Automatic Non-Volatile Memory Crash Consistency Testing for Kernel Space Applications

Samuel Kalbfleisch, Lukas Werling, Frank Bellosa

Karlsruhe Institute of Technology

September 21, 2021

Non-Volatile Memory (NVM)

- Crash consistency is desired
- Achieving it is difficult:
  - NVM is part of memory hierarchy
  - Cache not necessarily persisted
  - Stores to NVM reordered
  - flush(cache_line), fence(), atomic stores

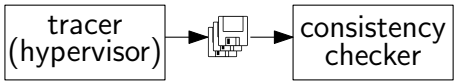
$\rightsquigarrow$ Crash consistency testing

- Existing work not well suitable for testing kernel level file systems

- We propose a new framework based on full system emulation
- We find bugs in the NOVA file system

Prior Work: NVM Bug Discovery

Brute force exploration (Yat '14)



⊖ recompilation, inefficient, ⊕ kernel

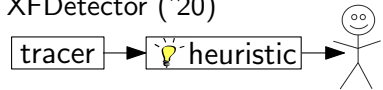
Annotations (PMTest '19)

```
write A
write B
flush A
isOrderedBefore A B
```

⊖ incomplete, manual effort, kernel modification

Heuristics

XFDetector ('20)

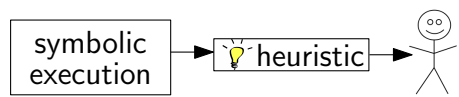


⊖ annotations, manual confirmation

WITCHER ('20)

⊕ automatic, ⊖ no kernel, recompilation

Symbolic execution (Agamotto '20)



$\Longrightarrow$ No efficient, generic, automatic solution that is applicable to file systems

Our Contribution

We contribute a framework that:

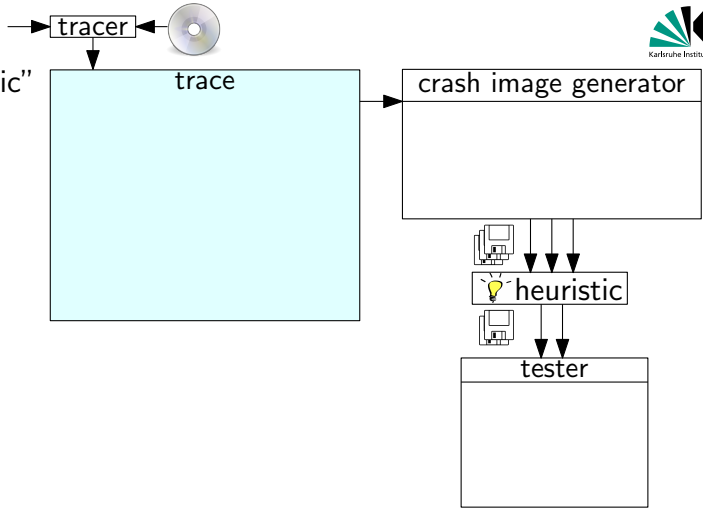
- supports full and unmodified systems
- applies state of the art heuristics
- automates bug discovery process (e.g., test atomicity)

⇝ Allows testing file system crash consistency & atomicity

Approach

"generic, efficient, automatic"

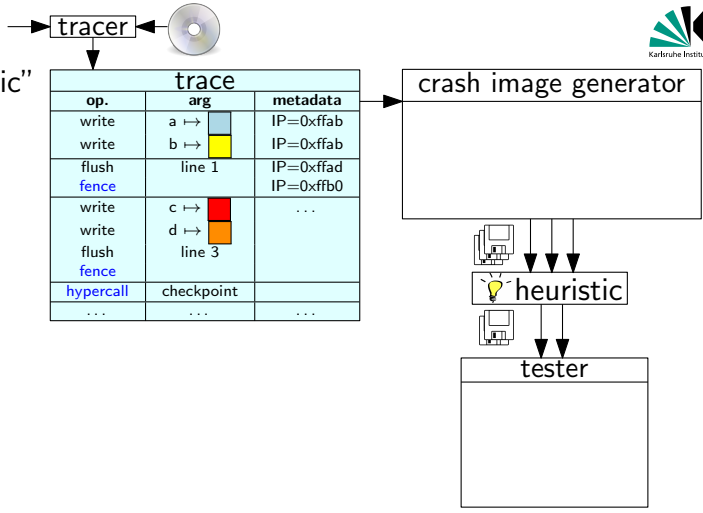
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

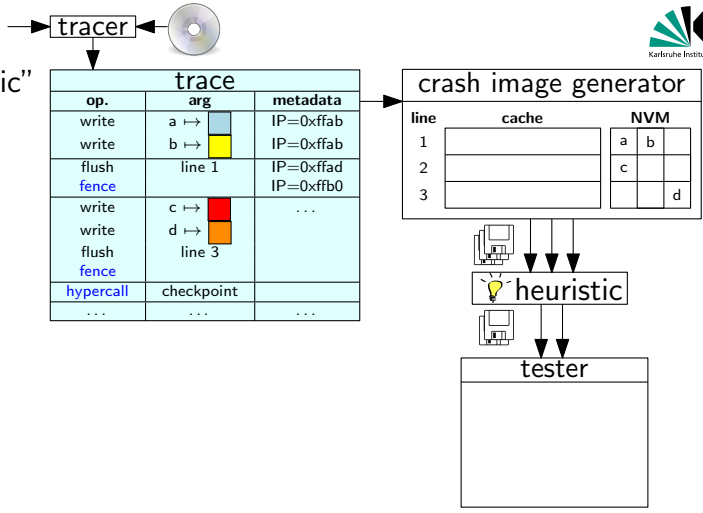
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

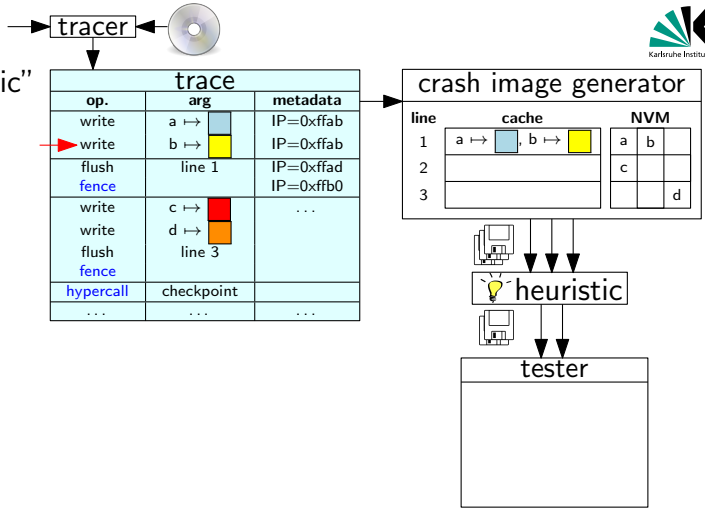
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

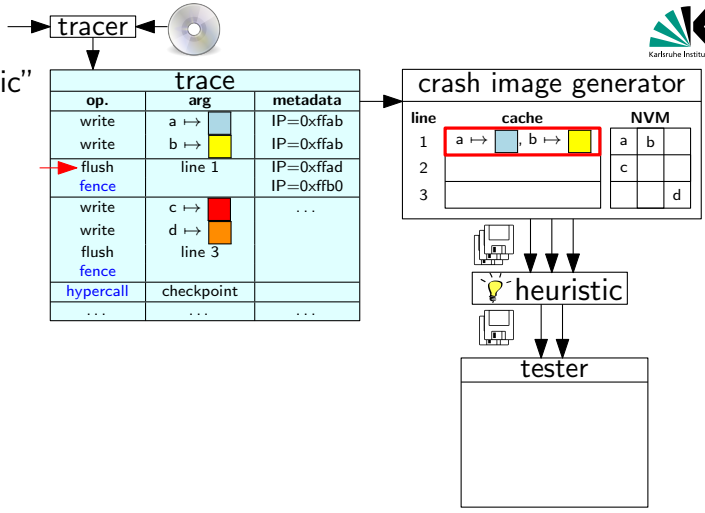
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

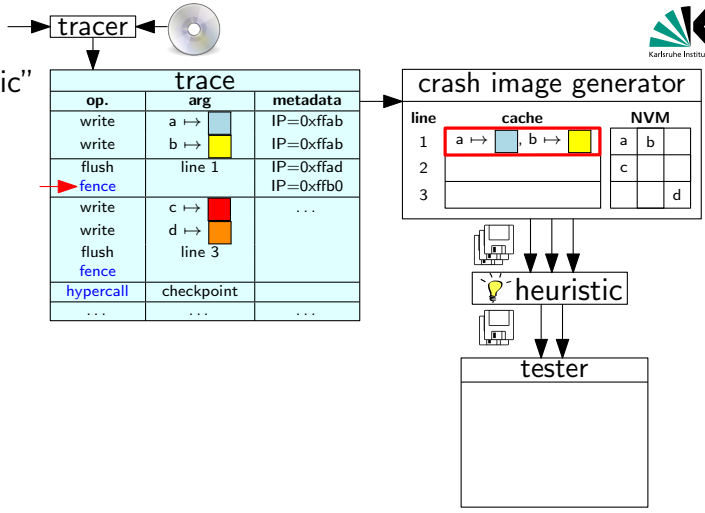
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

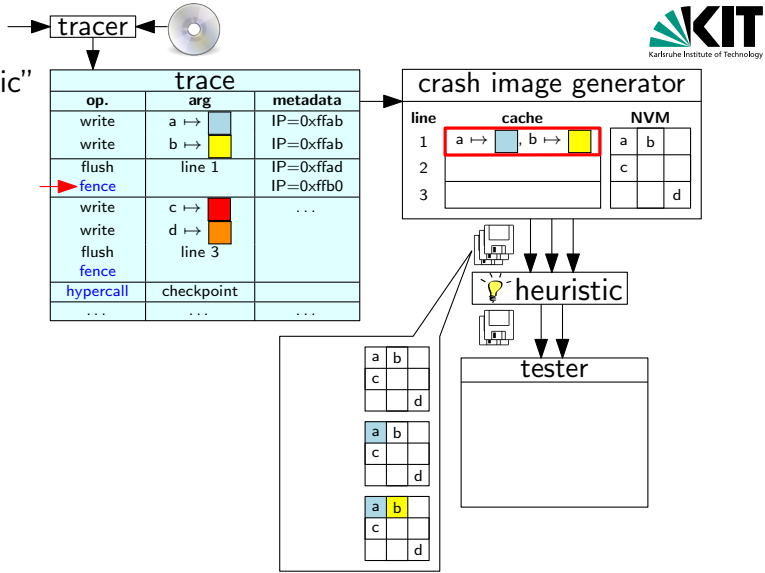
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

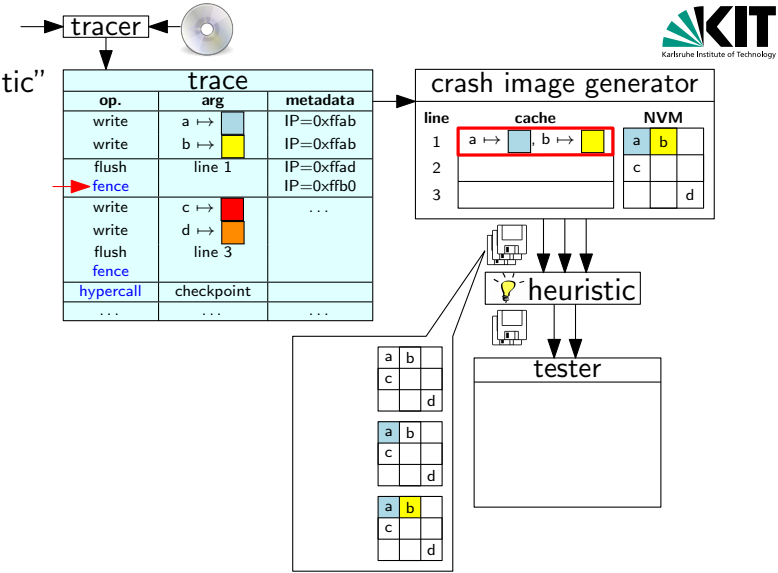
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

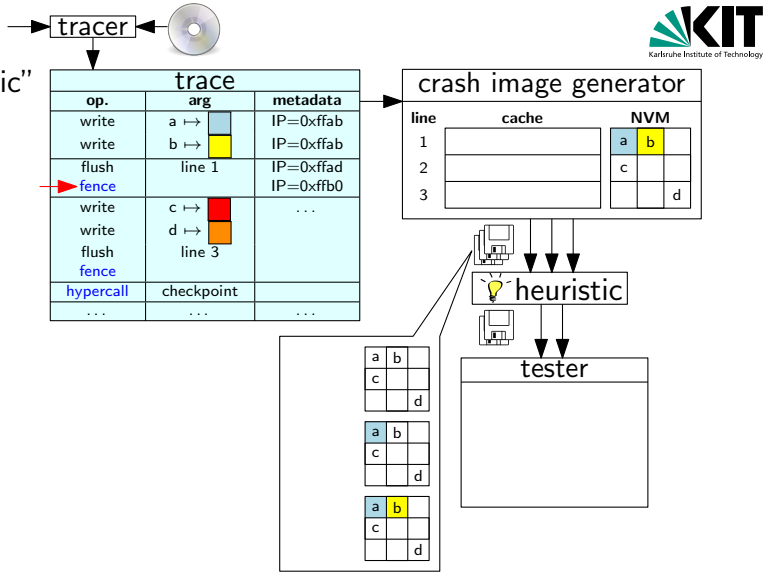
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

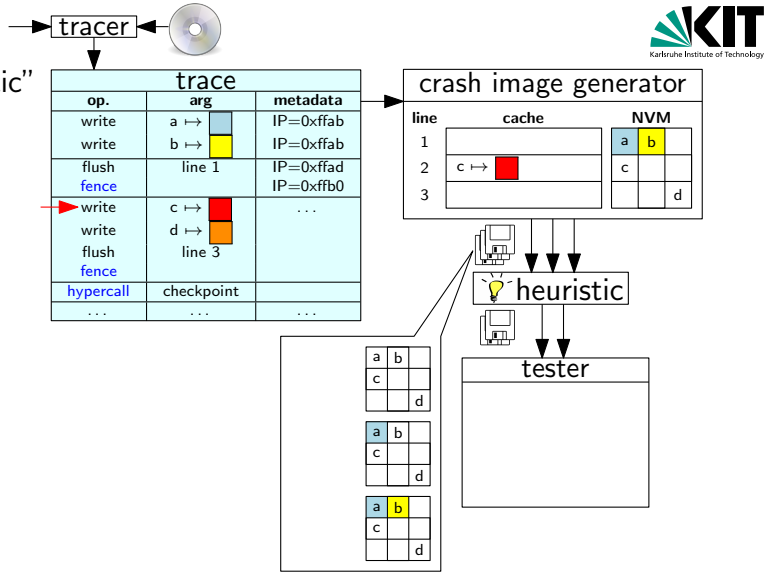
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

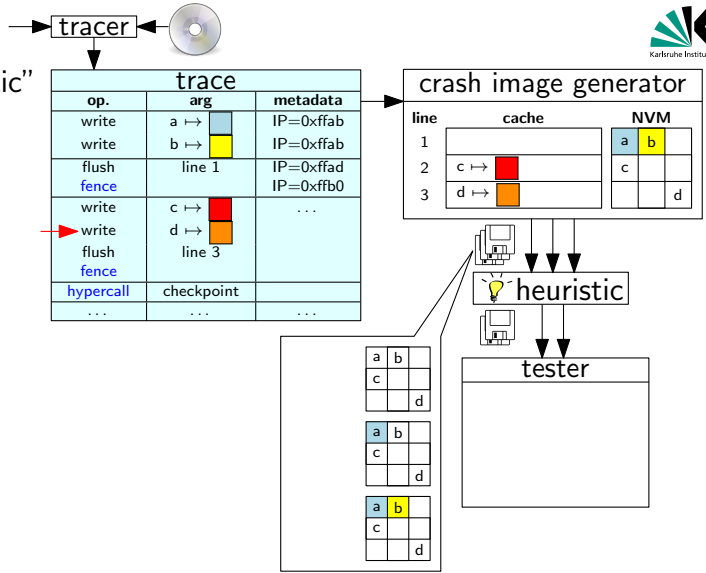
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

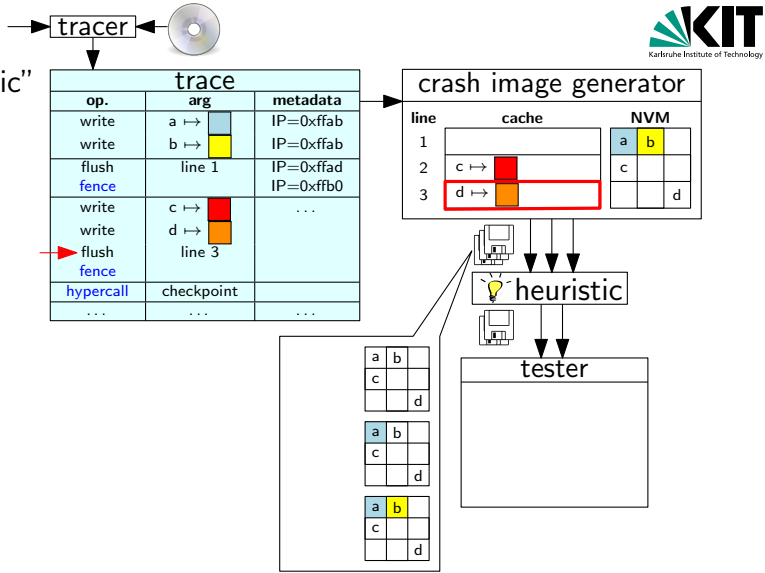
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

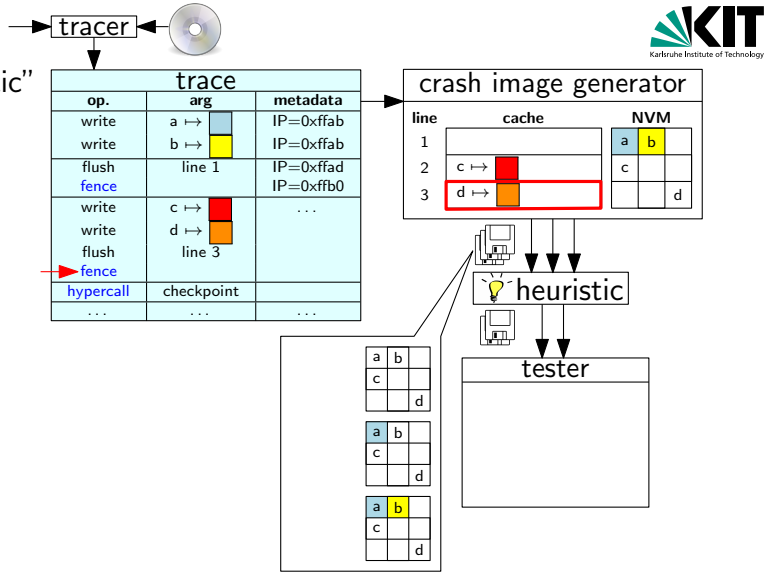
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

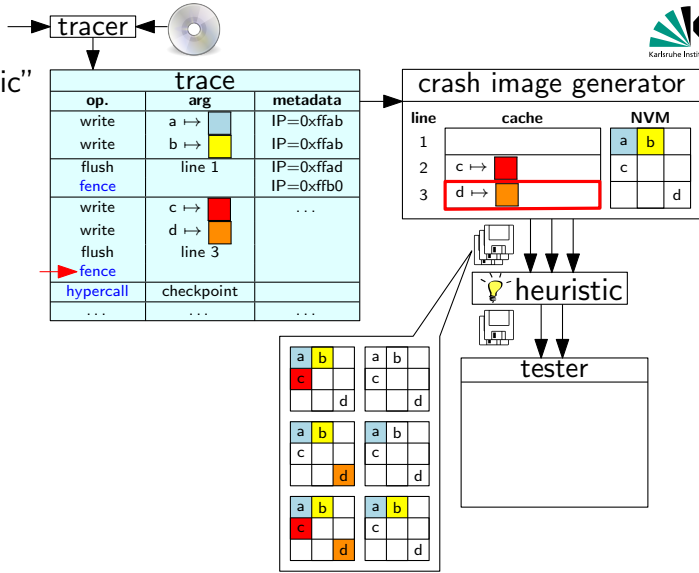
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

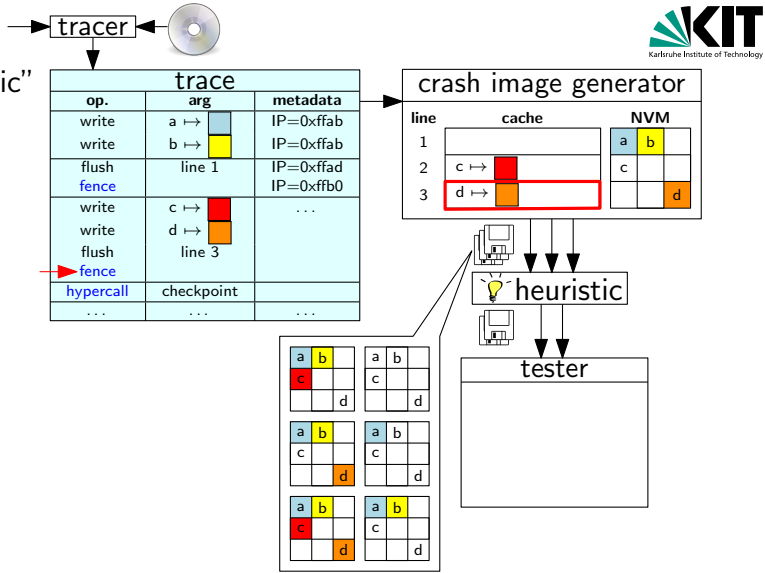
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

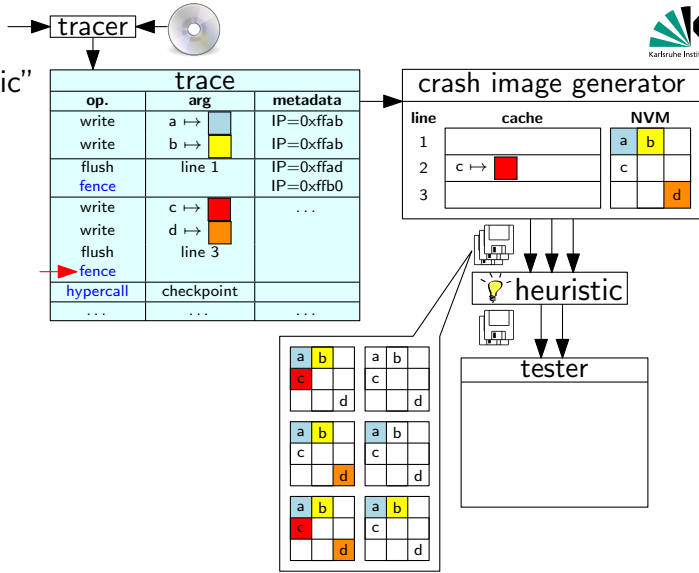
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

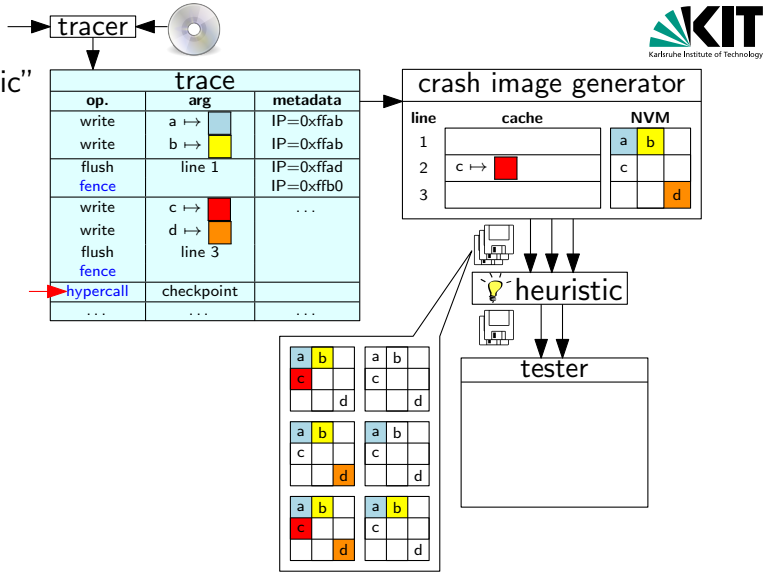
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

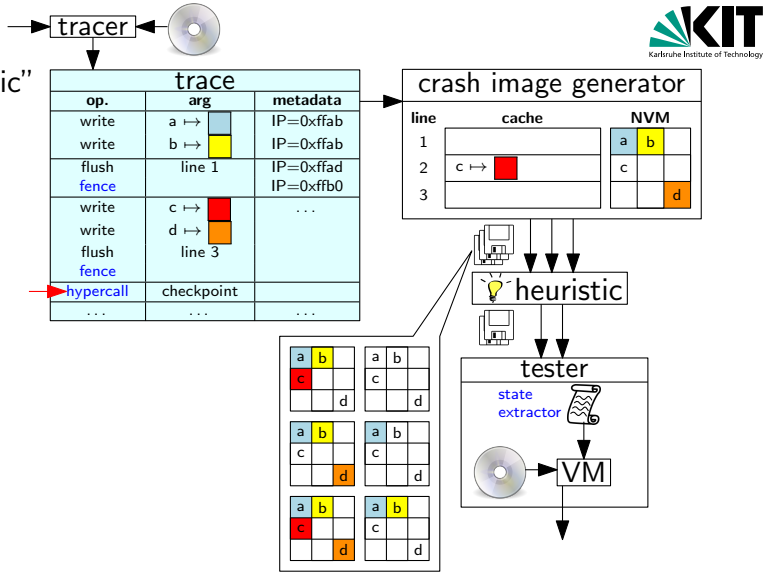
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

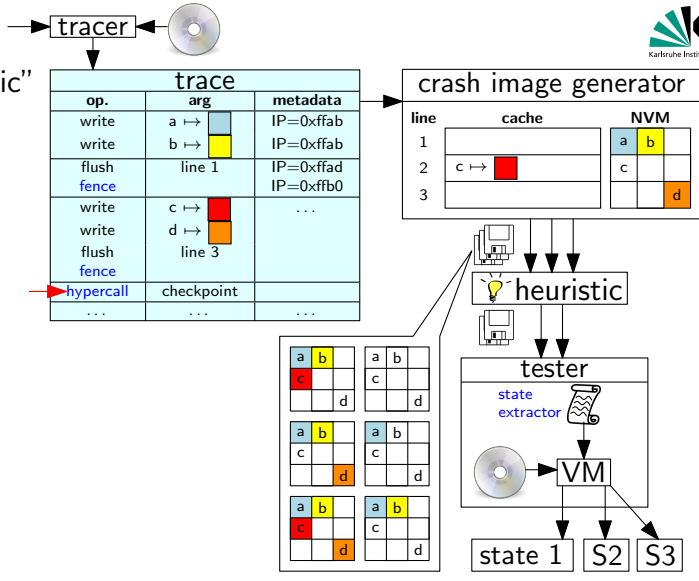
tracer:
full system emulation with
dynamic binary translation



Approach

"generic, efficient, automatic"

tracer:
full system emulation with
dynamic binary translation



Evaluation: In-Kernel NVM file systems

NOVA: Found bug candidates

NOVA is a state-of-the-art NVM-specific file system

`echo HelloWorld > /mnt/file && sync`

Observed intermediate states: no file, empty file, partial file, full file $\rightsquigarrow$ expected

However, after `sync`, two states possible:


```
{
  ...
  "/mnt/file":
  {
    "content":
      "HelloWorld\n",
    "st_atim_nsec": 0,
    "st_atim_sec": 1630407258,
    "st_blksize": 4096,
    "st_blocks": 8,
    "st_ctim_nsec": 0,
    "st_ctim_sec": 1630407258,
    "st_gid": 0,
    "st_ino": 33,
    "st_mode": 33188,
    "st_mtim_nsec": 0,
    "st_mtim_sec": 1630407258,
    ...
```

```
{
  ...
  "/mnt/file":
  {
    "content":
      "HelloWor\u0000\u0000\u0000",
    "st_atim_nsec": 0,
    "st_atim_sec": 1630407258,
    "st_blksize": 4096,
    "st_blocks": 8,
    "st_ctim_nsec": 0,
    "st_ctim_sec": 1630407258,
    "st_gid": 0,
    "st_ino": 33,
    "st_mode": 33188,
    "st_mtim_nsec": 0,
    "st_mtim_sec": 1630407258,
    ...
```

Evaluation: In-Kernel NVM file systems

NOVA: Found bug candidates

```
static inline int memcpy_to_pmem_nocache
    (void *dst, const void *src, unsigned int size)
{
     int ret;

     ret = __copy_from_user_inatomic_nocache(dst, src, size);

     return ret;
}
```

arch/x86/lib/copy_user_64.S:

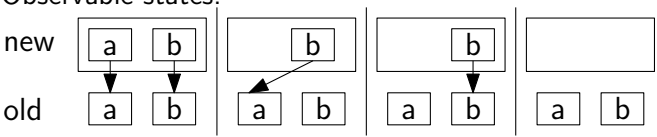
```
/*
 * copy_user_nocache - Uncached memory copy with exception handling
 * This will force destination out of cache for more performance.
 *
 * Note: Cached memory copy is used when destination or size is not
 * naturally aligned. That is:
 *  - Require 8-byte alignment when size is 8 bytes or larger.
 *  - Require 4-byte alignment when size is 4 bytes.
 */
```

Evaluation: In-Kernel NVM file systems

NOVA: Found bug candidates

Renaming files: given files *a*, *b*: `mv a b`

- Observable states:



- Can even lose whole directory trees

```
mv testfile testfile_renamed_to_a_long_filename0123456789...
```

- Intermediate state:
  `stat` on renamed file fails

Conclusion

- Crash consistency testing of full systems using dynamic binary translation
- Heuristics to choose crash images
- Automatic testing for consistency and atomicity
- We find bugs in a state-of-the-art NVM file system

Future Work

- Apply more advanced heuristics
- Approach applicable even to traditional file systems
  - Advantages over CrashMonkey ('19):
    - \+ No code modification
    - \+ Crashing in the middle of operations, write reordering