Neverlast and Beyond

Handling Volatility in NVRAM-centric Operating Systems

GI Fachgruppe Betriebssysteme - Herbsttreffen
September 21, 2021

Christian Eichler[1], Henriette Hofmeier[1], Wolfgang Schröder-Preikschat[2], and Timo Hönig[1]

[1]Ruhr-Universität Bochum (RUB)
[2]Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)

RUHR UNIVERSITÄT BOCHUM

RUB

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

Motivation: Energy-harvesting Edge-computing Systems

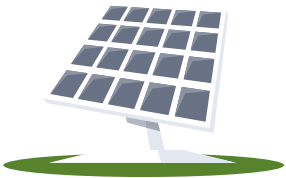
Energy-harvesting Systems...

- Obtain power from the environment (e.g., solar panels)

Motivation: Energy-harvesting Edge-computing Systems

Energy-harvesting Systems...

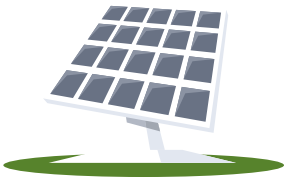
- Obtain power from the environment (e.g., solar panels)
- Depend on environmental conditions (e.g., sun, clouds)



Motivation: Energy-harvesting Edge-computing Systems

Energy-harvesting Systems...

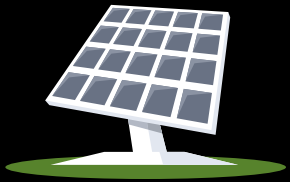
- Obtain power from the environment (e.g., solar panels)
- Depend on environmental conditions (e.g., sun, clouds)
- Often come with limited energy storage (e.g., capacitors)



Motivation: Energy-harvesting Edge-computing Systems

Energy-harvesting Systems...

- Obtain power from the environment (e.g., solar panels)
- Depend on environmental conditions (e.g., sun, clouds)
- Often come with limited energy storage (e.g., capacitors)
- **May lose power supply anytime**

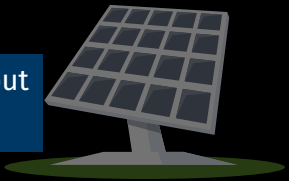


Motivation: Energy-harvesting Edge-computing Systems

Energy-harvesting Systems...

- Obtain power from the environment (e.g., solar panels)
- Depend on environmental conditions (e.g., sun, clouds)
- Often come with limited energy storage (e.g., capacitors)
- **May lose power supply anytime**

$\rightarrow$ Intermittently-powered systems may run out of power, but **must not lose system state**



Motivation: Non-volatile Memory

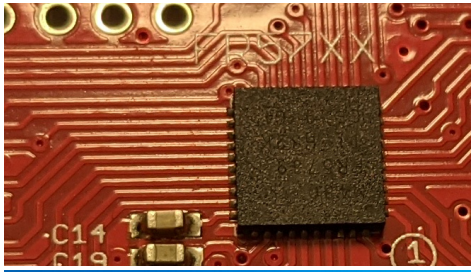
Non-volatile memory retains data on power outage

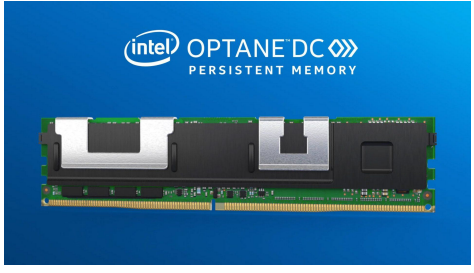
- \+ Ideal solution for energy-harvesting applications
- \+ Can be used as drop-in replacement for conventional volatile main memory

Motivation: Non-volatile Memory

Non-volatile memory retains data on power outage

- Ideal solution for energy-harvesting applications
- Can be used as drop-in replacement for conventional volatile main memory
- Commercially available





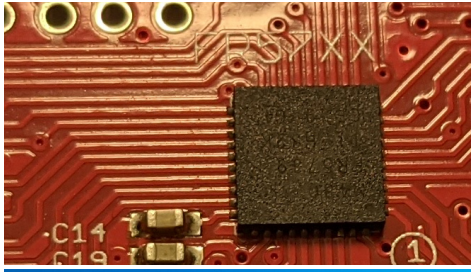
Motivation: Non-volatile Memory

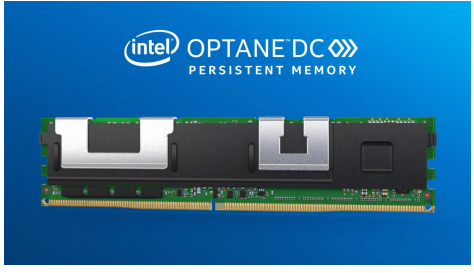
Non-volatile memory retains data on power outage

- \+ Ideal solution for energy-harvesting applications
- \+ Can be used as drop-in replacement for conventional volatile main memory
- \+ Commercially available

But some volatile components remain:

- – CPU registers & caches
- – Hardware configuration

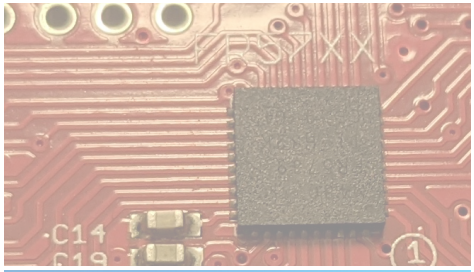




Motivation: Non-volatile Memory

Non-volatile memory retains data on power outage

+ Ideal solution for energy-harvesting applications
+ Can be used as drop-in replacement for conventional volatile main memory
+ Commercially available



But some volatile components remain:

− CPU registers & caches
− Hardware co

→ **Data in volatile components** must be taken into account when persisting the system state

Motivation: Consistency is Key

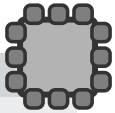
Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.*
In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
```



Sensor Configuration

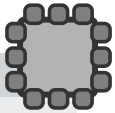
Oversampling:	x1
Offset:	+0

[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.*
In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
write(sensor, "oversampling x16");
```

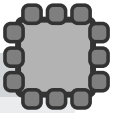


[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.*
In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
write(sensor, "oversampling x16");
```
ϟ power failure



Sensor Configuration

Oversampling: -

Offset: -

[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.* In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

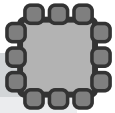
Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
write(sensor, "oversampling x16");
```

power failure

Bootup after power failure



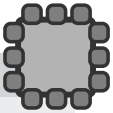
[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.*
In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
write(sensor, "oversampling x16");
```
ϟ power failure

Bootup after power failure
```
write(sensor, "offset +2");
```



Sensor Configuration

Oversampling:	x1
Offset:	**+2**

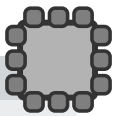
[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.*
In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
write(sensor, "oversampling x16");
```
⚡ power failure

Bootup after power failure
```
write(sensor, "offset +2");
```

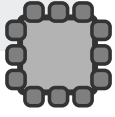


Sensor Configuration

Oversampling:	x1
Offset:	**+2**

Expected Configuration

Oversampling:	**x16**
Offset:	**+2**



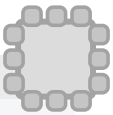
[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.* In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Motivation: Consistency is Key

Losing not-yet persisted volatile state leads to **inconsistent** system states [o].

```
int sensor = open(TEMPERATURE_SENSOR);
write(sensor, "oversampling x16");
```
ϟ power failure

Bootup after power failure
```
write(sensor, "offset +2");
```

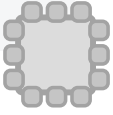


Sensor Configuration

Oversampling: x1
Offset: **+2**

Expected Configuration

Oversampling: **x16**



→ Losing **volatile state** eventually leads to **inconsistencies**

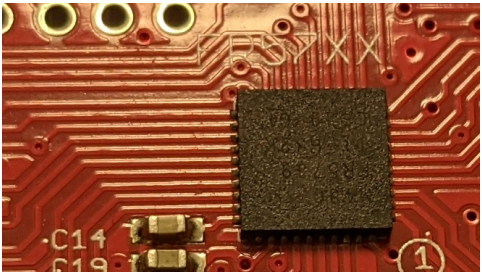
[o] Benjamin Ransford and Brandon Lucia. *Nonvolatile memory is a broken time machine.* In: Proceedings of the ACM Workshop on Memory Systems Performance and Correctness (MSPC '14)

Design & Implementation of Neverlast

Neverlast: Handling Device and System Consistency

Consistency comprises...

- Application and OS data
- Volatile cache and register contents
- But also **volatile hardware configuration**



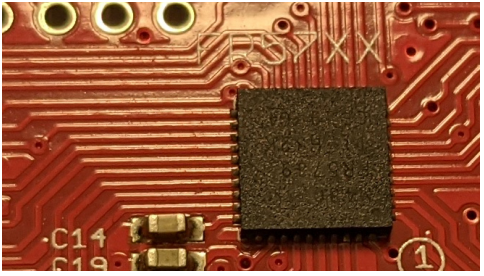
Therefore, Neverlast...

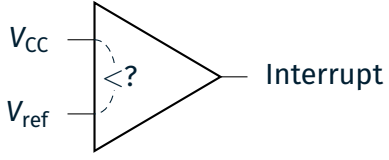
1. Minimizes the volatile state by running in NVRAM
2. Saves registers & caches on power outage
3. Traces device-configuration syscalls

Neverlast: Detecting & Handling Power Outages[1]

Detecting Power Outages

1. Voltage comparator monitors V_{CC} and interrupts the CPU when V_{CC} falls below threshold



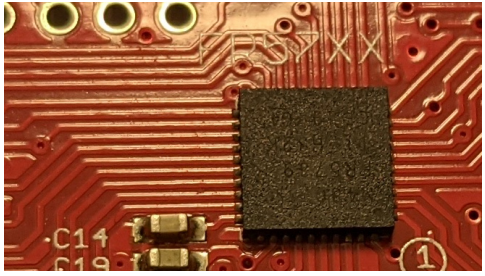


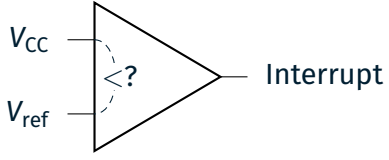
[1] *Neverlast: Towards the Design and Implementation of the NVM-based Everlasting Operating System.* Christian Eichler, Henriette Hofmeier, Stefan Reif, Timo Hönig, Jörg Nolte, and Wolfgang Schröder-Preikschat. In: Proceedings of the 54th Hawai'i International Conference on System Sciences (HICSS-54)

Neverlast: Detecting & Handling Power Outages[1]

Detecting Power Outages

1. Voltage comparator monitors V_{CC} and interrupts the CPU when V_{CC} falls below threshold
2. ISR stores remaining volatile data (registers)
3. Marks stored system state as valid
4. Stops system execution



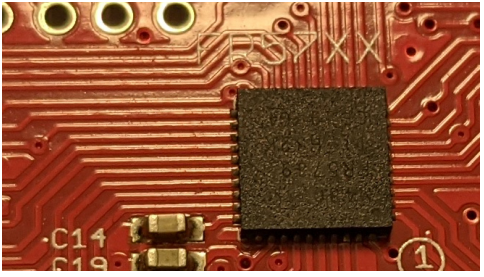


[1] *Neverlast: Towards the Design and Implementation of the NVM-based Everlasting Operating System.* Christian Eichler, Henriette Hofmeier, Stefan Reif, Timo Hönig, Jörg Nolte, and Wolfgang Schröder-Preikschat. In: Proceedings of the 54th Hawai'i International Conference on System Sciences (HICSS-54)

Neverlast: Detecting & Handling Power Outages[1]

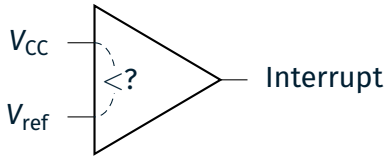
Detecting Power Outages

1. Voltage comparator monitors V_{CC} and interrupts the CPU when V_{CC} falls below threshold
2. ISR stores remaining volatile data (registers)
3. Marks stored system state as valid
4. Stops system execution



Recovering from Power Outages

1. Check stored system state for validity
2. Restore volatile state
3. Continue execution



[1] *Neverlast: Towards the Design and Implementation of the NVM-based Everlasting Operating System.* Christian Eichler, Henriette Hofmeier, Stefan Reif, Timo Hönig, Jörg Nolte, and Wolfgang Schröder-Preikschat. In: Proceedings of the 54th Hawai'i International Conference on System Sciences (HICSS-54)

Neverlast: Tracing Device-configuration System Calls[2]

Interrupt-based Mechanism Problematic for Peripheral Devices

- Unknown device state
- Configuration not readable
- Slow communication bus
- High power consumption

Neverlast: Tracing Device-configuration System Calls[2]

Interrupt-based Mechanism Problematic for Peripheral Devices

- Unknown device state
- Configuration not readable
- Slow communication bus
- High power consumption

Neverlast's Approach: Trace System Calls

- Transaction-like behavior (via `replay_start` and `replay_stop`)
- Log system calls (including parameters) to persistent log
- Replay log after power outage

[2] *Neverlast: an NVM-centric Operating System for Persistent Edge Systems.*
Christian Eichler, Henriette Hofmeier, Stefan Reif, Timo Hönig, Jörg Nolte, and Wolfgang Schröder-Preikschat.
In: Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2021)

Neverlast: Tracing Device-configuration System Calls[2]

Interrupt-based Mechanism Problematic for Peripheral Devices

- Unknown device state
- Configuration not readable
- Slow communication bus
- High power consumption

Neverlast's Approach: Trace System Calls

- Transaction-like behavior (via `replay_start` and `replay_stop`)
- Log system calls (including parameters) to persistent log
- Replay log after power outage

→ Use **syscall-based transaction mechanism** for peripheral devices

[2] *Neverlast: an NVM-centric Operating System for Persistent Edge Systems.*
Christian Eichler, Henriette Hofmeier, Stefan Reif, Timo Hönig, Jörg Nolte, and Wolfgang Schröder-Preikschat.
In: Proceedings of the 12th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys 2021)

Neverlast: Tracing Device-configuration Syscalls

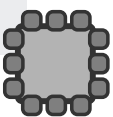
```
1 void app_main() {
2   int sen = open(TEMP_SENSOR)
3   replay_start(sen, 1)
4   write(sen, "oversampling x16")

5   write(sen, "offset +2")
6   replay_stop(sen)
7   /*... */
8 }
```

Neverlast: Tracing Device-configuration Syscalls

```
1 void app_main() {
2   int sen = open(TEMP_SENSOR)
3   replay_start(sen, 1)
4   write(sen, "oversampling x16")
          power-failure interrupt
5   write(sen, "offset +2")
6   replay_stop(sen)
7   /*... */
8 }
```

Sensor Configuration

Oversampling: **x16** Offset: +0



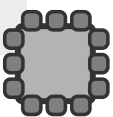
Neverlast: Tracing Device-configuration Syscalls

```
1 void app_main() {
2   int sen = open(TEMP_SENSOR)
3   replay_start(sen, 1)
4   write(sen, "oversampling x16")
          power-failure interrupt
5   write(sen, "offset +2")
6   replay_stop(sen)
7   /*... */
8 }
```

```
//store volatile registers
pushm.a #12, r15
poweroff()
```

Sensor Configuration

Oversampling: - Offset: -



Neverlast: Tracing Device-configuration Syscalls

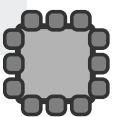
```
1 void app_main() {
2   int sen = open(TEMP_SENSOR)
3   replay_start(sen, 1)
4   write(sen, "oversampling x16")
          power-failure interrupt
5   write(sen, "offset +2")
6   replay_stop(sen)
7   /*... */
8 }
```

```
//store volatile registers
pushm.a #12, r15
poweroff()
```

```
//on power recovery
int sen = open(TEMP_SENSOR)
write(sen, "oversampling x16")

popm.a #12, r15
return
```

Sensor Configuration

Oversampling: **x16** Offset: +0



Neverlast: Tracing Device-configuration Syscalls

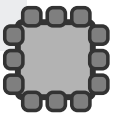
```
1 void app_main() {
2   int sen = open(TEMP_SENSOR)
3   replay_start(sen, 1)
4   write(sen, "oversampling x16")
          power-failure interrupt
5   write(sen, "offset +2")
6   replay_stop(sen)
7   /*... */
8 }
```

```
//store volatile registers
pushm.a #12, r15
poweroff()
```

```
//on power recovery
int sen = open(TEMP_SENSOR)
write(sen, "oversampling x16")

popm.a #12, r15
return
```

Sensor Configuration

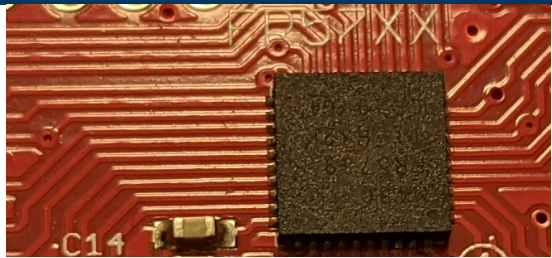
Oversampling: **x16** Offset: **+2**

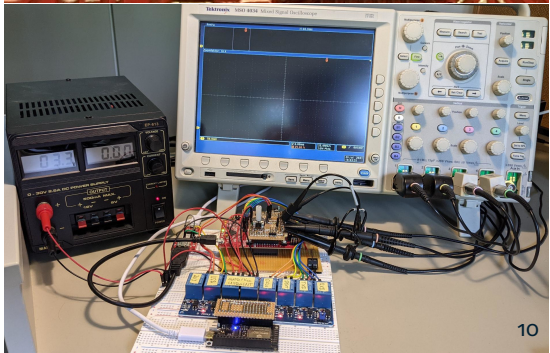


Evaluation

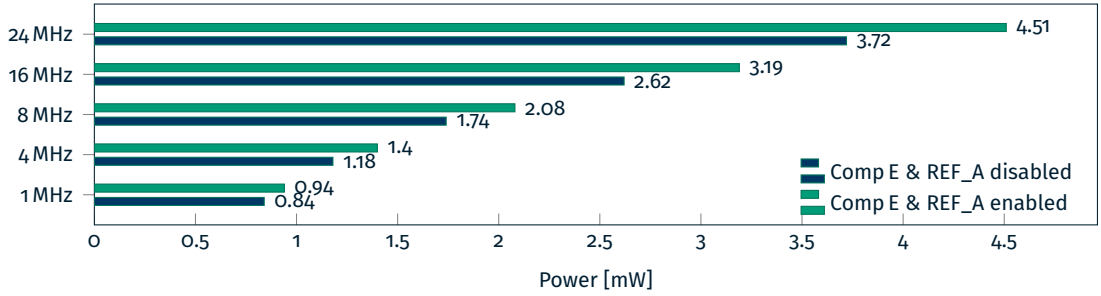
Evaluation - Setup

- MSP-EXP430FR5994 LaunchPad Development Kit
  - 16 bit RISC processor, up to 24 MHz
  - 8 KB of SRAM
  - 256 KB of FRAM with 2-way-2-line read cache (2x 8 Byte)
- Remote-controlled via relais card
- Shunt-based energy measurement Tektronix MSO 4034

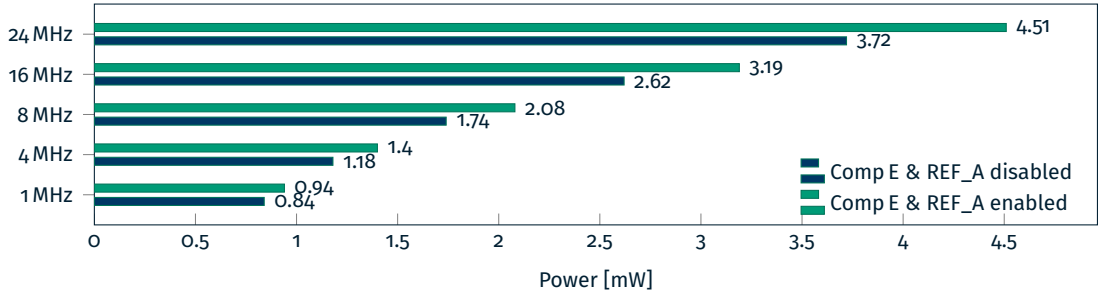


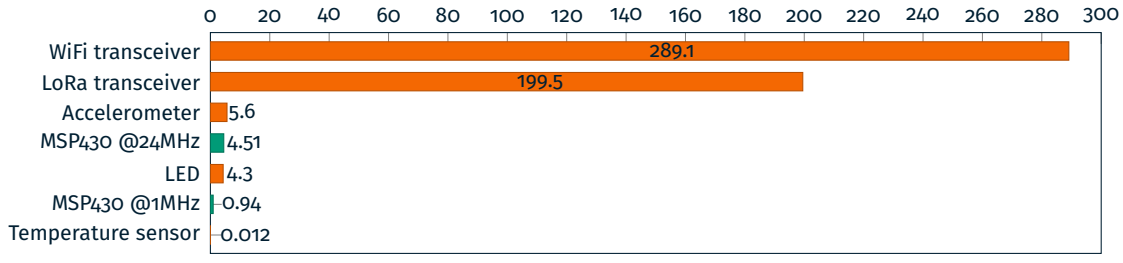


Evaluation - Power Consumption by Supply Monitoring

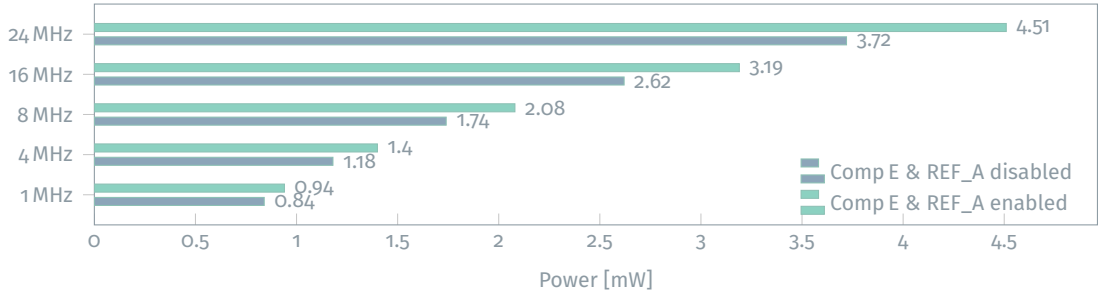


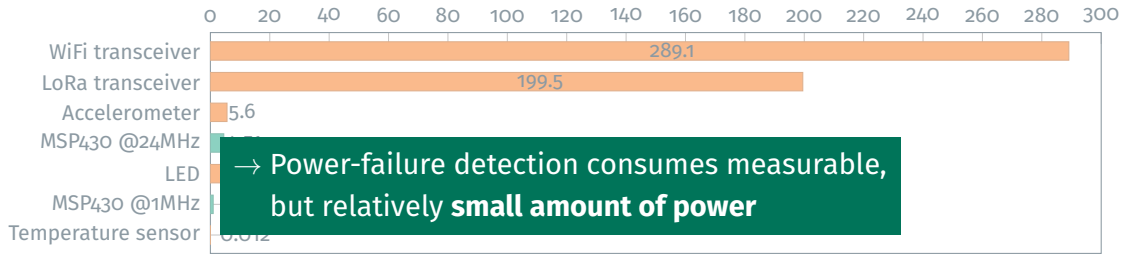
Evaluation - Power Consumption by Supply Monitoring





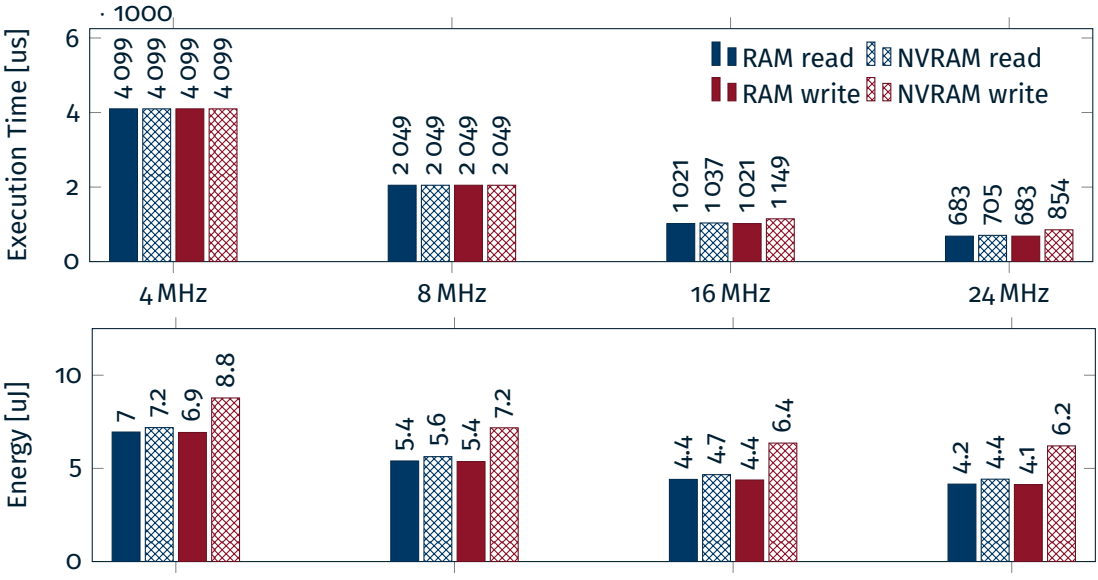
Evaluation - Power Consumption by Supply Monitoring



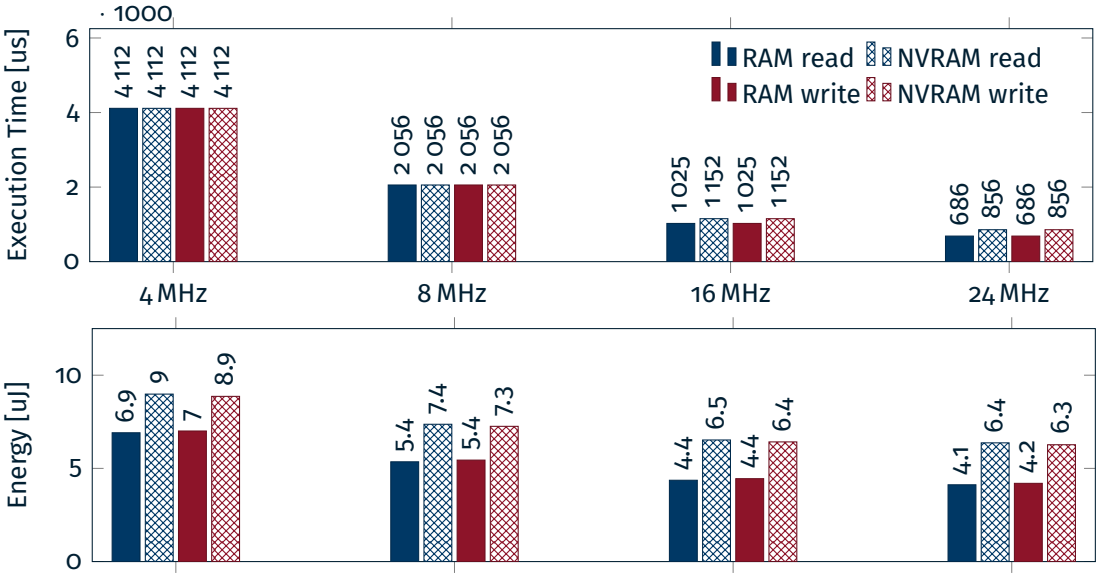


→ Power-failure detection consumes measurable, but relatively **small amount of power**

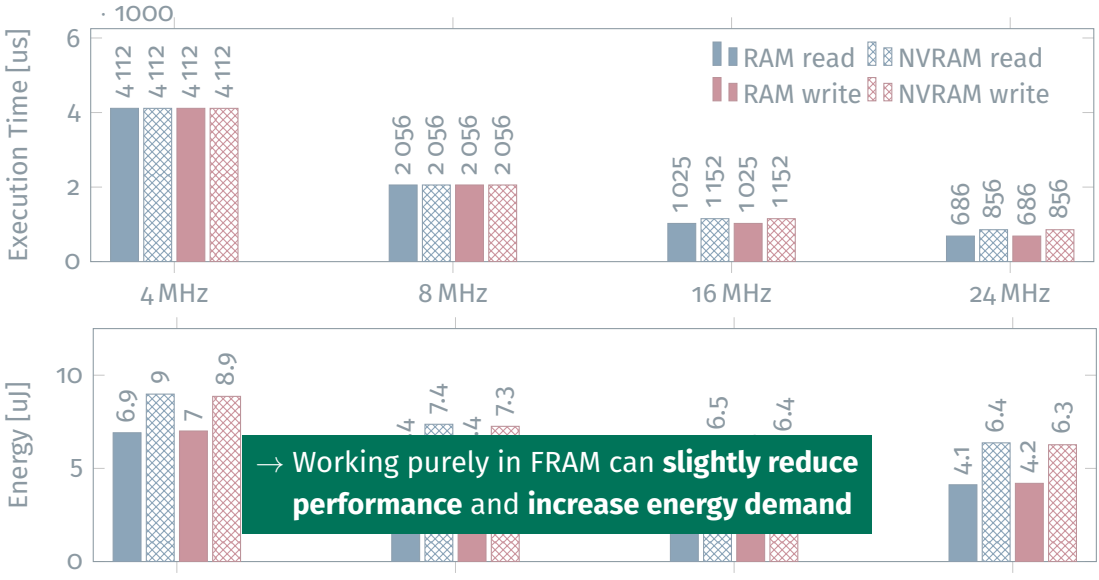
Evaluation - Performance of NVRAM vs. RAM (sequential read/write)



Evaluation - Performance of NVRAM vs. RAM (non-sequential read/write)



Evaluation - Performance of NVRAM vs. RAM (non-sequential read/write)



Neverlast – Conclusion

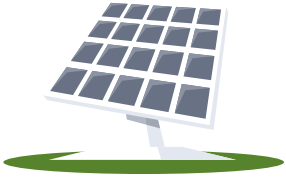
Systems with non-volatile memory (will) become more and more widespread, but come with **challenges**:

- ? Energy-harvesting systems may lose power anytime
- ? Consistency in systems with mixed volatility...
- ? and peripheral devices

Neverlast – Conclusion

Systems with non-volatile memory (will) become more and more widespread, but come with **challenges**:

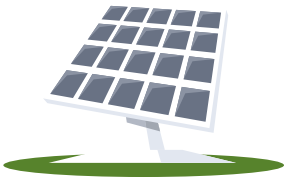
- ✓ Energy-harvesting systems may lose power anytime
  → **Monitor power supply** & detect power outages
- ? Consistency in systems with mixed volatility...
- ? and peripheral devices



Neverlast – Conclusion

Systems with non-volatile memory (will) become more and more widespread, but come with **challenges**:

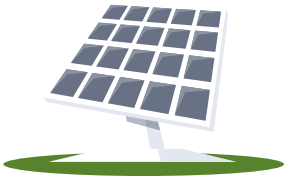
- ✓ Energy-harvesting systems may lose power anytime
  → **Monitor power supply** & detect power outages
- ✓ Consistency in systems with mixed volatility...
  → **Save volatile system state** (registers) on power failure
- ? and peripheral devices



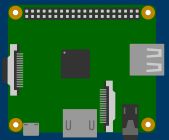
Neverlast – Conclusion

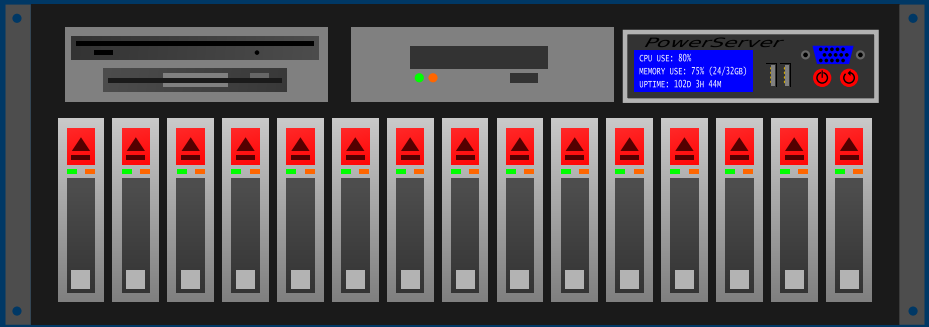
Systems with non-volatile memory (will) become more and more widespread, but come with **challenges**:

- ✓ Energy-harvesting systems may lose power anytime
  → **Monitor power supply** & detect power outages
- ✓ Consistency in systems with mixed volatility...
  → **Save volatile system state** (registers) on power failure
- ✓ and peripheral devices
  → **Trace & replay** device-configuration syscalls



A Peak into the Future...



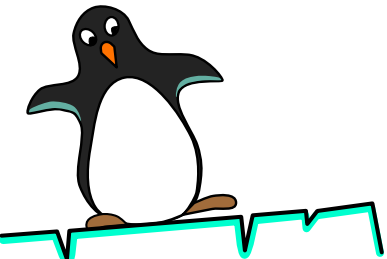


Operating Systems and their Memories

Persistent memory used to be (really) slow and high latency

- Think of: Mechanical hard drives
- OSes avoid persistent memory when possible

→ Use of volatile caches

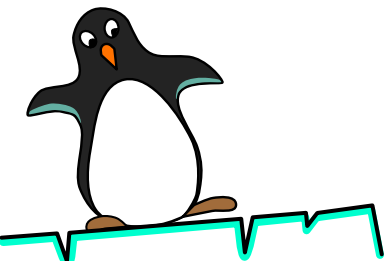


Operating Systems and their Memories

Persistent memory used to be (really) slow and high latency

- Think of: Mechanical hard drives
- OSes avoid persistent memory when possible

→ Use of volatile caches

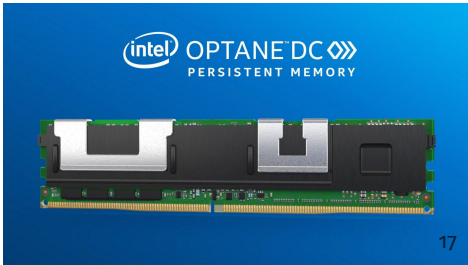


The OS handles RAM volatility in software:

- Periodic flush of write-back file caches
- Periodic flush of the superblock & inode cache
- „Hibernate to disk“

Running Linux in NVRAM – without Volatile RAM

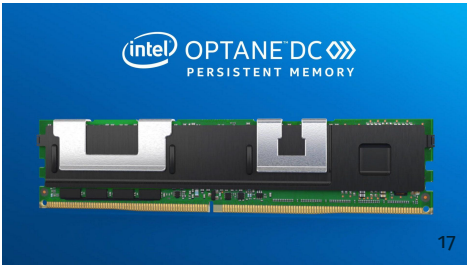
Imagine using (only) non-volatile main memory:



Running Linux in NVRAM – without Volatile RAM

Imagine using (only) non-volatile main memory:

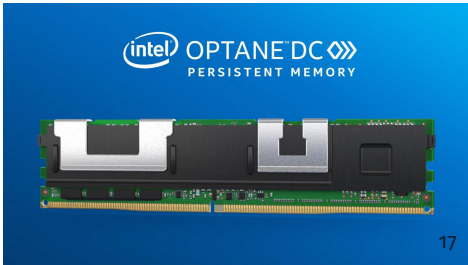
- Previously volatile data is no longer volatile
  $\rightarrow$ OS persistency measures less/no longer important



Running Linux in NVRAM – without Volatile RAM

Imagine using (only) non-volatile main memory:

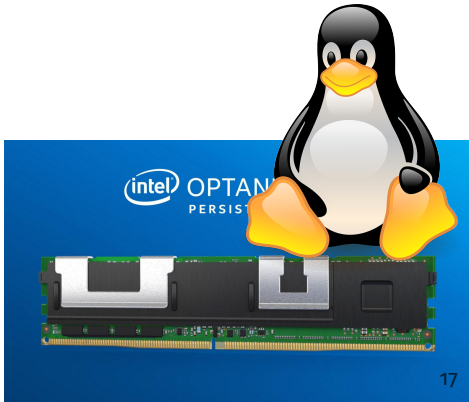
- Previously volatile data is no longer volatile
  $\rightarrow$ OS persistency measures less/no longer important
- But volatile hardware components remain:
  caches, registers, device configuration



Running Linux in NVRAM – without Volatile RAM

Imagine using (only) non-volatile main memory:

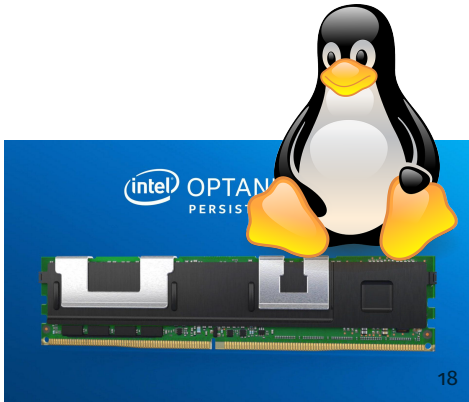
- Previously volatile data is no longer volatile
  $\rightarrow$ OS persistency measures less/no longer important
- But volatile hardware components remain:
  caches, registers, device configuration
- DFG NEON / Linux on Intel Optane:
  - Port Neverlast's ideas to Linux
  - Identify & remove obsolete persistency measures



Conclusion

NVRAM-centric operating systems come with **challenges and opportunities**:

- ? Persistency & consistency in presence of volatile components
- \+ Optimization of the OS:
  - \+ Simplifications
  - \+ Reduced TCB
  - \+ Reduced jitter



DFG HO 6277/1-1 SCHR 603/10-2 CRC/TRR 89PC1

Bundesministerium für Bildung und Forschung 16KIS1315

Conclusion

NVRAM-centric operating systems come with **challenges and opportunities**:

- ? Persistency & consistency in presence of volatile components
- \+ Optimization of the OS:
  - \+ Simplifications
  - \+ Reduced TCB
  - \+ Reduced jitter

DFG HO 6277/1-1 SCHR 603/10-2 CRC/TRR 89PC1

Bundesministerium für Bildung und Forschung 16KIS1315

We're Hiring

Bochum Operating Systems and System Software Group

RUHR UNIVERSITÄT BOCHUM RUB

CCS center of computer science