How PERSISTENT MEMORY Changed Distributed Computing Theory

Wojciech Golab
wgolab@uwaterloo.ca

EMERALD Workshop, Nantes, France
June 21, 2024

UNIVERSITY OF WATERLOO | FACULTY OF ENGINEERING

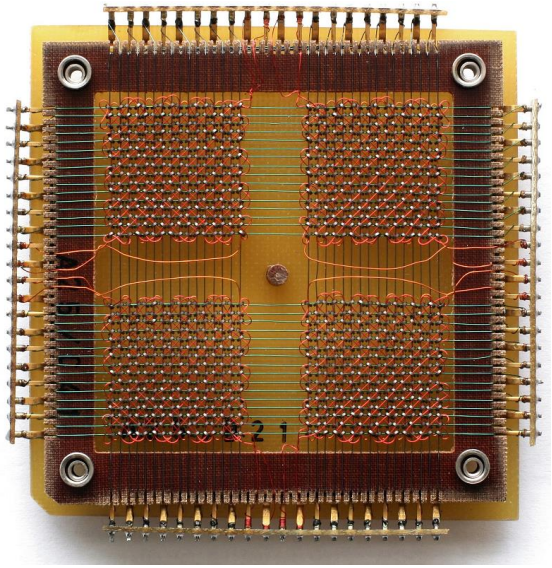
Outline

- Overview of PMem
- Highlights of advances in distributed computing theory
  - failure models
  - synchronization problems
  - algorithmic ideas
- Future of PMem research

OVERVIEW OF PMEM

DRAM and Optane memory side-by-side

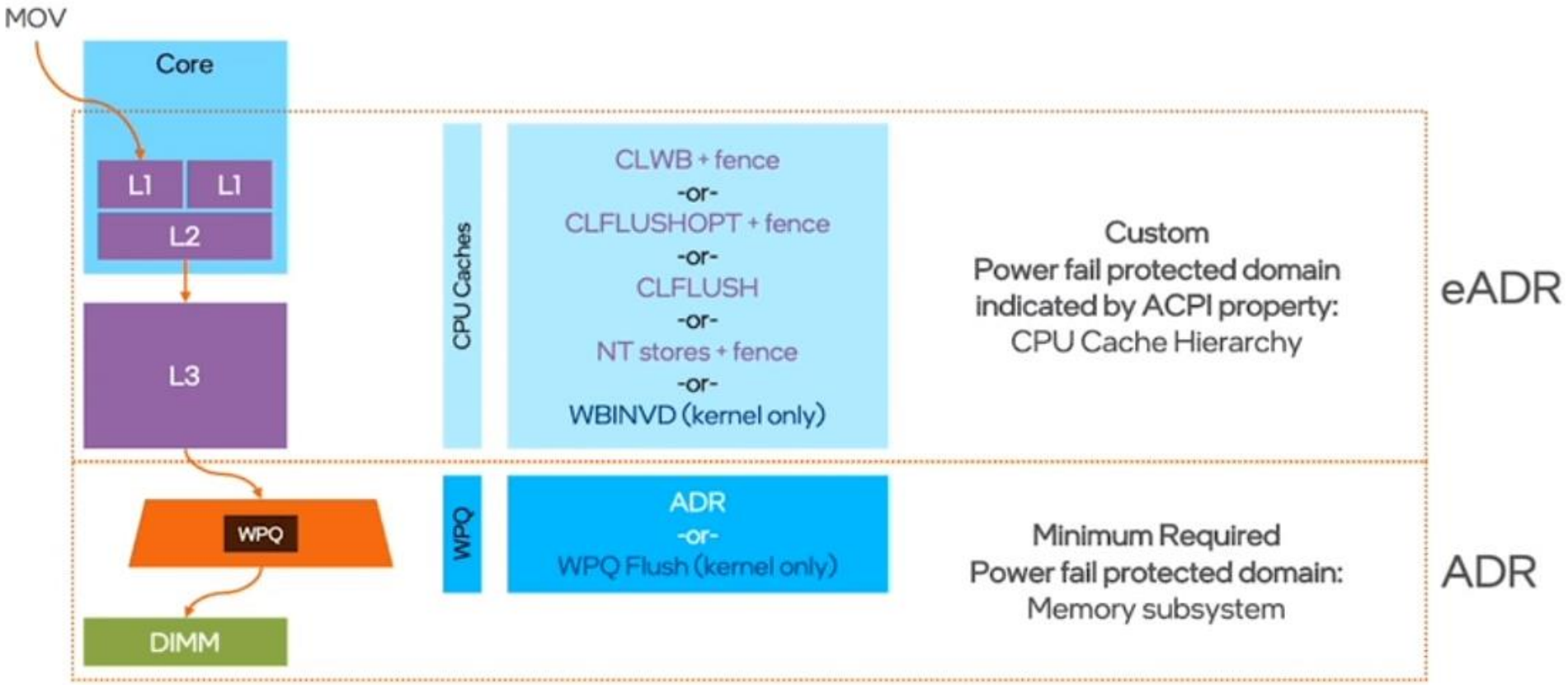
Déjà vu?



Credit: Konstantin Lanzet, CC-BY-SA-3.0, via Wikimedia Commons

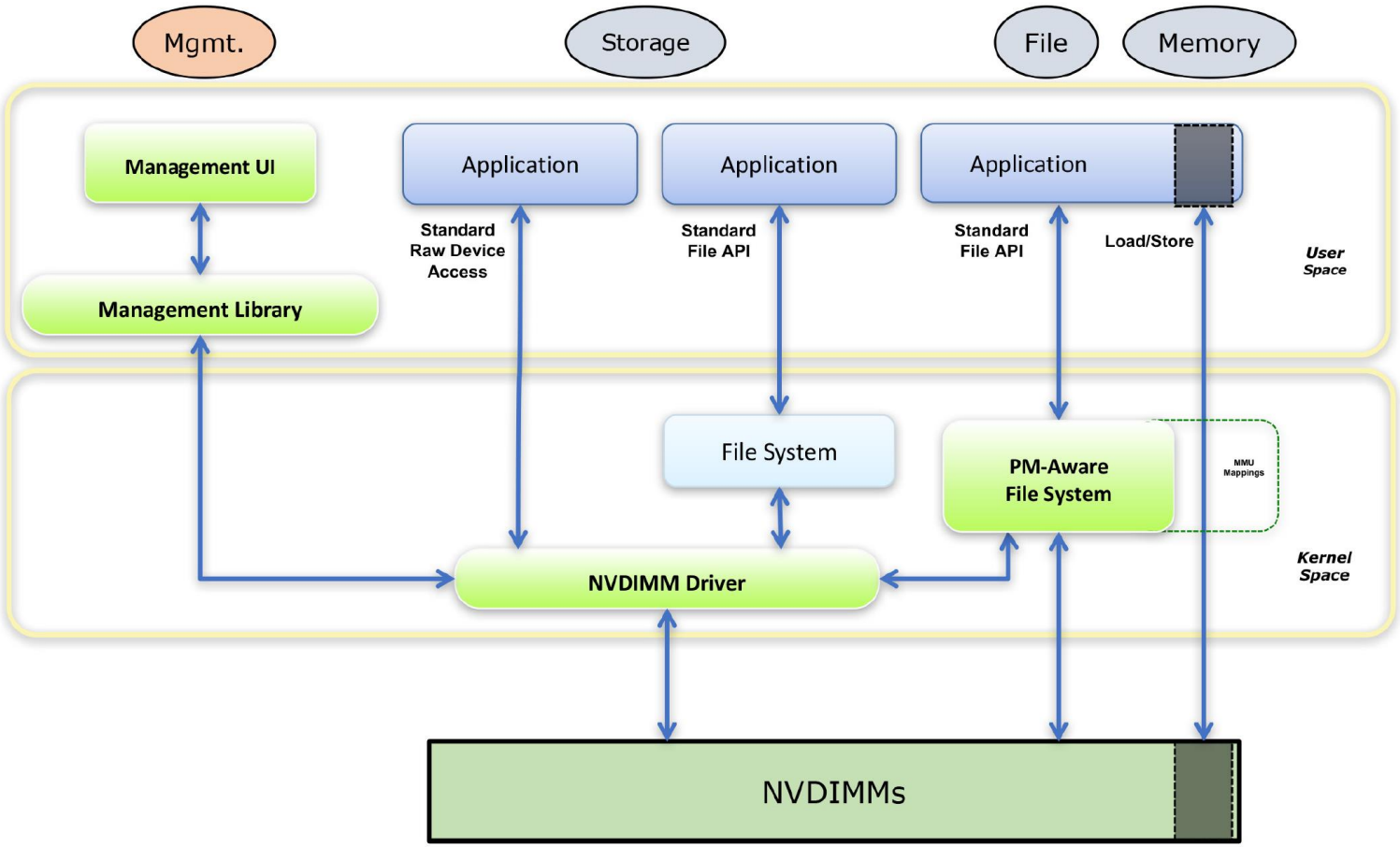
Credit: Viking Technology

How Intel's implementation works



Credit: Intel, https://www.intel.com/content/www/us/en/developer/articles/technical/eadr-new-opportunities-for-persistent-memory-applications.html

The persistent memory programming model



Credit: Andy Rudoff / Intel, USENIX ;login: 2017

The persistent memory programming model

- "App Direct" mode used for data persistence
- hybrid memory hierarchy
  - ordinary RAM for OS kernel, application binaries, call stack
  - memory-mapped files for byte-addressable access to PMem
  - volatile CPU registers (including program counter)
- no support for restarting processes from the point of failure
  - hardware-managed call stack is volatile
  - address space relocation can shift an application's binary code
  - live updates can even change an application's binary code

Whiteboard

Consequences of volatile CPU registers

Leader election example:

```
if T.TestAndSet() then
        // loser
else
        // winner
end if
```

Consequences of volatile CPU registers

Leader election example:

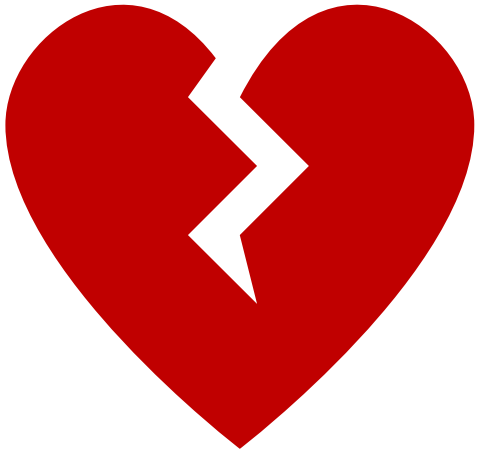
```
CPU register := T.TestAndSet()
if CPU register = 1 then
        // loser
else
        // winner
end if
```

The bittersweet end of Intel Optane PMem

Following Intel's release of second quarter 2022 earnings:

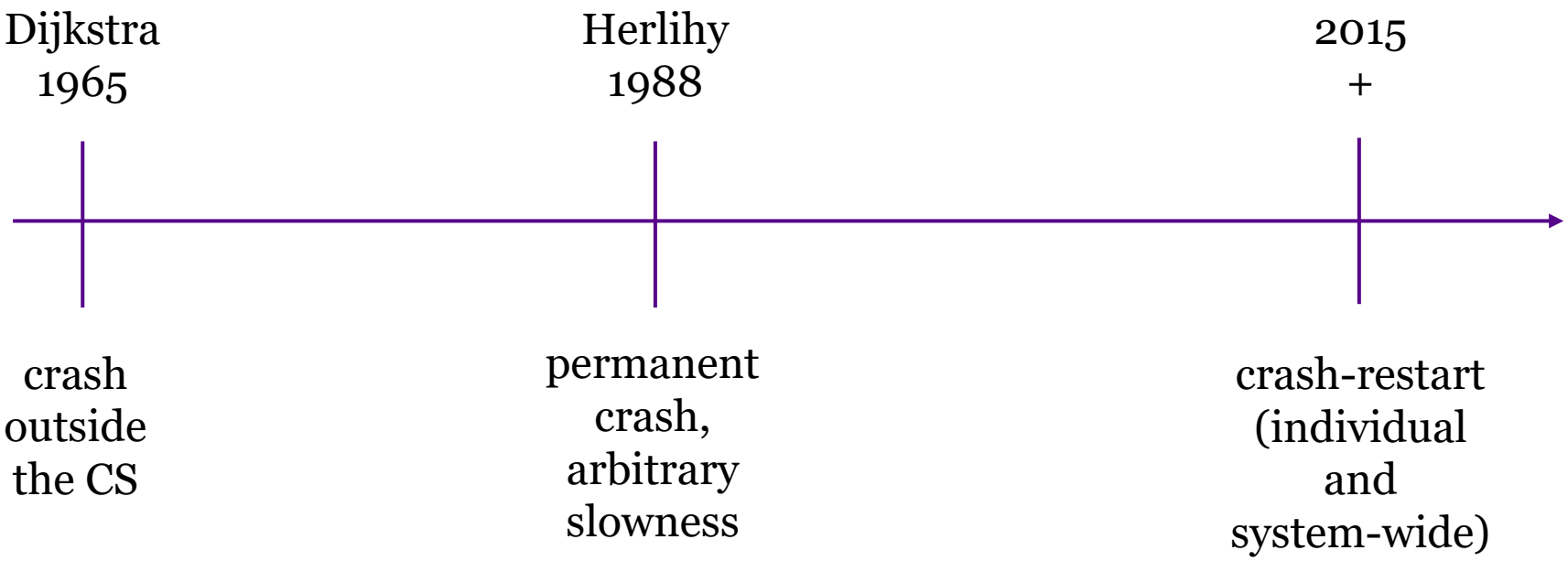
"We are planning for volatility as the world adjusts to the end of a 2-plus-year pandemic and the unprecedented stimulus governments used to fight it."

"First, we further sharpened our focus in Q2, selling our drone business and making the difficult decision to wind down our efforts in Optane as we embrace CXL, a standard which Intel Corporation pioneered."



HIGHLIGHTS OF ADVANCES IN DISTRIBUTED COMPUTING THEORY

Failure models



Failure models

Individual crash failures:

- Ryan Berryhill, Wojciech Golab, Mahesh Tripunitara: Robust Shared Objects for Non-Volatile Main Memory. OPODIS 2015: 20:1-20:17

System-wide crash failures:

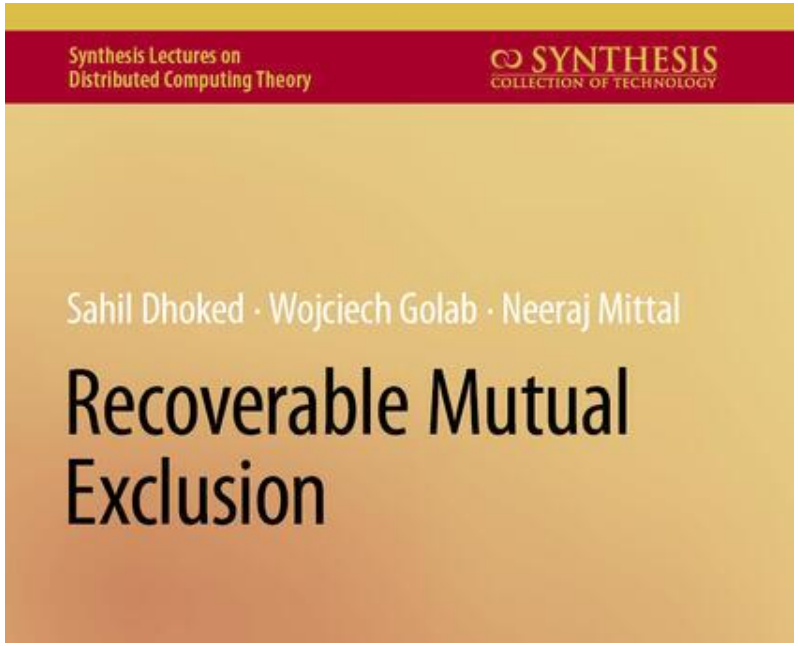
- Joseph Izraelevitz, Hammurabi Mendes, Michael L. Scott: Linearizability of Persistent Memory Objects Under a Full-System-Crash Failure Model. DISC 2016: 313-327

Individual crash failures with auto-restart:

- Hagit Attiya, Ohad Ben-Baruch, Danny Hendler: Nesting-Safe Recoverable Linearizability: Modular Constructions for Non-Volatile Memory. PODC 2018: 7-16

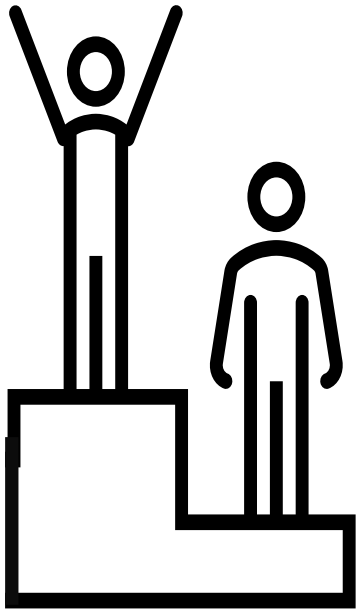
New take on classic synchronization problems

- Wojciech Golab, Aditya Ramaraju: Recoverable Mutual Exclusion. PODC 2016: 65-74
- Wojciech Golab: The Recoverable Consensus Hierarchy. SPAA 2020: 281-291



RMR complexity

FAS
FAI
O(1)



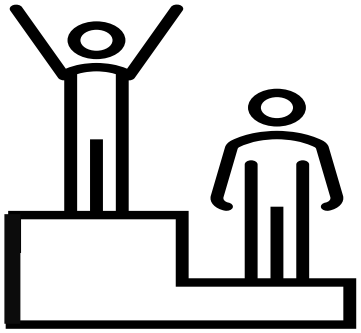
O(log *N*)

TAS
CAS
read/write
registers

ME

O(log *N* / log log *N*)

FAS

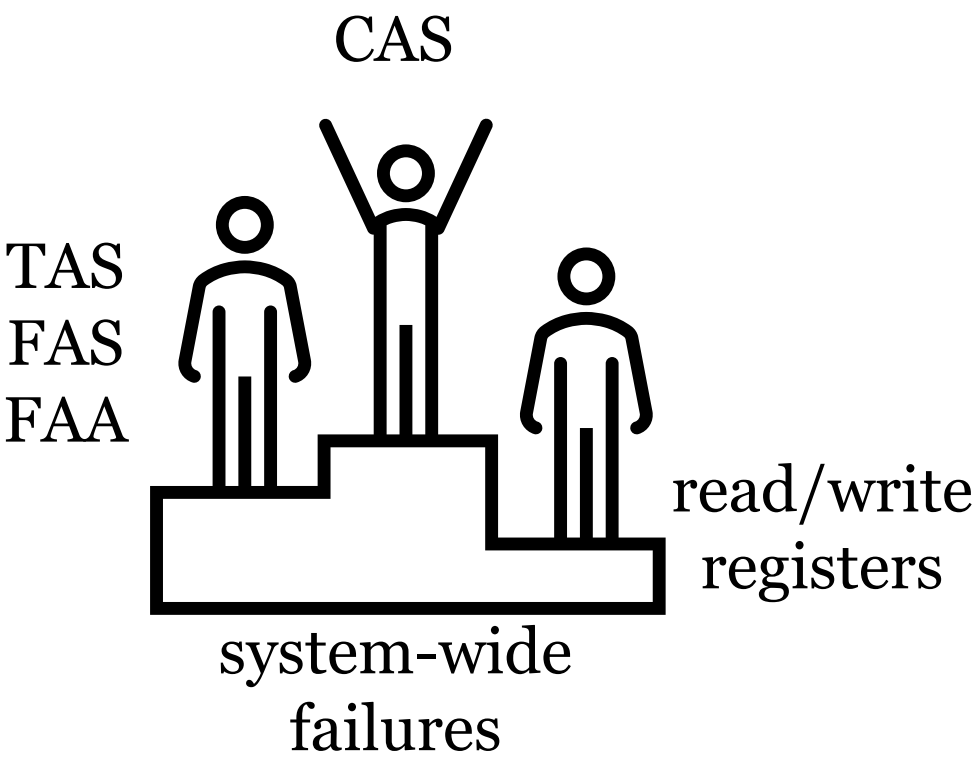


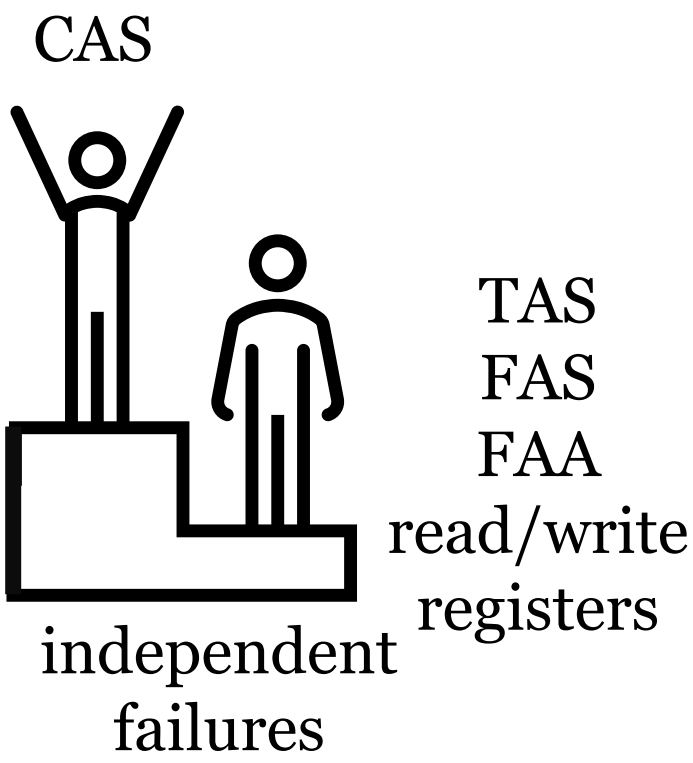
O(log *N*)

TAS
CAS
read/write
registers

RME
(individual process failures)

Recoverable consensus numbers





Additional references for RMRs and consensus numbers

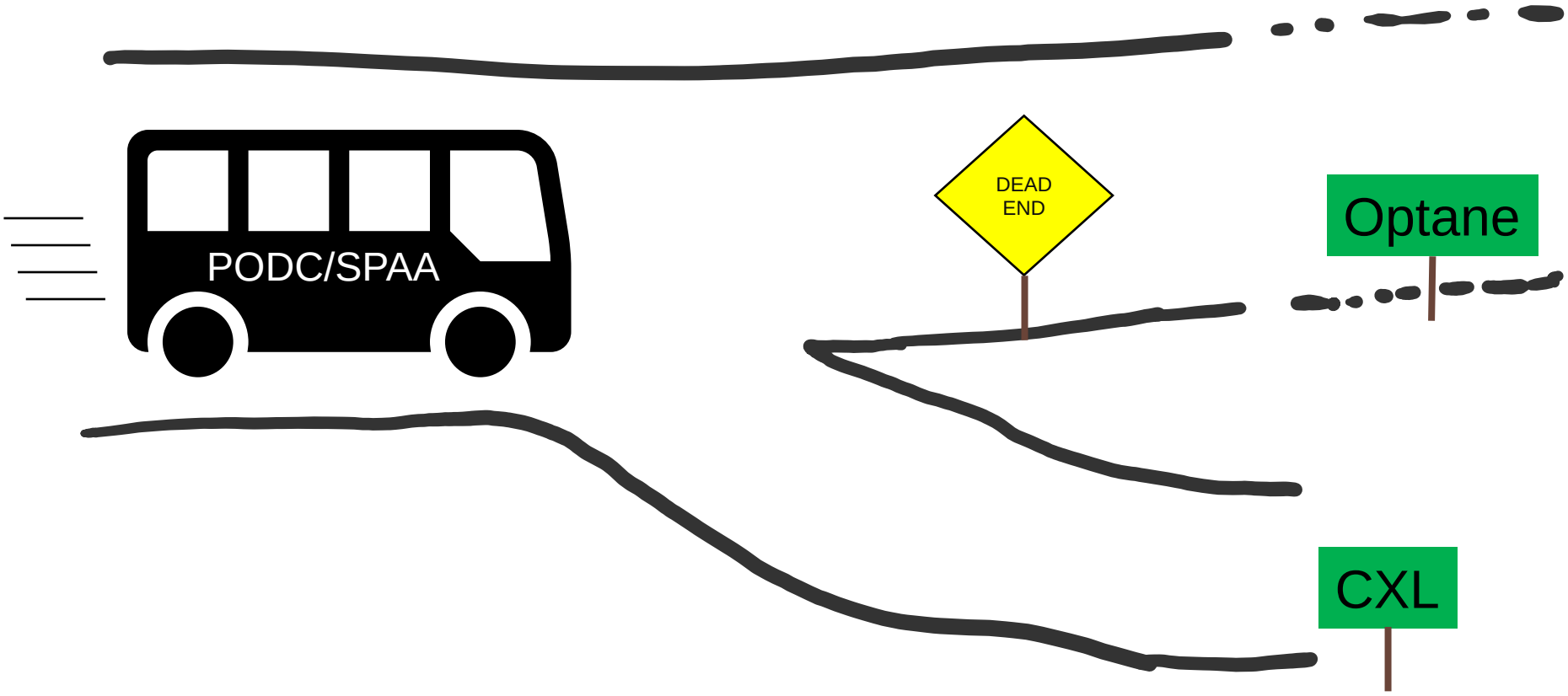
- Wojciech Golab, Danny Hendler: Recoverable Mutual Exclusion in Sub-logarithmic Time. PODC 2017: 211-220
- Prasad Jayanti, Siddhartha V. Jayanti, Anup Joshi: A Recoverable Mutex Algorithm with Sub-logarithmic RMR on Both CC and DSM. PODC 2019: 177-186
- Daniel Katzan, Adam Morrison: Recoverable, Abortable, and Adaptive Mutual Exclusion with Sublogarithmic RMR Complexity. OPODIS 2020: 15:1-15:16
- David Yu Cheng Chan, Philipp Woelfel: Tight Lower Bound for the RMR Complexity of Recoverable Mutual Exclusion. PODC 2021: 533-543
- David Yu Cheng Chan, George Giakkoupis, Philipp Woelfel: Word-Size RMR Tradeoffs for Recoverable Mutual Exclusion. PODC 2023: 79-89
- Carole Delporte-Gallet, Panagiota Fatourou, Hugues Fauconnier, Eric Ruppert: When is Recoverable Consensus Harder Than Consensus? PODC 2022: 198-208
- Prasad Jayanti, Siddhartha Jayanti, Anup Joshi: Constant RMR System-wide Failure Resilient Durable Locks with Dynamic Joining. SPAA 2023: 227-237
- Sahil Dhoked, Wojciech Golab, Neeraj Mittal: Modular Recoverable Mutual Exclusion Under System-Wide Failures. DISC 2023: 17:1-17:24
- Prasad Jayanti, Siddhartha Jayanti, Sucharita Jayanti: Durable Algorithms for Writable LL/SC and CAS with Dynamic Joining. DISC 2023: 25:1-25:20
- Sean Ovens: Determining Recoverable Consensus Numbers. PODC 2024: 3-13

Other pmem-inspired algorithmic ideas

- Tianzheng Wang, Justin J. Levandoski, Per-Åke Larson: Easy Lock-Free Indexing in Non-Volatile Memory. ICDE 2018: 461-472
- Tudor David, Aleksandar Dragojevic, Rachid Guerraoui, Igor Zablotchi: Log-Free Concurrent Data Structures. USENIX Annual Technical Conference 2018: 373-386
- Nachshon Cohen, Rachid Guerraoui, Igor Zablotchi: The Inherent Cost of Remembering Consistently. SPAA 2018: 259-269
- Michal Friedman, Maurice Herlihy, Virendra J. Marathe, Erez Petrank: A persistent lock-free queue for non-volatile memory. PPoPP 2018: 28-40
- Naama Ben-David, Guy E. Blelloch, Michal Friedman, Yuanhao Wei: Delay-Free Concurrency on Faulty Persistent Memory. SPAA 2019: 253-264
- Hagit Attiya, Ohad Ben-Baruch, Panagiota Fatourou, Danny Hendler, Eleftherios Kosmas: Tracking in Order to Recover - Detectable Recovery of Lock-Free Data Structures. SPAA 2020: 503-505
- Gal Sela, Erez Petrank: Durable Queues: The Second Amendment. SPAA 2021: 385-397
- Nan Li, Wojciech Golab: Detectable Sequential Specifications for Recoverable Shared Objects. DISC 2021: 29:1-29:19
- Gaetano C. Coccimiglio, Trevor Alexander Brown, Srivatsan Ravi: PREP-UC: A Practical Replicated Persistent Universal Construction. SPAA 2022: 217-229
- Panagiota Fatourou, Nikolaos D. Kallimanis, Eleftherios Kosmas: The performance power of software combining in persistence. PPoPP 2022: 337-352
- Jakeb Chouinard, Kush Kansara, Xialin Liu, Nihal Potdar, Wojciech Golab: Brief Announcement: On Implementing Wear Leveling in Persistent Synchronization Structures. DISC 2023: 38:1-38:7

FUTURE OF PMEM RESEARCH

Time to take the off-ramp



Some reassuring comments from Intel

“Intel’s long-term collaboration with hardware and software developers will help customers to migrate from today’s Intel Optane PMem-based data and memory tiering solutions to tomorrow’s usage models with CXL devices.”

Source: https://www.intel.com/content/www/us/en/products/docs/memory-storage/optane-persistent-memory-to-cxl-attached-memory.html

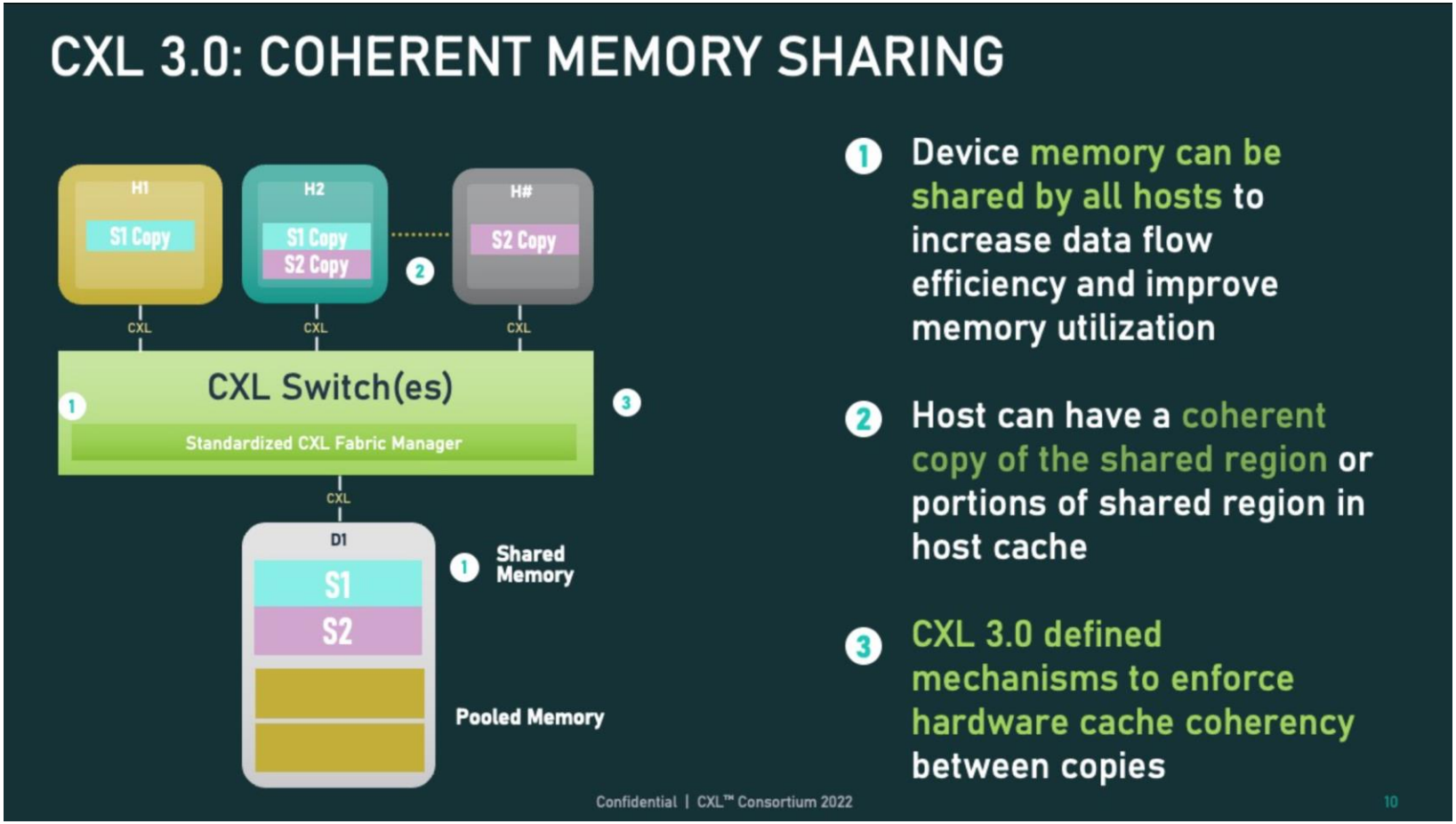
More reassurance from the SNIA

CXL Software ecosystem

CXL Memory Configuration	Administrative steps	Use cases	Programming model (same as PMem)
Default Global volatile memory (system ram as NUMA)	None.	Adding more volatile memory capacity, potentially with software tiering.	Unmodified apps: Traditional memory management, OS-managed NUMA locality. Modified apps: Speciality NUMA allocators (e.g., `libnuma`, `memkind`). All apps: Direct use of `mmap`/`mbind`.
Volatile devdax	Reconfiguring namespace to devdax.	Adding new isolated memory capacity, manual tiering.	Speciality allocators capable of operating on raw memory ranges (e.g., `memkind`), manual use of `mmap`.
Volatile use of fsdax	Configuring pmem region and fsdax namespace.	Named volatile regions of volatile memory using file system to control access.	Speciality allocators capable of managing pools on top of file systems (e.g., `memkind`). **Note** For new software, a better alternative may be using tmpfs bound to a system-ram NUMA node. It's likely to be faster and less error-prone.
Persistent fsdax	Configuring pmem region and fsdax namespace.	Existing PMem-aware or storage-based software that uses regular files.	SNIA Persistent Memory Programming Model. Unmodified apps just work. New ones can still use PMDK.
Persistent devdax	Configuring pmem region and devdax namespace.	Custom software requiring full control of memory.	Raw access through `mmap`, can flush using CPU instructions. Apps can use PMDK.

Source:
https://www.snia.org/sites/default/files/SSSI/20230920_PM_Hackathon_at_SDC.pdf

A “confidential” slide from the CXL Consortium



Does anything change?

1. part of the address space is shared only with processes on the same host, and part is shared globally
2. one host can suffer a “system-wide” crash independently of another
3. CXL-attached memory could theoretically fail while processes continue running

Whiteboard

UNIVERSITY OF WATERLOO

FACULTY OF ENGINEERING

Contact address for questions: wgolab@uwaterloo.ca

More bad news …

"… 0.44% annualized failure rate …"

Source:
https://www.tomshardware.com/news/intel-extends-availability-of-optane-dimms