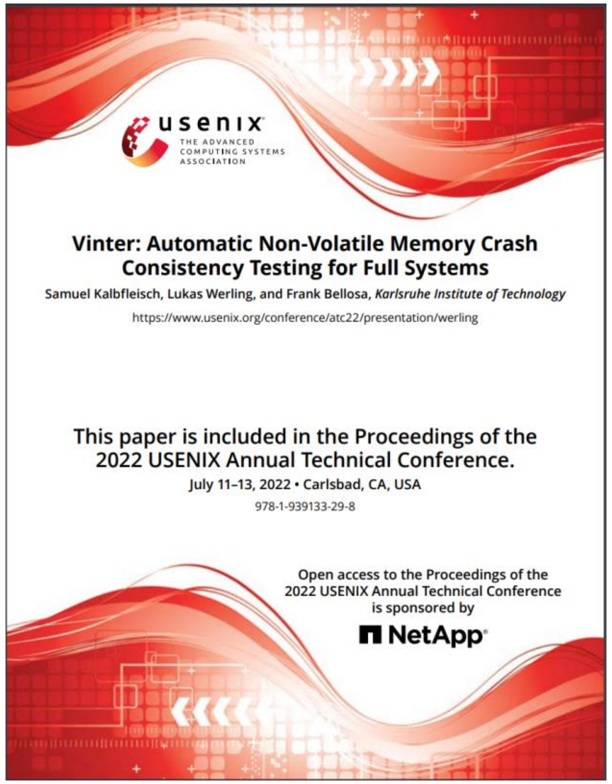
Karlsruhe Institute of Technology

Crash Consistency Testing for Non-Volatile Memory Systems

Lukas Werling (Operating Systems Group)

Motivation

- Crash consistency is a common and important goal
  - Recover semantically correct state
- … but rarely evaluated!
  - 1 of 11 papers advertising crash consistency published this year
- Vinter (ATC’22): Automatic testing of file systems
- More tools available for other use cases


usenix THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Vinter: Automatic Non-Volatile Memory Crash Consistency Testing for Full Systems

Samuel Kalbfleisch, Lukas Werling, and Frank Bellosa, *Karlsruhe Institute of Technology*

https://www.usenix.org/conference/atc22/presentation/werling

This paper is included in the Proceedings of the 2022 USENIX Annual Technical Conference.

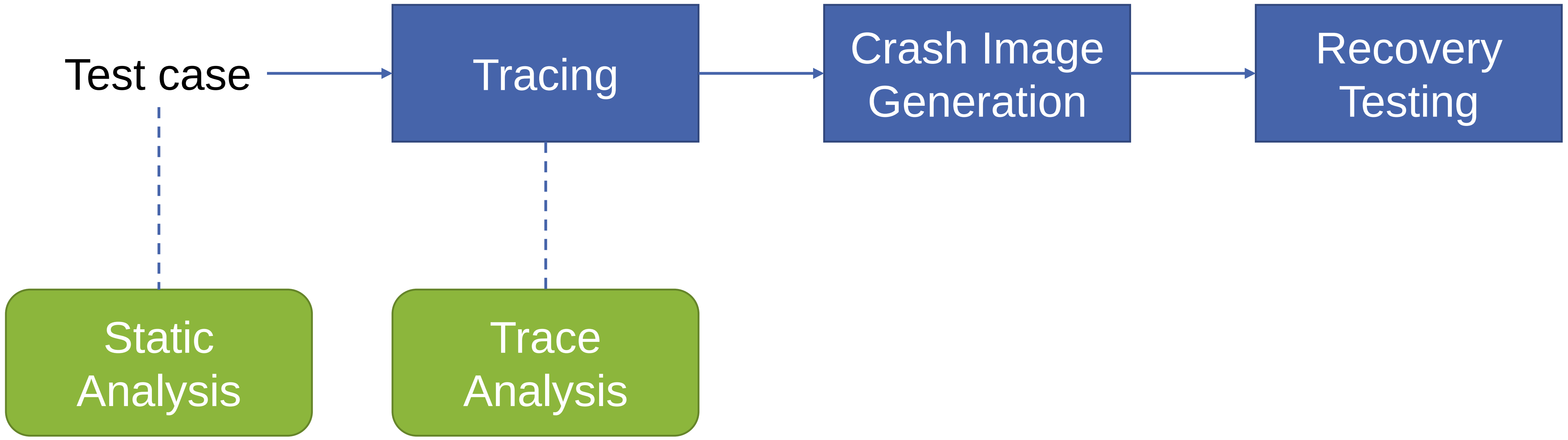
July 11–13, 2022 • Carlsbad, CA, USA

978-1-939133-29-8

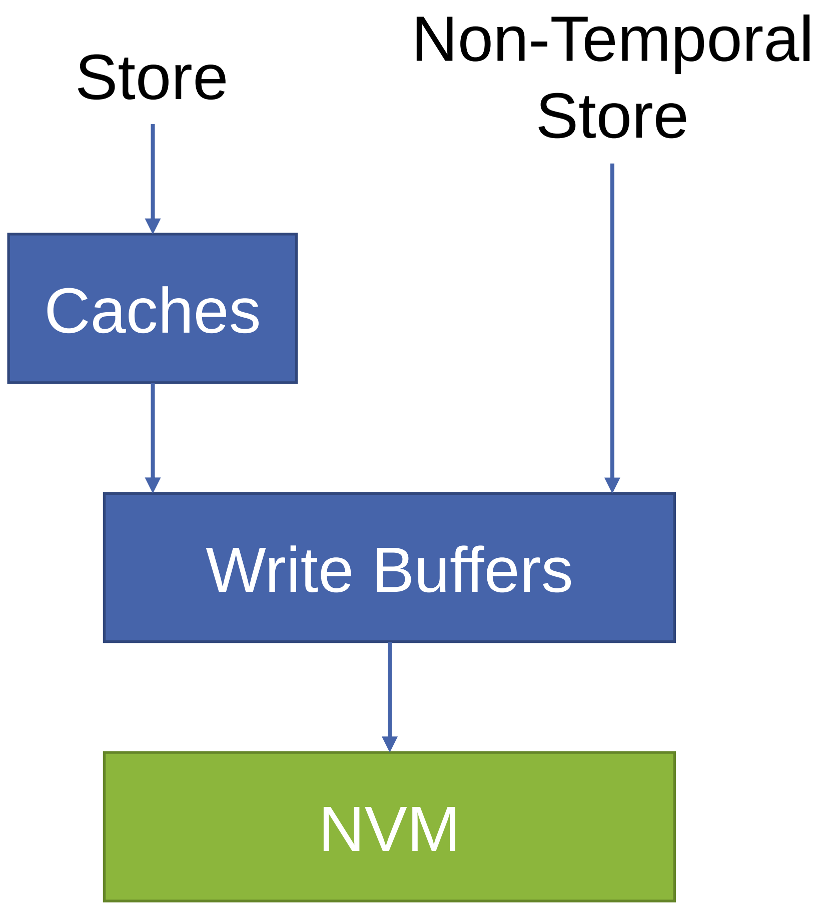
Open access to the Proceedings of the 2022 USENIX Annual Technical Conference is sponsored by NetApp

Consider testing crash consistency in **your** systems!

Crash Consistency Testing Pipeline



Tracing: ISA Semantics



Which store instructions?
Weak ordering → fence

Volatile caches → clflush
Intra cache line ordering

Volatile buffers → commit

Tracing Methods

Manual Annotation	Binary Instrumentation	Virtualization
• Easy to implement • Fast • Usually high level (e.g., library calls) • Mistakes likely	• Automatic (e.g., compiler plugin) • Limited to user space applications • Additional context from source code	• Automatic, black box • Captures full system (user and kernel) • Expensive to trace arbitrary instructions (e.g., emulation)

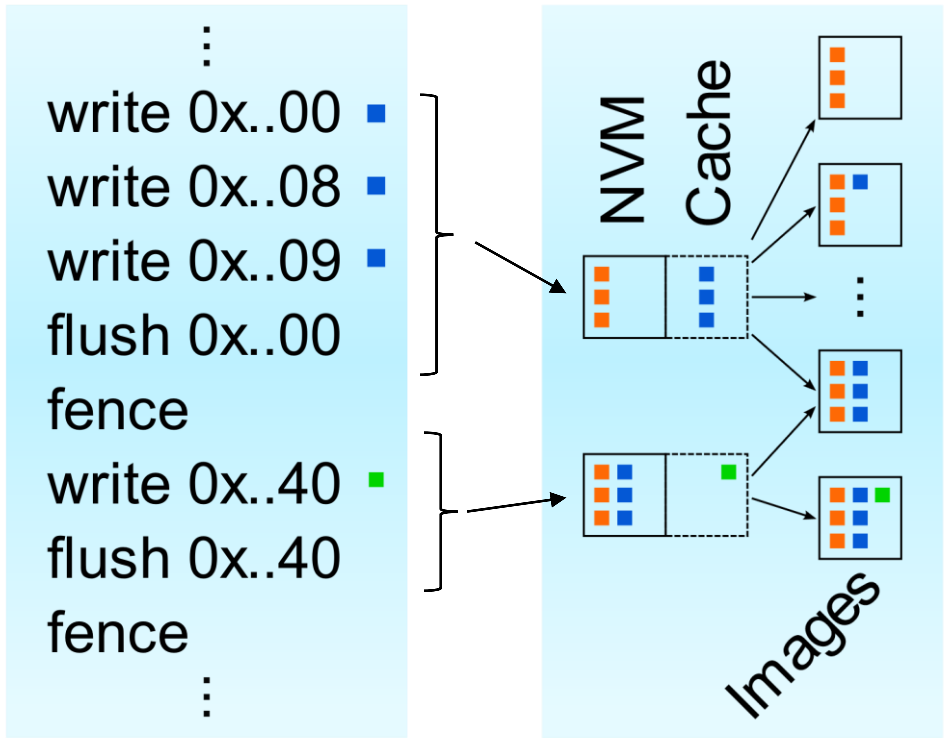
Crash Image Generation

- Replay memory trace: NVM, cache
- Generate crash images at each fence
  - “Happy path”: every write persisted
  - Subsets of writes

NOVA: Up to 512 modified cache lines

State explosion!

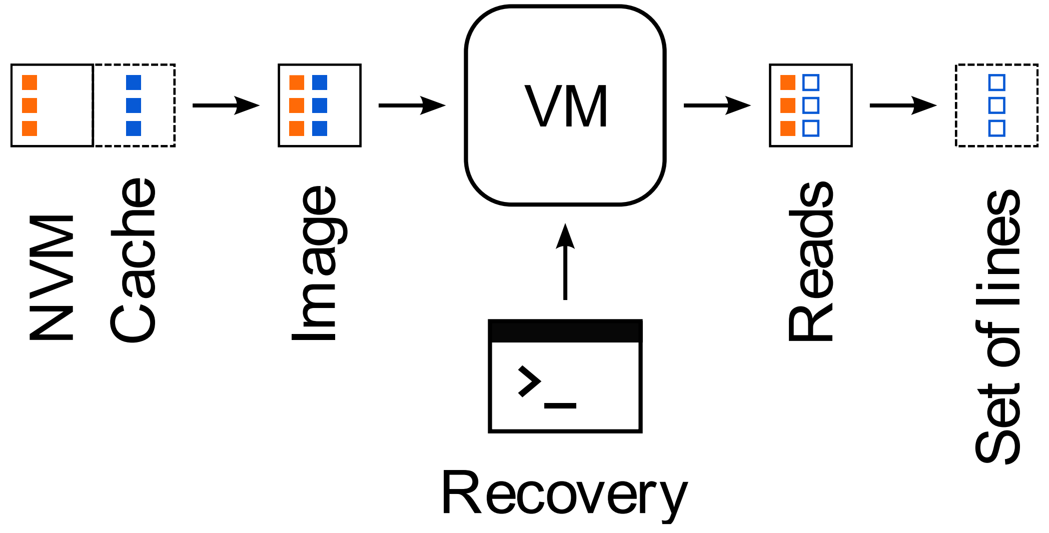
Crash Image Generator



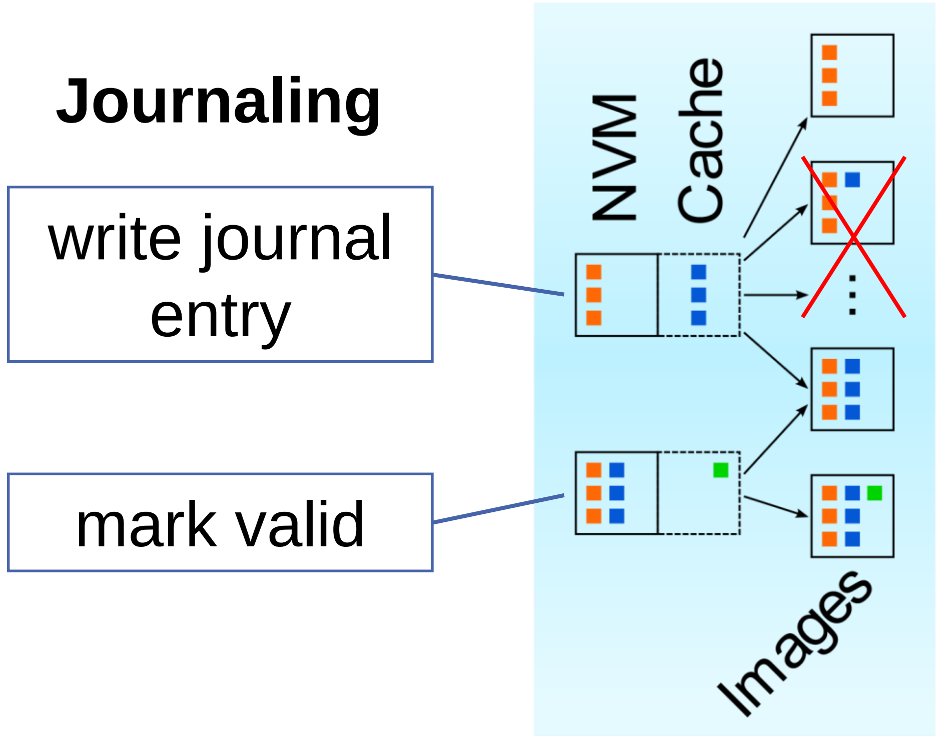
Crash Image Generation: Heuristic

Crash Image Generator

- Observation: Recovery ignores incomplete journal entries
- Idea: Trace NVM reads during recovery

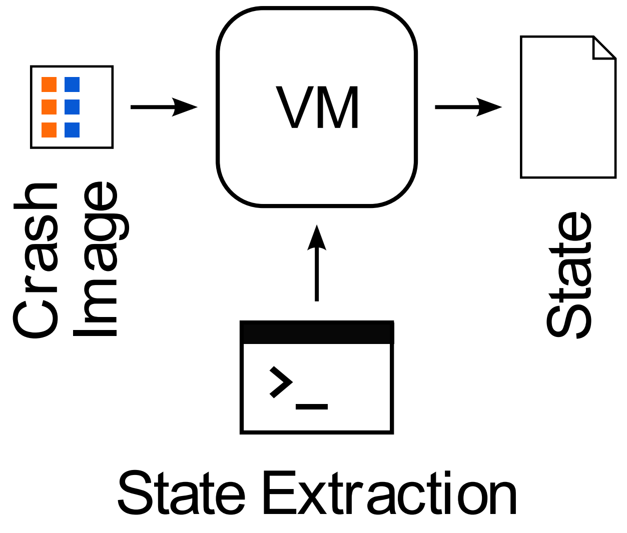


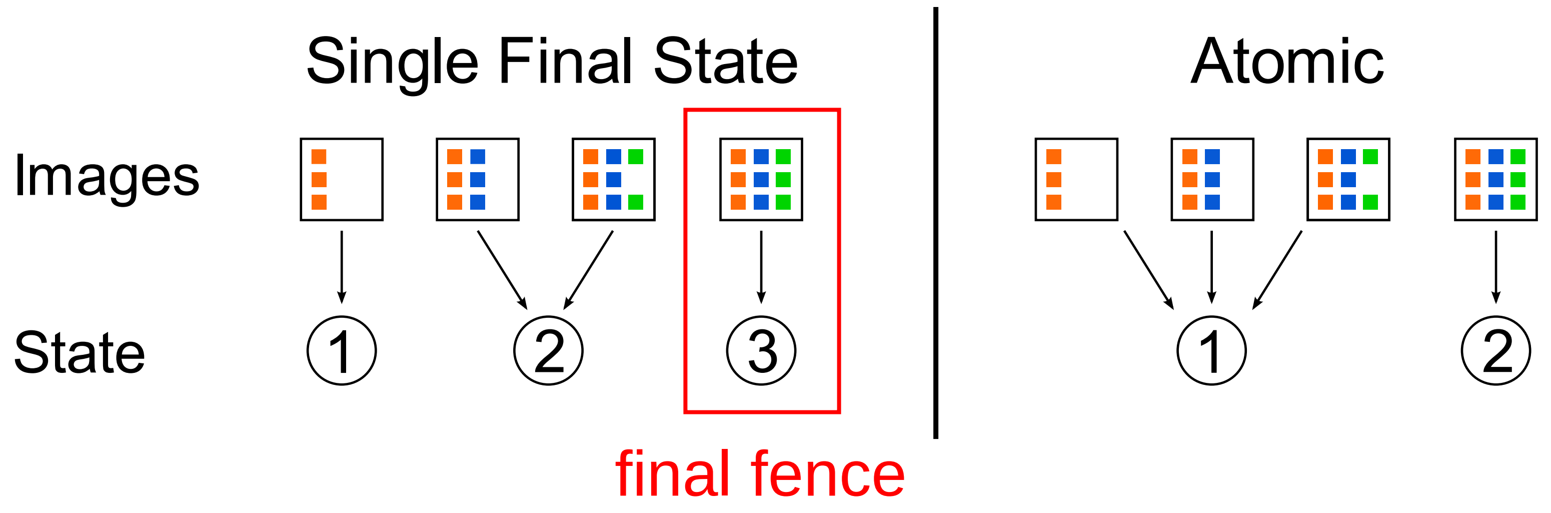
Efficient generation of interesting crash images



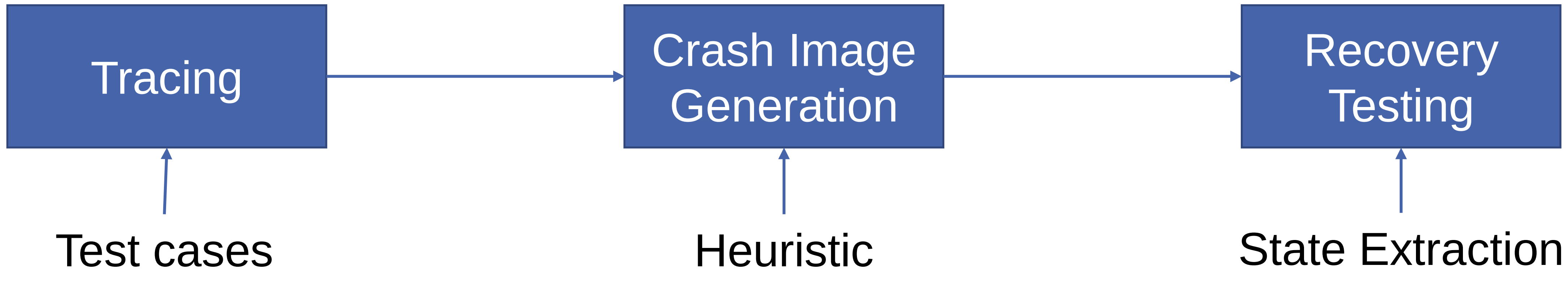
Recovery Testing

- Idea: Extract semantic state from crash images
  - File systems: serialized file listing
- Find unique states for analysis





Pipeline Inputs



- Hand-written
- Existing test suite
- Fuzzing

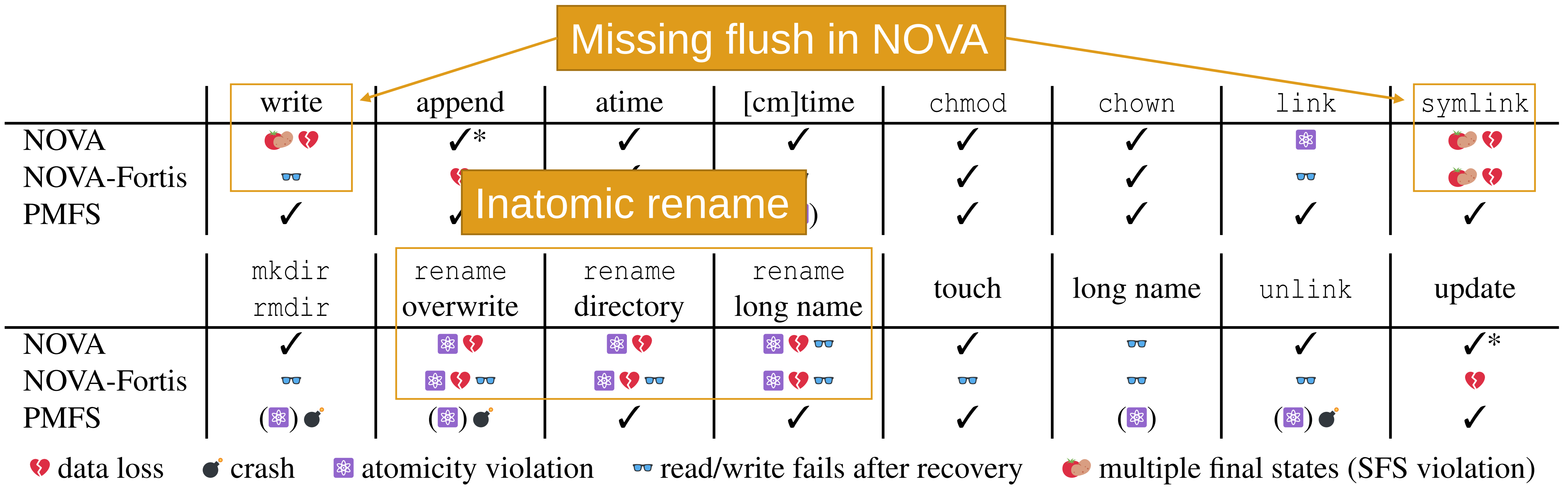
- Recovery reads (Vinter)
- Data/control dependencies (Witcher[1])
- Library calls (Chipmunk[2])
- Ignore partial flushes

- Serialize all state
- Determine expected semantics

[1] SOSP'21 [2] EuroSys'23

Crash Consistency Bugs in Practice

- Vinter: Analysis of three NVM file systems


Missing flush in NOVA
Inatomic rename

	write	append	atime	[cm]time	chmod	chown	link	symlink
NOVA	🍅 💔	✓*	✓	✓	✓	✓	⚛	🍅 💔
NOVA-Fortis	👓	💔	✓	✓	✓	✓	👓	🍅 💔
PMFS	✓	✓	[illegible]	(⚛)	✓	✓	✓	✓

	mkdir rmdir	rename overwrite	rename directory	rename long name	touch	long name	unlink	update
NOVA	✓	⚛ 💔	⚛ 💔	⚛ 💔 👓	✓	👓	✓	✓*
NOVA-Fortis	👓	⚛ 💔 👓	⚛ 💔 👓	⚛ 💔 👓	👓	👓	👓	💔
PMFS	(⚛) 💣	(⚛) 💣	✓	✓	✓	(⚛)	(⚛) 💣	✓

💔 data loss 💣 crash ⚛ atomicity violation 👓 read/write fails after recovery 🍅 multiple final states (SFS violation)

Figure 4: Crash consistency bugs discovered by VINTER.

Bug Detail: Missing Flush in NOVA

Test command: `echo HelloWorld > /mnt/myfile`

Vinter report: 7 states, 4 final states

```
HelloWorld\n    HelloWor\0\0\0    HelloWorl\0\0    HelloWorld\0
```

```
NT-write 46784 + 0  "HelloWor"
write    46784 + 8  'l'
write    46784 + 9  'd'
write    46784 + 10 '\n'
```

No flush!

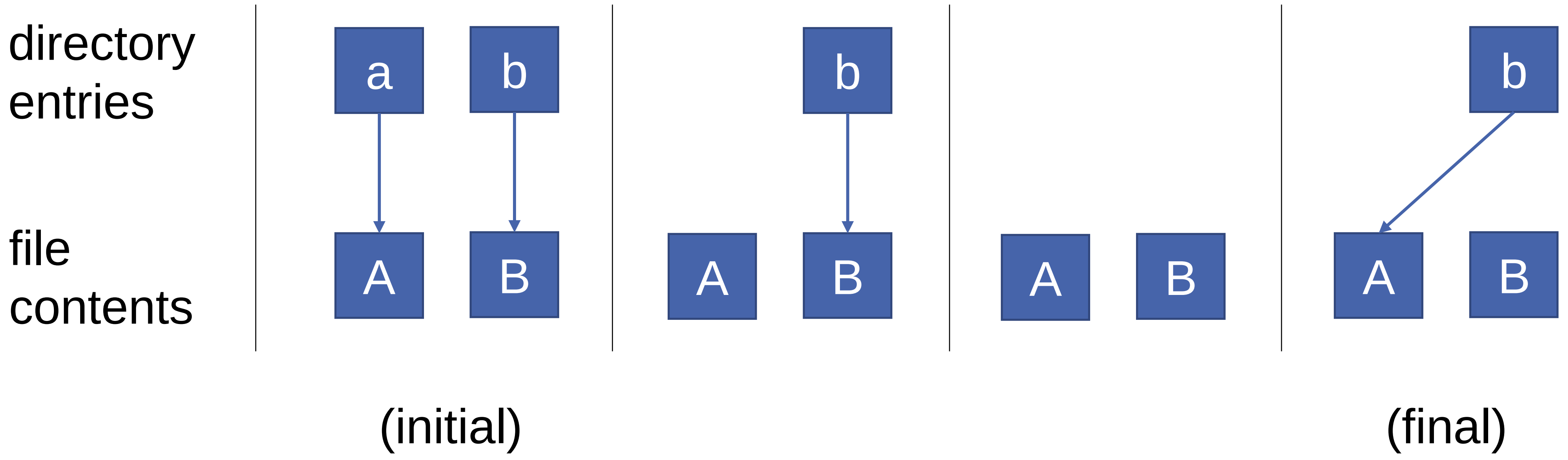
Stack trace:

```
__copy_user_nocache
do_nova_inplace_file_write
...
vfs_write
...
```

Bug Detail: NOVA Rename Atomicity

Test command: `mv /mnt/a /mnt/b`

Vinter report: 4 states, 1 final state → not atomic!



Observations

- Instruction-level tracing important for NVM primitives
  - ASM implementations!
- Testing of simple operation sufficient for good coverage
- Logic bugs are still prevalent

Conclusion

- Crash consistency is important, but rarely evaluated

- Crash consistency testing pipeline
  - Tracing NVM events
  - Crash image generation with heuristic
  - Recovery testing

- Crash consistency bugs in practice

Consider testing crash consistency in **your** systems!

https://github.com/KIT-OSGroup/vinter