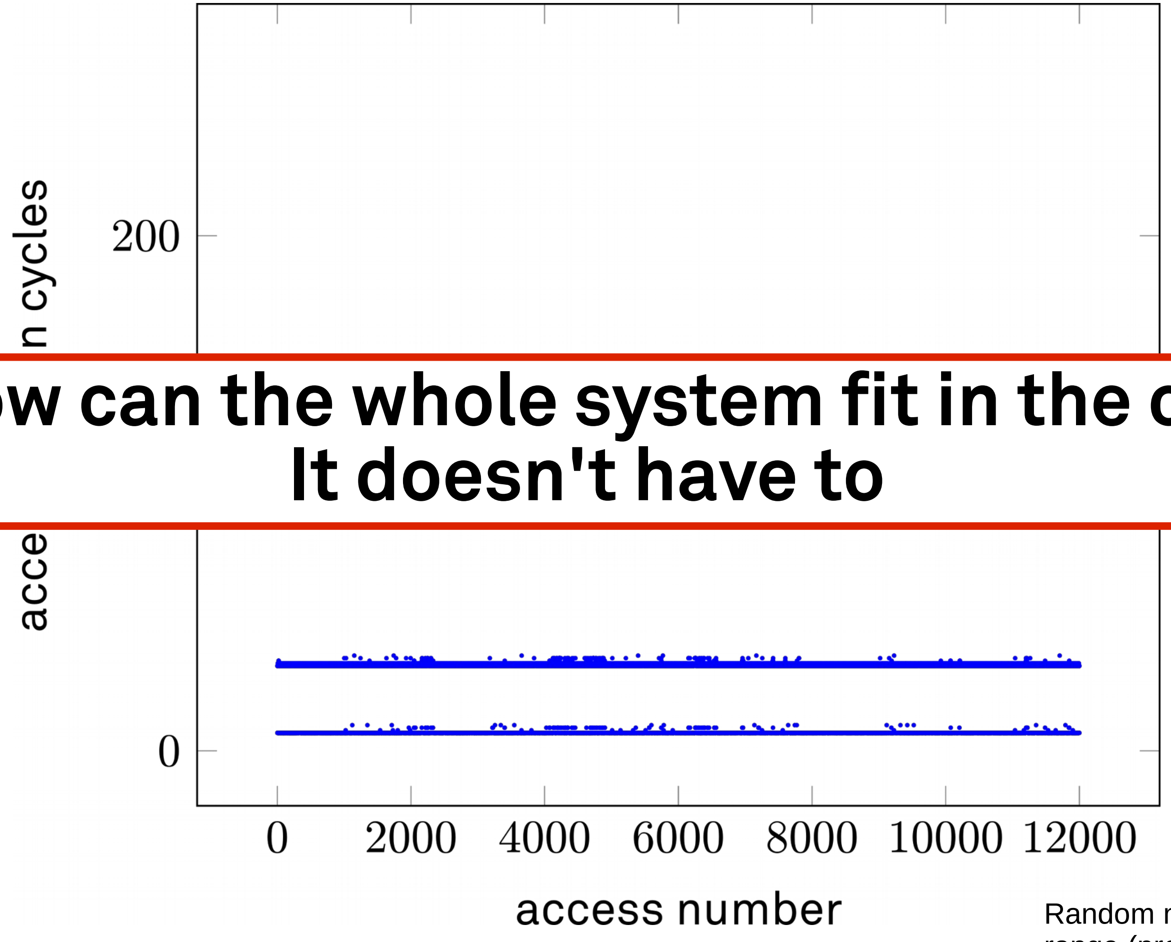
# OS-controlled cache



**But how can the whole system fit in the cache?**

Random memory access within a 64kB range (prefetched/locked)

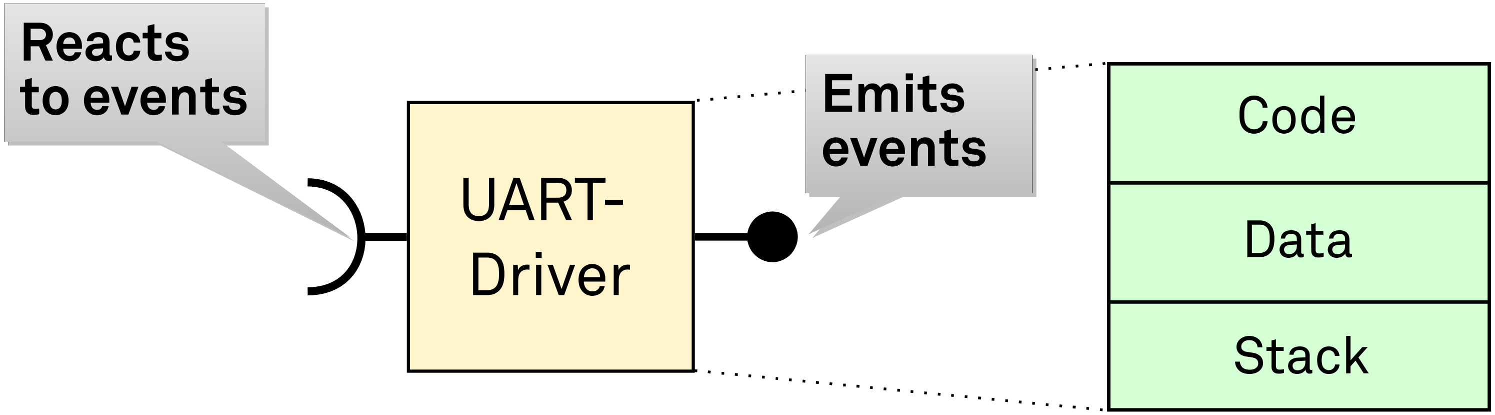
# OS-controlled cache



**But how can the whole system fit in the cache?**
**It doesn't have to**

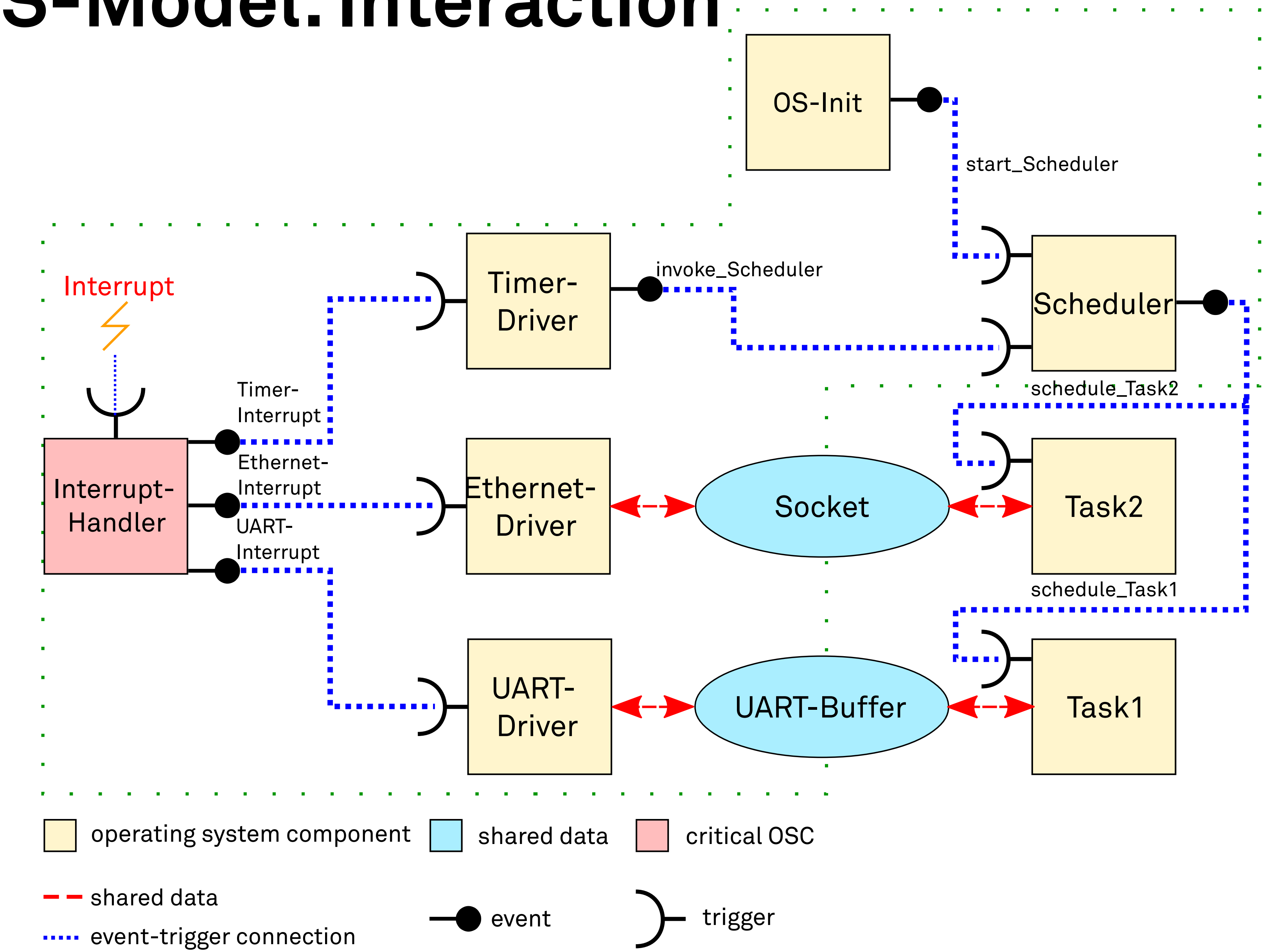
Random memory access within a 64kB range (prefetched/locked)

# OS-Model: Component



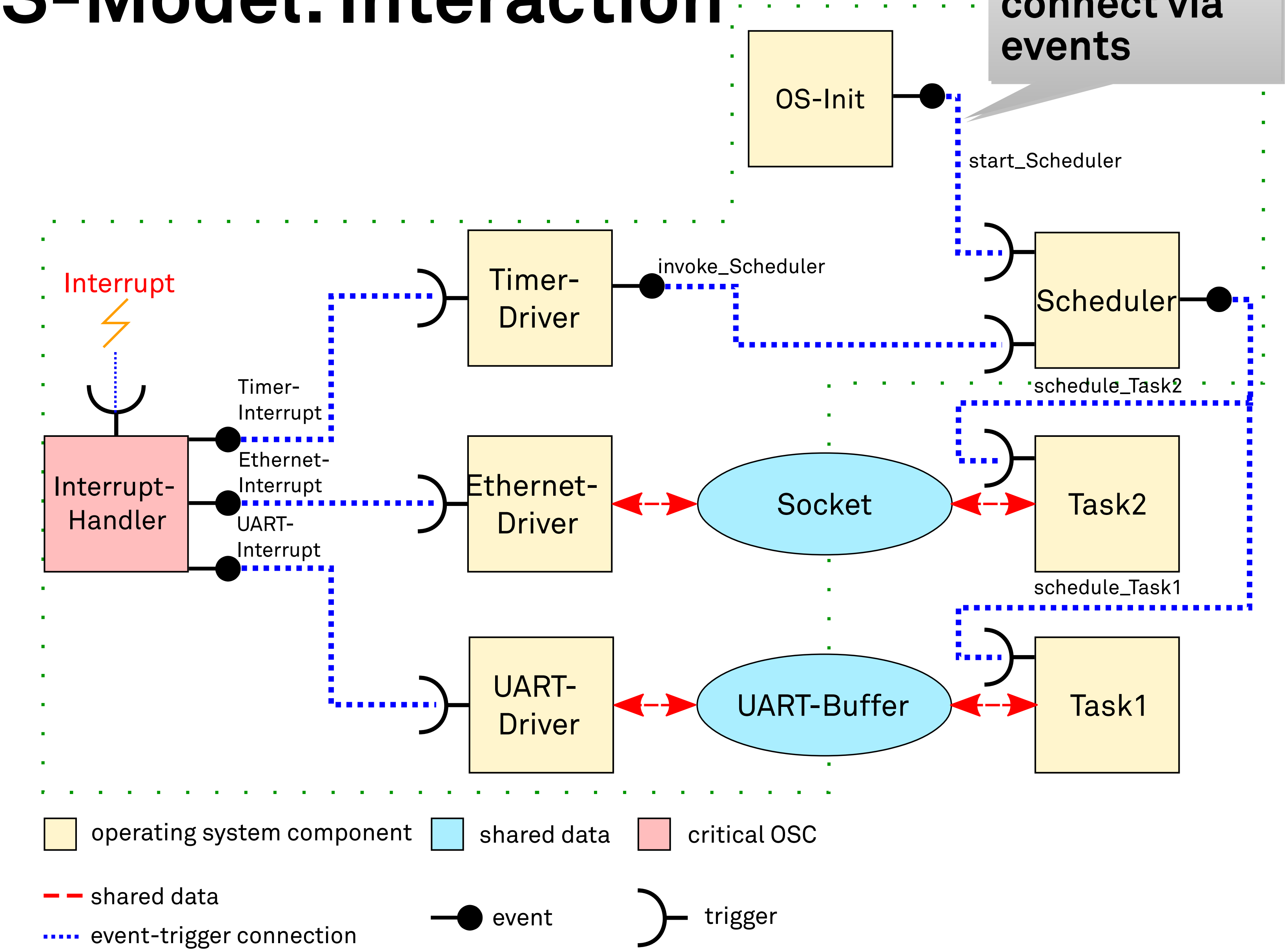
- Small components
- Mostly independent
- No external calls/data accesses (cache misses)
- All necessary data confined

# OS-Model: Interaction

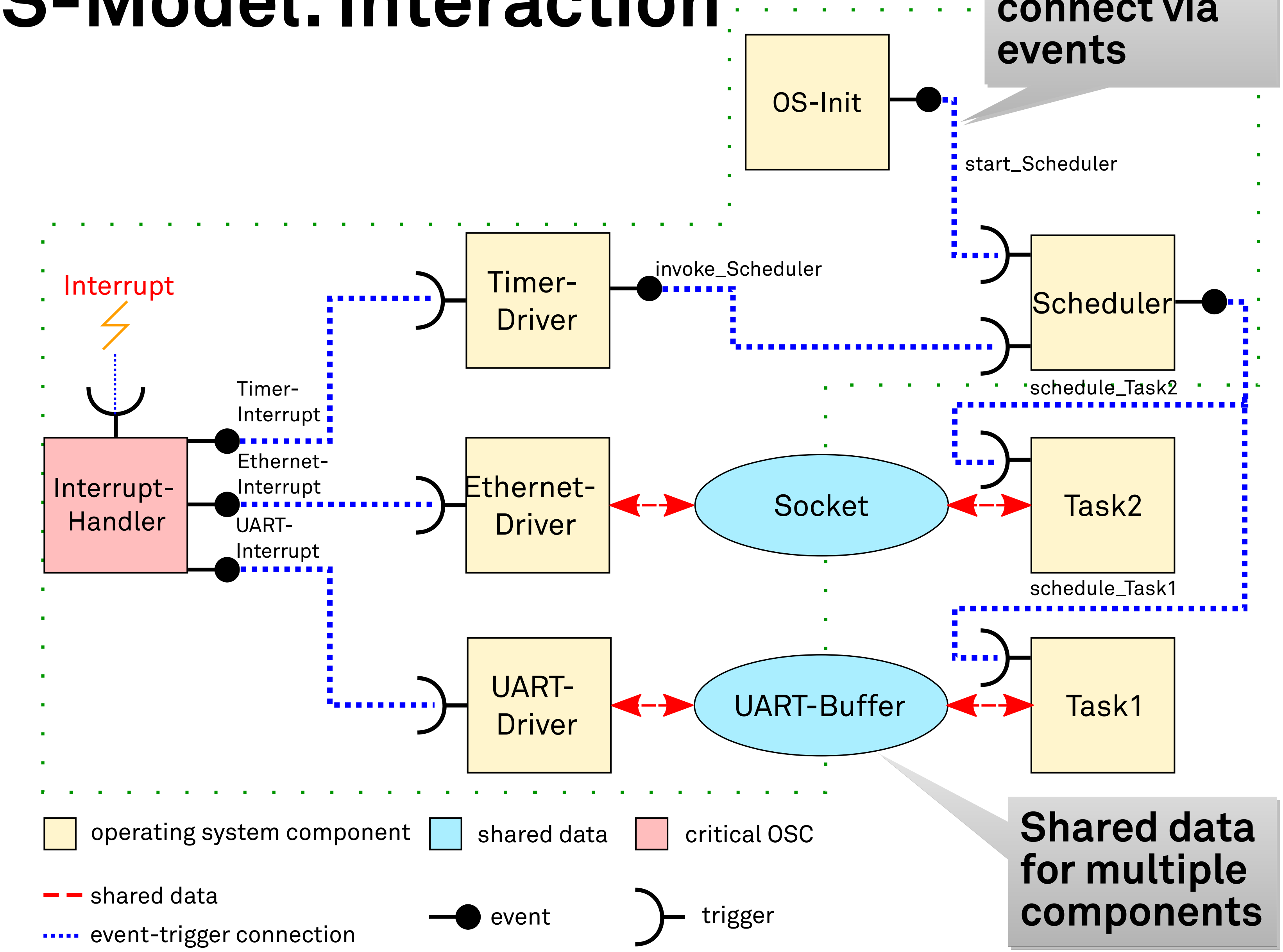


# OS-Model: Interaction

Components connect via events



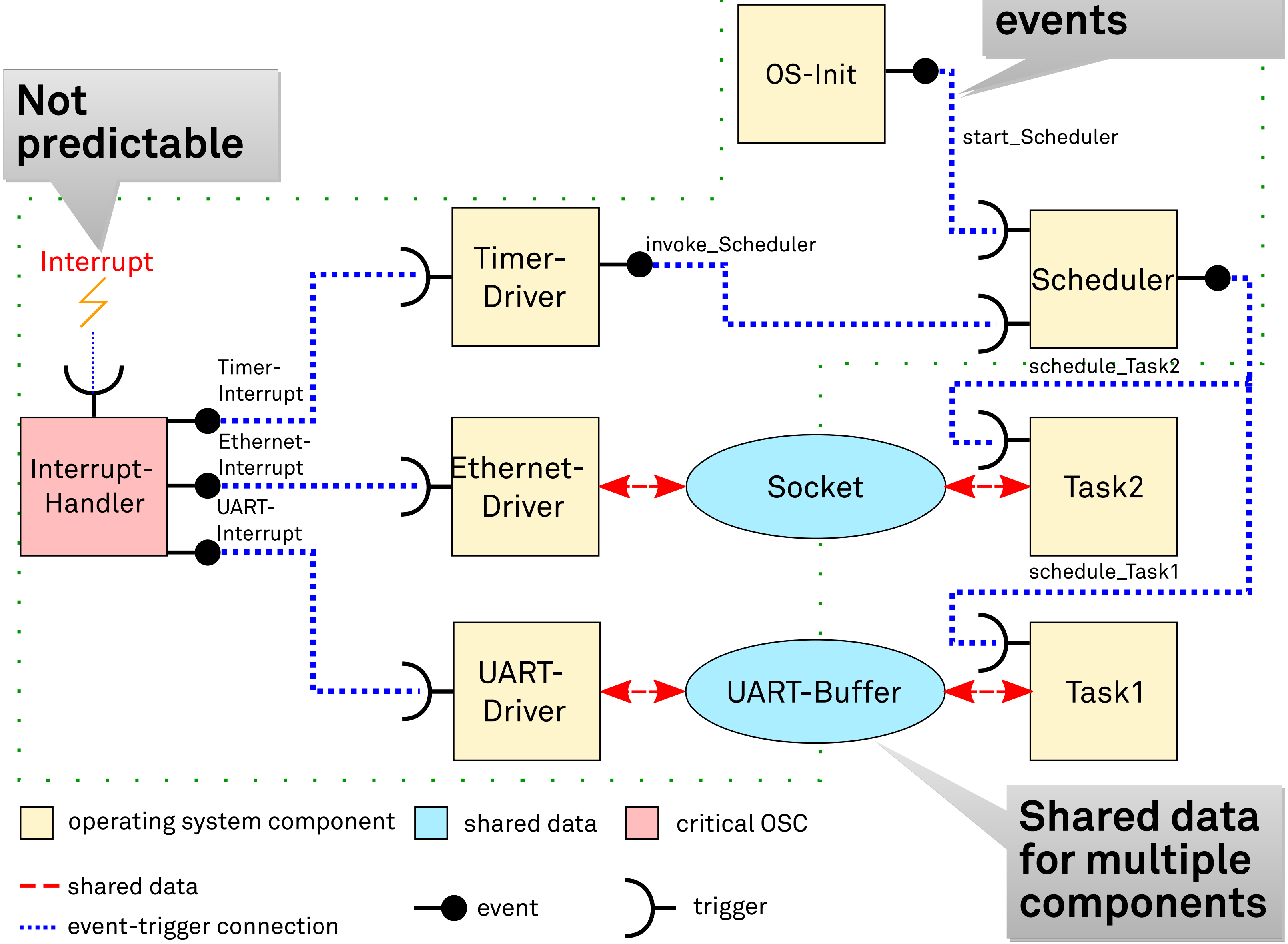
# OS-Model: Interaction



Components connect via events

Shared data for multiple components

# OS-Model: Interaction



Components connect via events

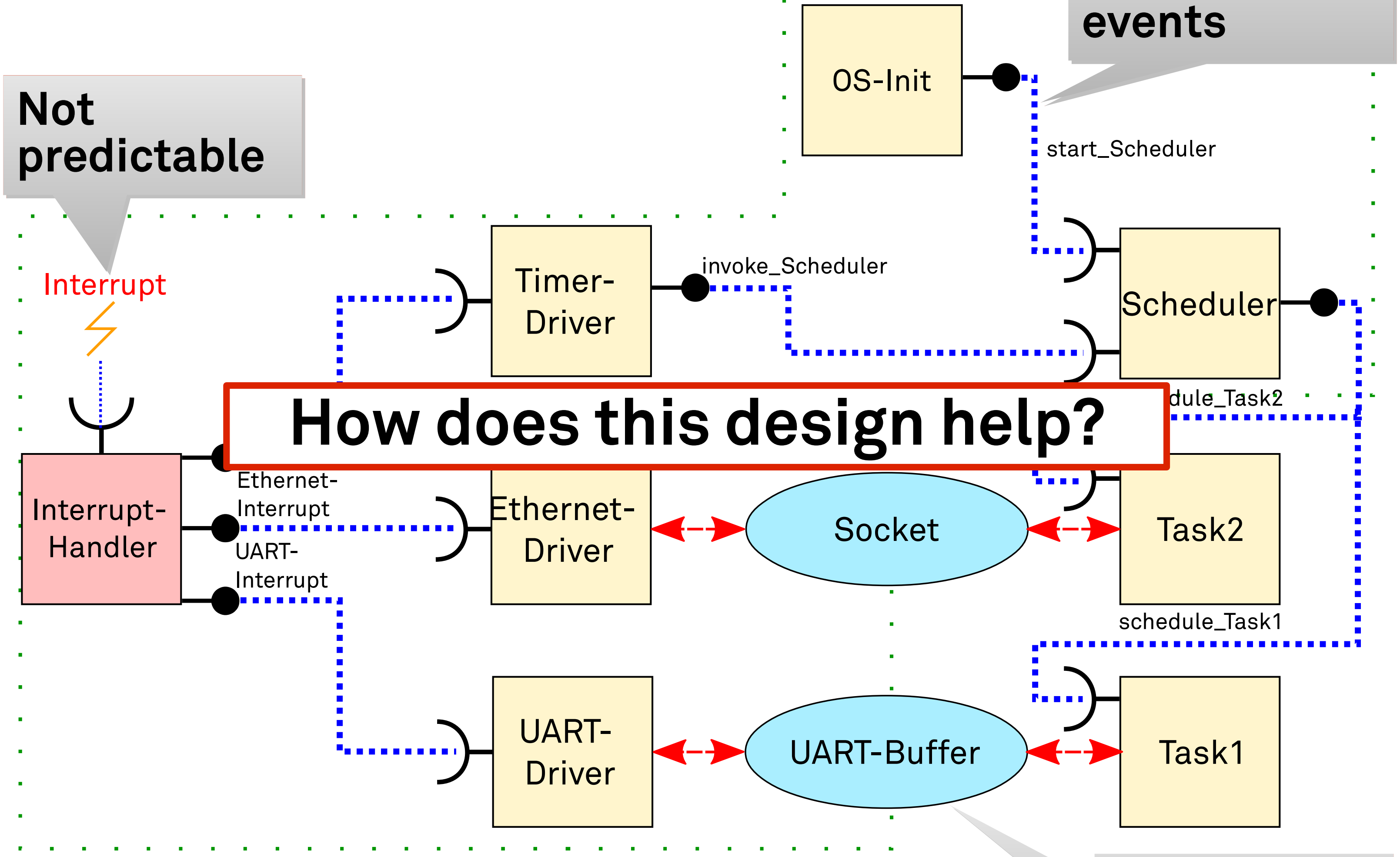
Not predictable

Shared data for multiple components

# OS-Model: Interaction

Components connect via events

Not predictable



**How does this design help?**

Shared data for multiple components

🟨 operating system component 🟦 shared data 🟥 critical OSC

- - shared data

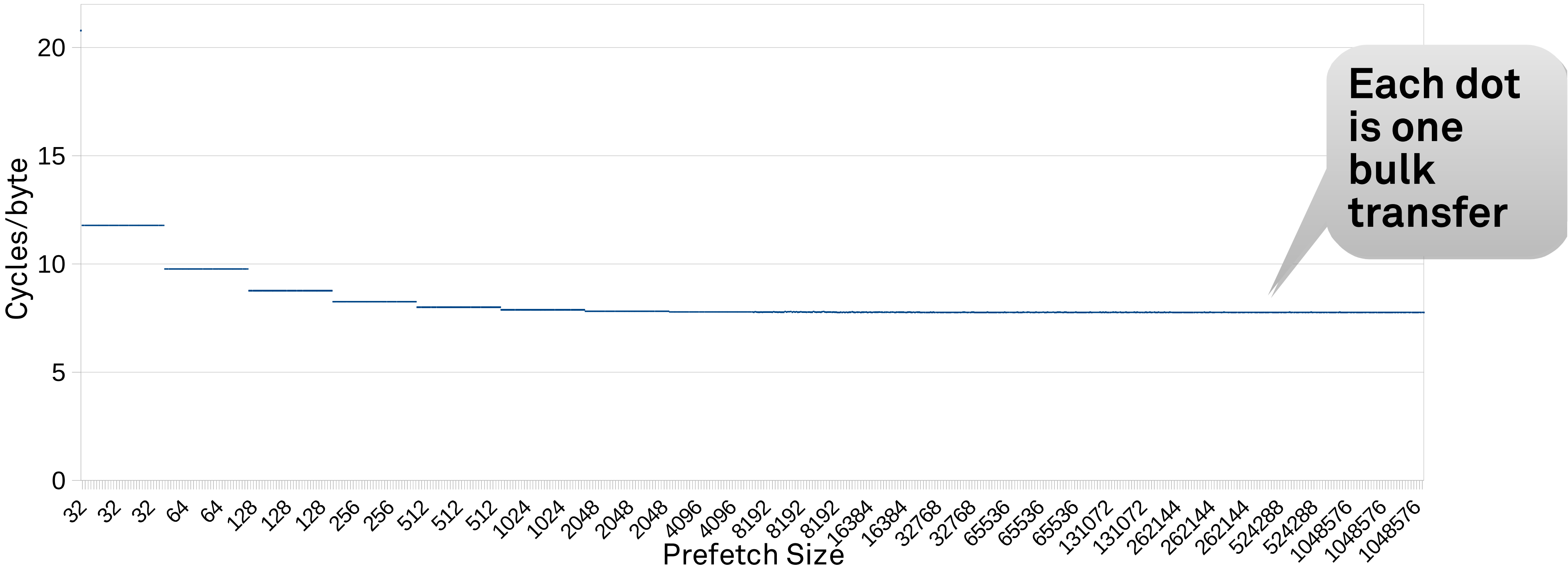
····· event-trigger connection

●— event

)— trigger

# Component handling

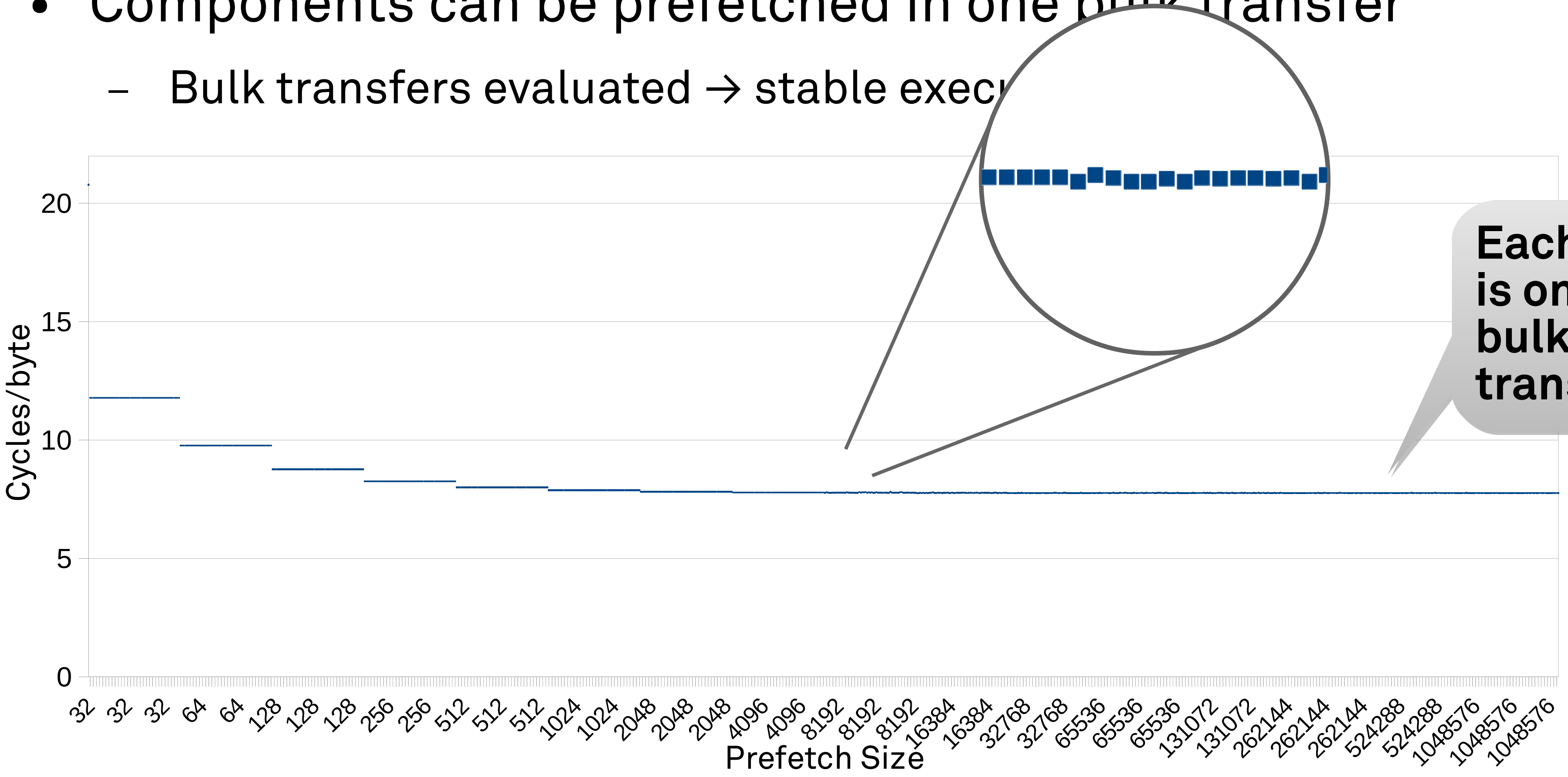
- All data confined to one continuous data block
- Enables complete knowledge over necessary data
- Components can be prefetched in one bulk transfer
  - Bulk transfers evaluated → stable execution times



Each dot is one bulk transfer

# Component handling

- All data confined to one continuous data block
- Enables complete knowledge over necessary data
- Components can be prefetched in one bulk transfer
  - Bulk transfers evaluated → stable exec

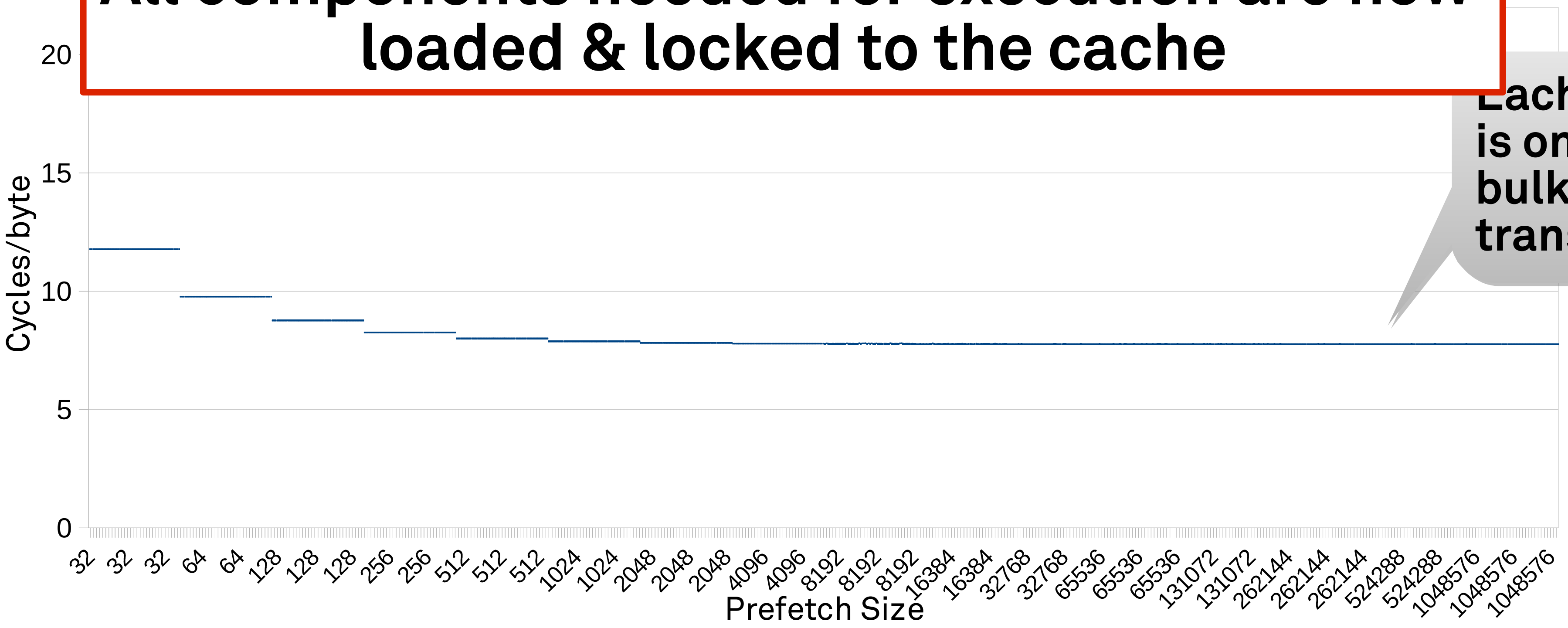


Each dot is one bulk transfer

# Component handling

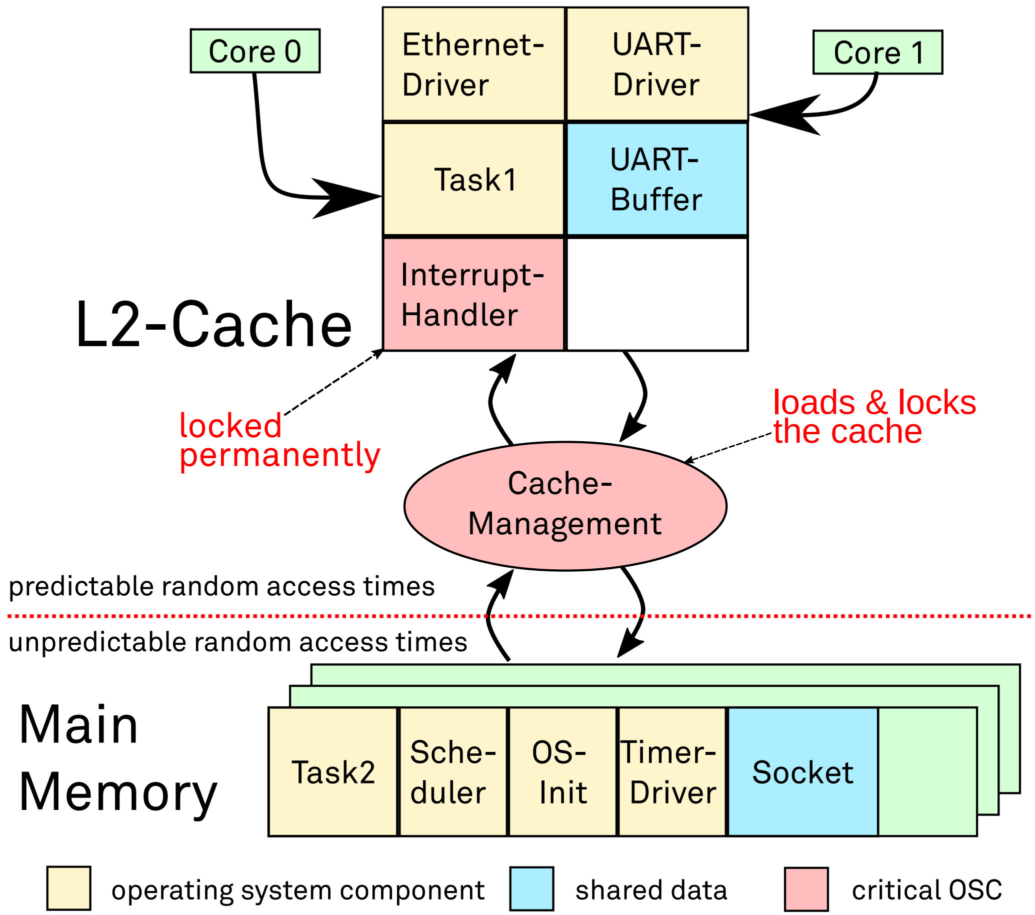
- All data confined to one continuous data block
- Enables complete knowledge over necessary data
- Components can be prefetched in one bulk transfer
  - Bulk transfers evaluated → stable execution times

**All components needed for execution are now loaded & locked to the cache**

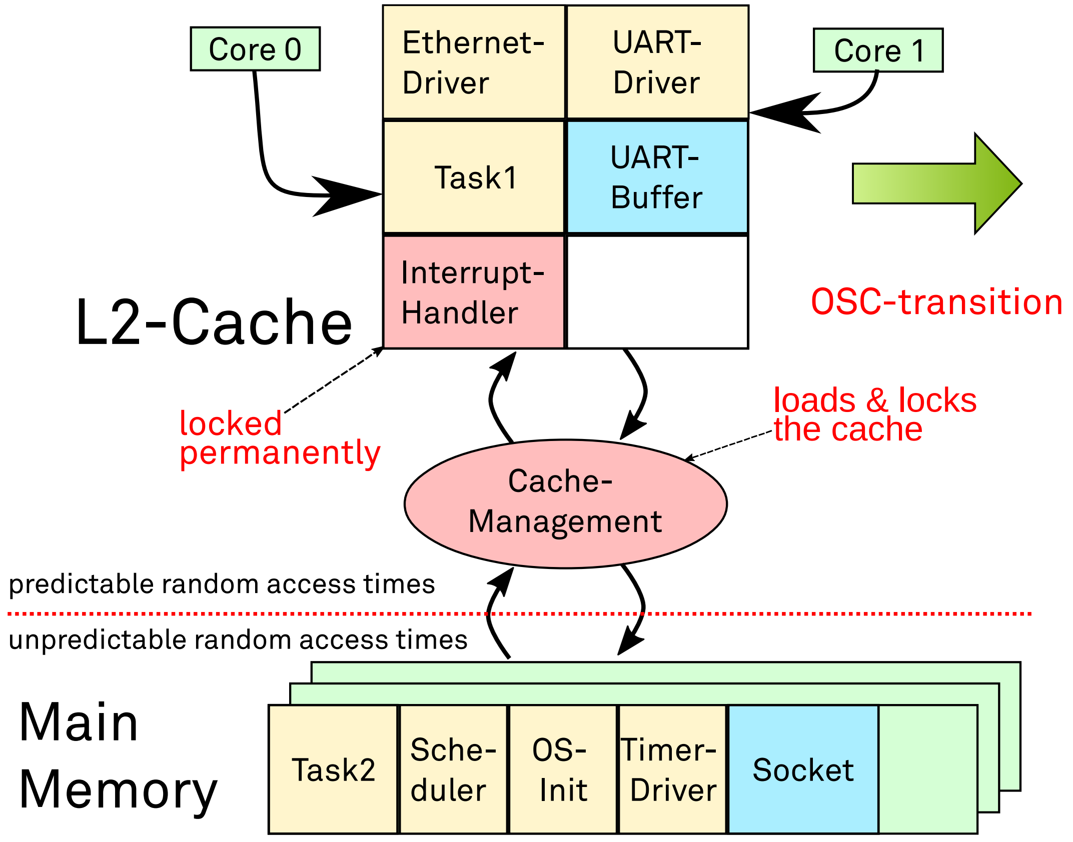


**Each dot is one bulk transfer**

# OS-Transition

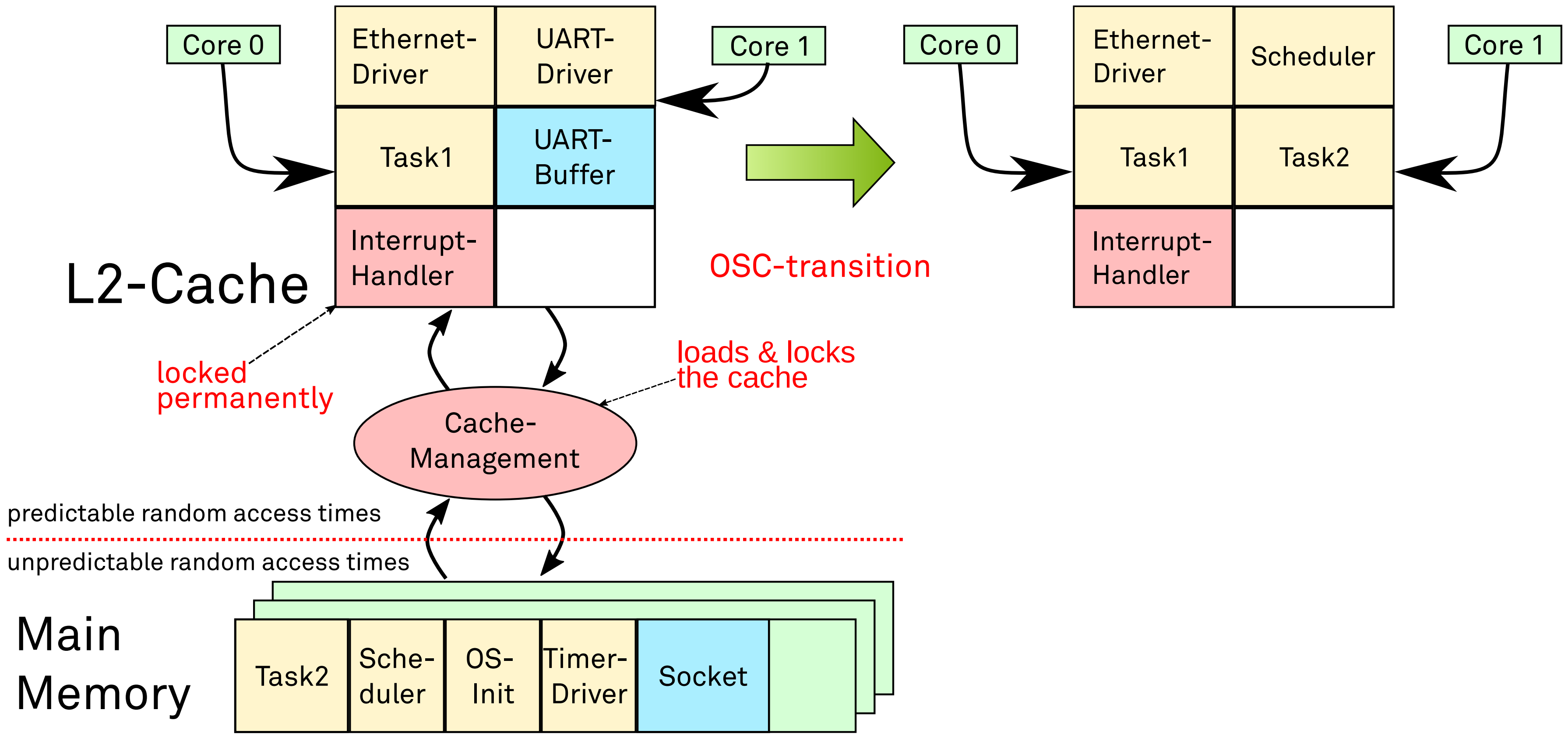


# OS-Transition



🟨 operating system component 🟦 shared data 🟥 critical OSC

# OS-Transition



🟨 operating system component 🟦 shared data 🟥 critical OSC

# Architecture details

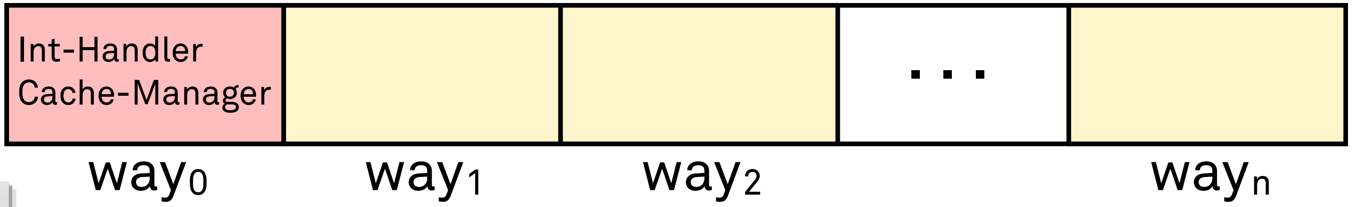
- Current approach needs HW support for
  - Cache prefetching
  - Cache locking
- Current platform: **Dual-core ARM Cortex-A9** (COTS)
- Associative shared level cache
  - 16 cache ways (**64kB** each, **1MB** total)
- Cache management features:
  - Cache **prefetching** of data/code
  - Cache **locking** per cache way & core

# Architecture details

- Current approach needs HW support for
  - Cache prefetching
  - Cache locking
- Current platform: **Dual-core ARM Cortex-A9** (COTS)
- Associative shared level cache
  - 16 cache ways (**64kB** each, **1MB** total)
- Cache management features:
  - Cache **prefetching** of data/code
  - Cache **locking** per cache way & core

**HW allows complete control over the cache content**

# Cache Management

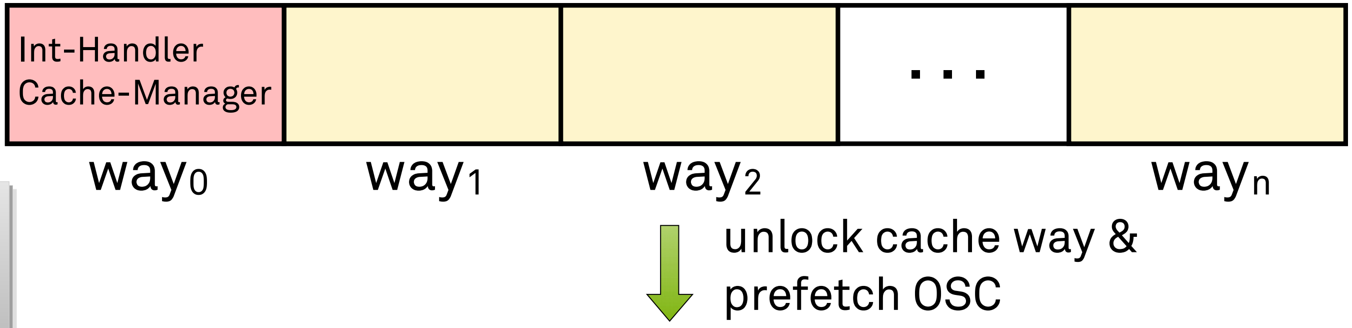


**Components aligned at way-size**

🟥 Permanently locked 🟨 Temporarily locked

🟩 Temporarily unlocked

# Cache Management



Int-Handler Cache-Manager

. . .

way$_0$ way$_1$ way$_2$ way$_n$

unlock cache way & prefetch OSC

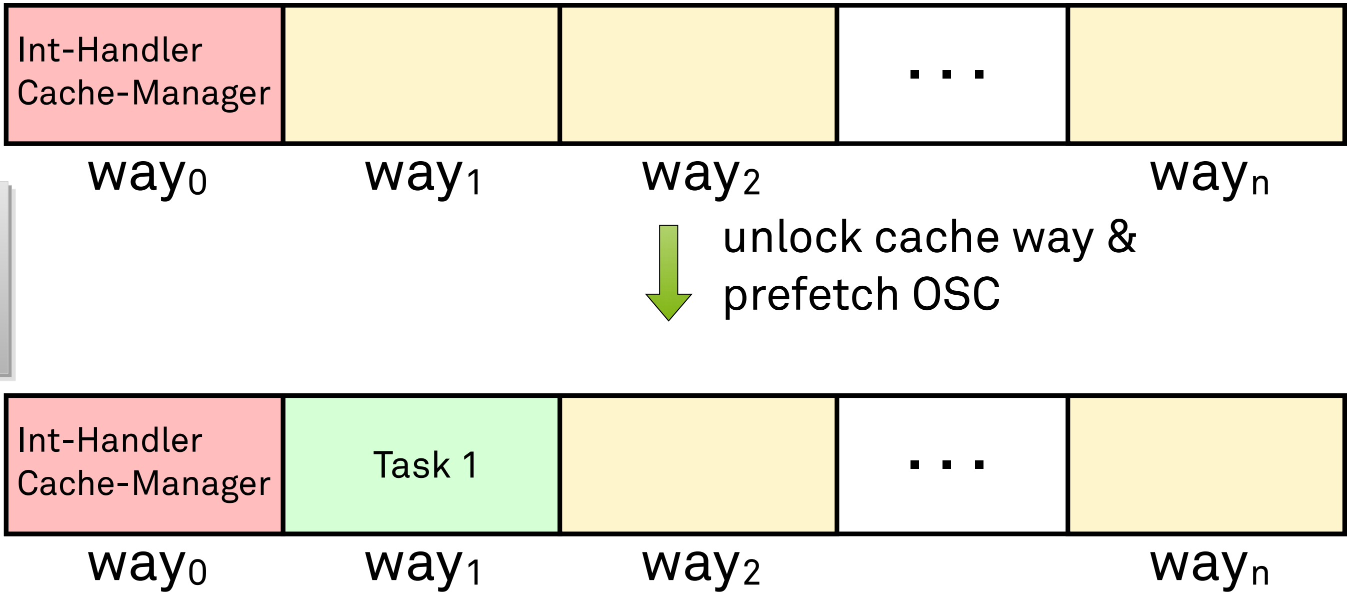
**Components aligned at way-size**

🟥 Permanently locked 🟨 Temporarily locked

🟩 Temporarily unlocked

# Cache Management

**Components aligned at way-size**



Int-Handler Cache-Manager | | | . . . |

way$_0$ way$_1$ way$_2$ way$_n$

unlock cache way & prefetch OSC

Int-Handler Cache-Manager | Task 1 | | . . . |

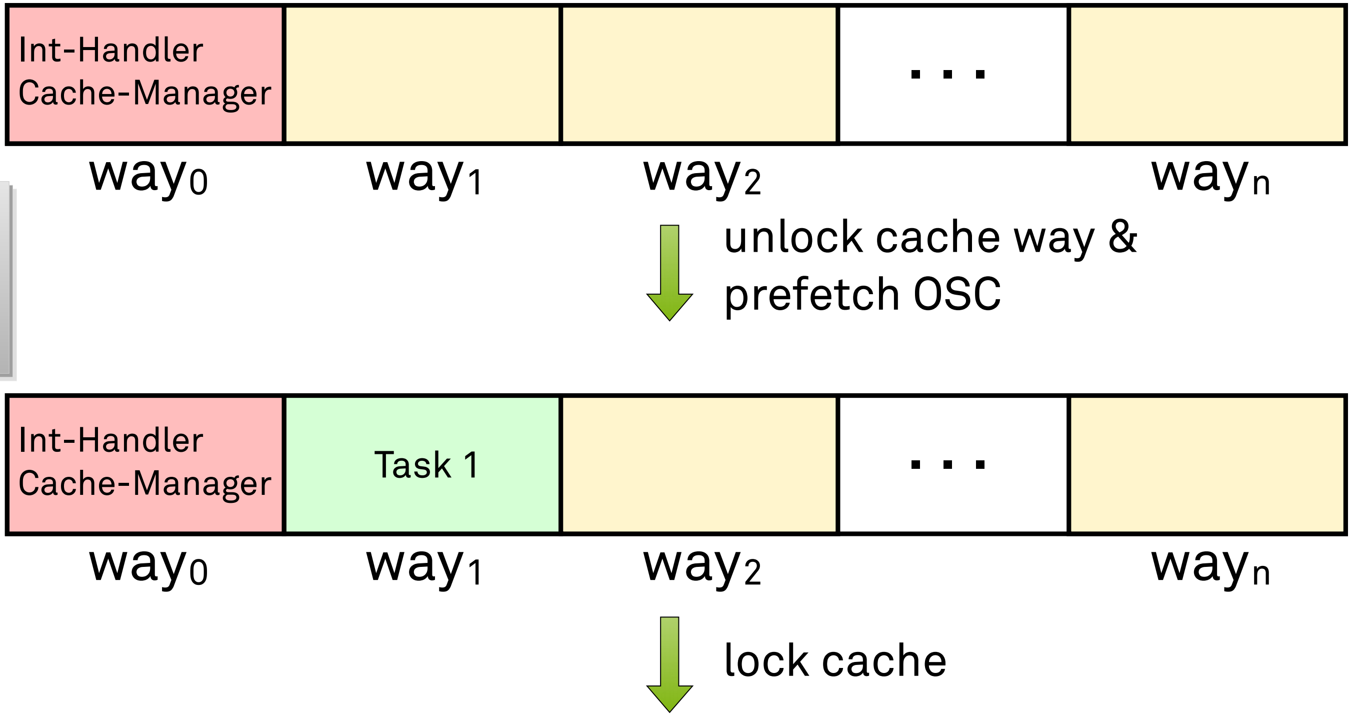
way$_0$ way$_1$ way$_2$ way$_n$

🟥 Permanently locked 🟨 Temporarily locked

🟩 Temporarily unlocked

# Cache Management

**Components aligned at way-size**



Int-Handler
Cache-Manager

. . .

way$_0$ way$_1$ way$_2$ way$_n$

unlock cache way &
prefetch OSC

Int-Handler
Cache-Manager

Task 1

. . .

way$_0$ way$_1$ way$_2$ way$_n$

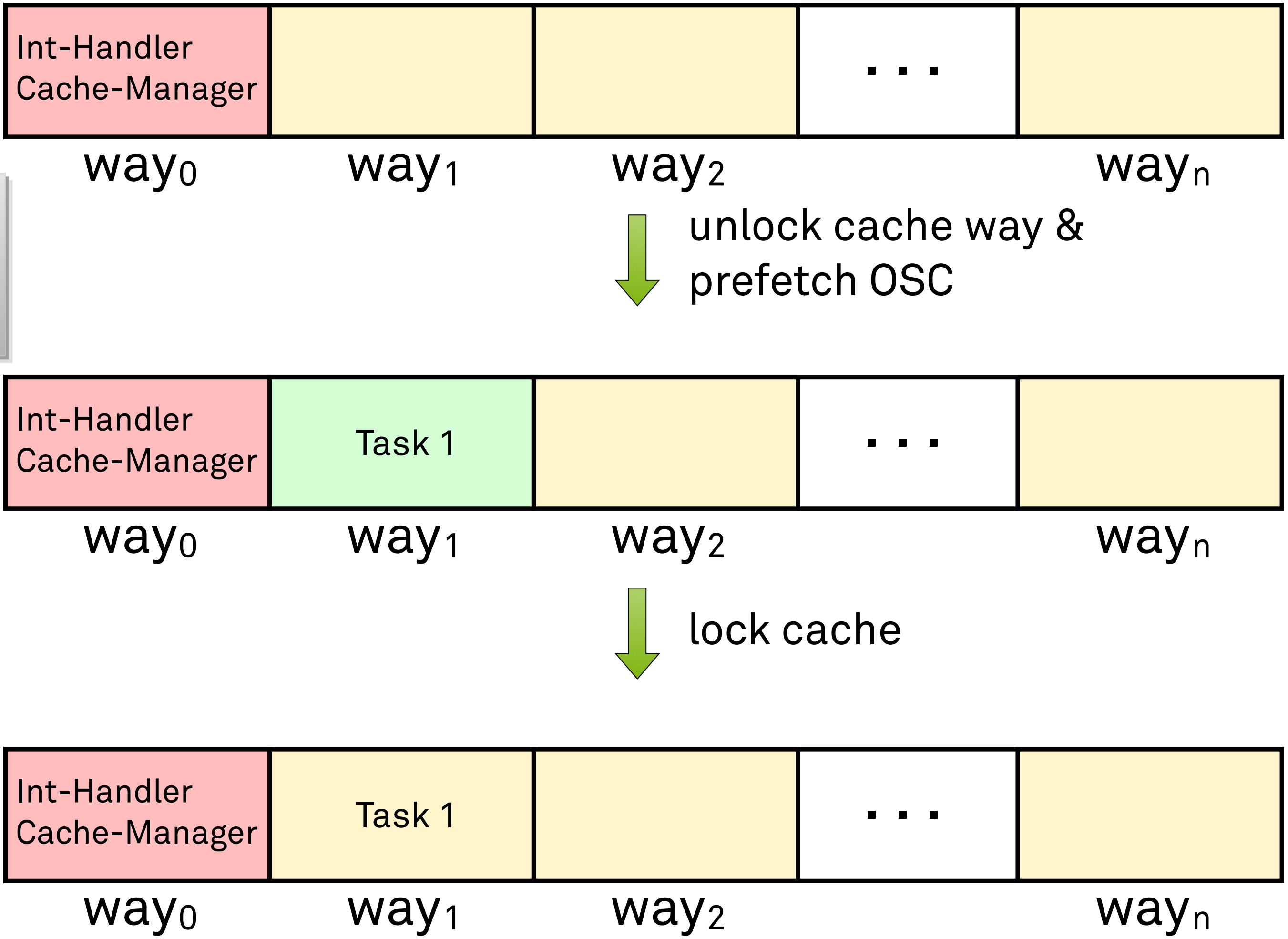
lock cache

🟥 Permanently locked 🟨 Temporarily locked

🟩 Temporarily unlocked

# Cache Management

**Components aligned at way-size**



| way$_0$ | way$_1$ | way$_2$ | | way$_n$ |
|---|---|---|---|---|
| Int-Handler<br>Cache-Manager | | | . . . | |

unlock cache way & prefetch OSC

| way$_0$ | way$_1$ | way$_2$ | | way$_n$ |
|---|---|---|---|---|
| Int-Handler<br>Cache-Manager | Task 1 | | . . . | |

lock cache

| way$_0$ | way$_1$ | way$_2$ | | way$_n$ |
|---|---|---|---|---|
| Int-Handler<br>Cache-Manager | Task 1 | | . . . | |

🟥 Permanently locked 🟨 Temporarily locked

🟩 Temporarily unlocked

# Ongoing & Future Work

- Optimize event scheduling
- Automatic adapation to HW platform
- Eliminate dead code/data prefetching
- Reduce the dependency on hardware cache management
- Compare against other RTOS

# Summary

- Modern COTS-HW unpredictable (DRAM,buses,...)
- Caches hide DRAM-access latency
- Small structured OS proposed
- Components fit in cache

→ Shift random DRAM-access to bulk transfer

→ Predictable access times

# Summary

- Modern COTS-HW unpredictable (DRAM,buses,...)
- Caches hide DRAM-access latency
- Small structured OS proposed
- Components fit in cache

→ Shift random DRAM-access to bulk transfer

→ Predictable access times

Questions?