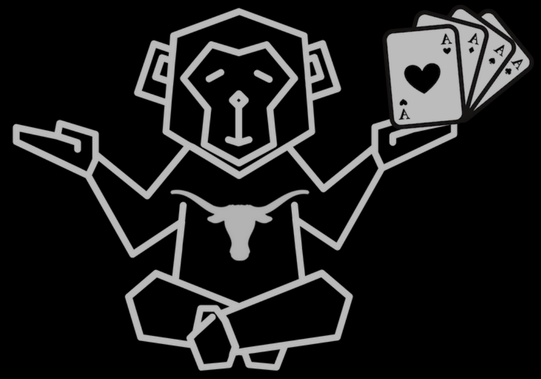


# CRASH MONKEY & ACE

## Systematically Testing File-System Crash Consistency

Jayashree Mohan, Ashlie Martinez, Soujanya Ponnapalli,
Pandian Raju, Vijay Chidambaram

**[Published at OSDI 2018]**



# Crashes

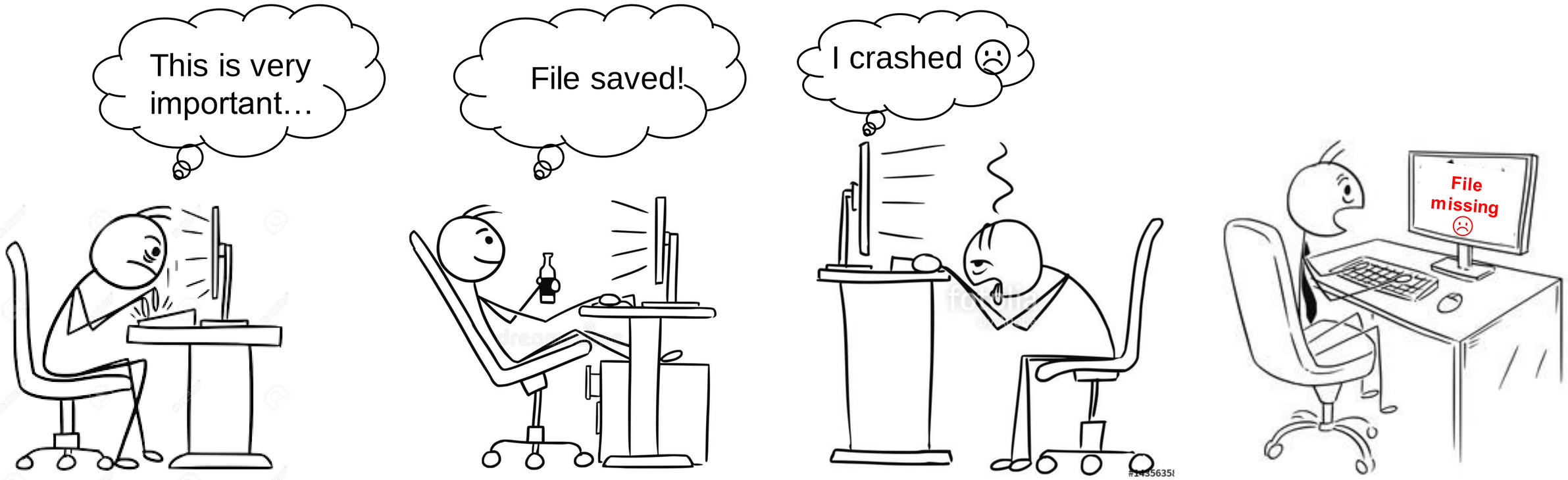
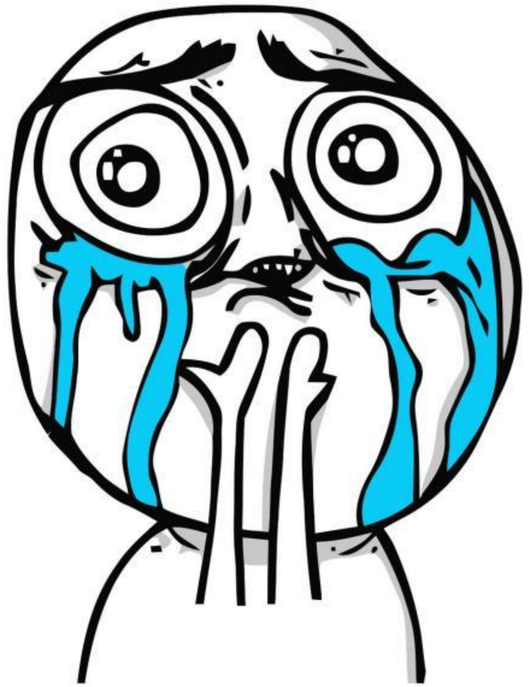



Image source : https://www.fotolia.com

**I wish filesystems were crash-consistent!**

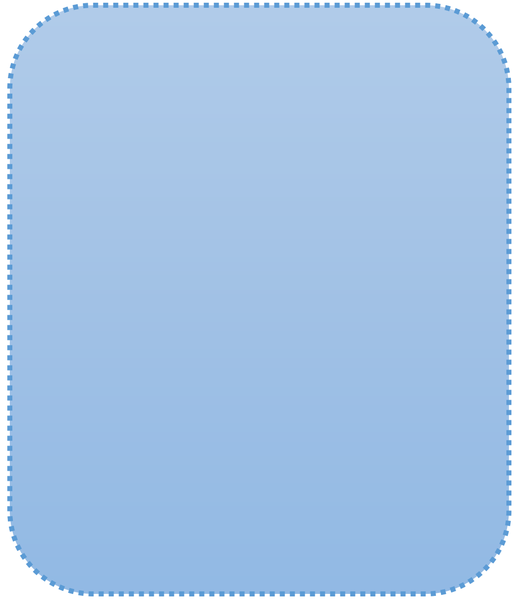


# Rename atomicity bug in btrfs

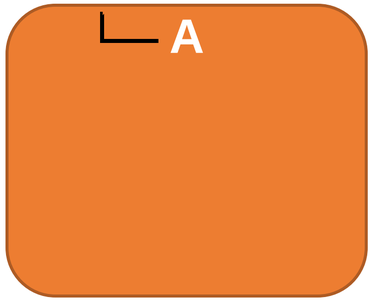
Memory

---

Storage

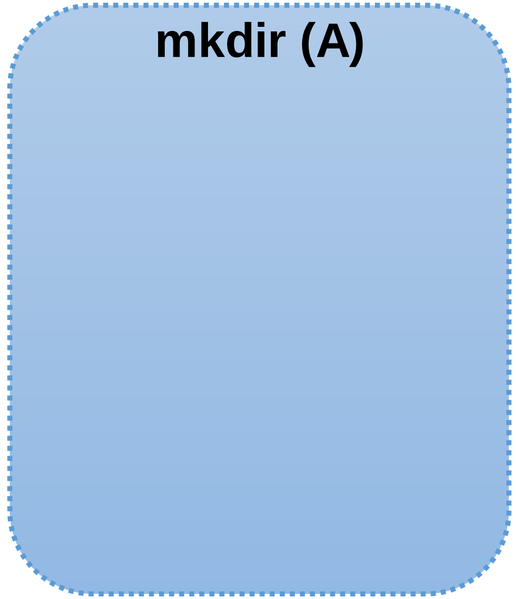


# Rename atomicity bug in btrfs

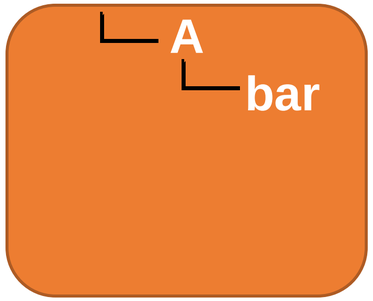



Memory

Storage



# Rename atomicity bug in btrfs



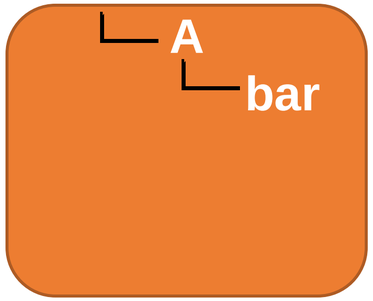

Memory

---

Storage

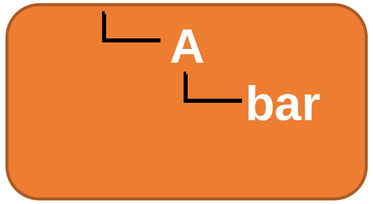
**mkdir (A)**
**touch (A/bar)**

# Rename atomicity bug in btrfs



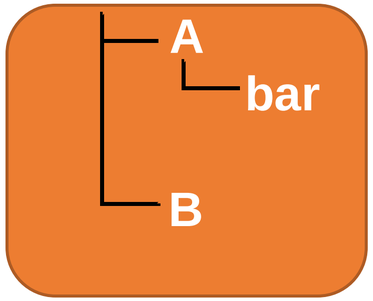

Memory

Storage



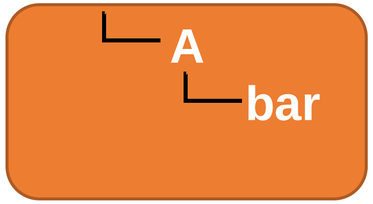

**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**

# Rename atomicity bug in btrfs



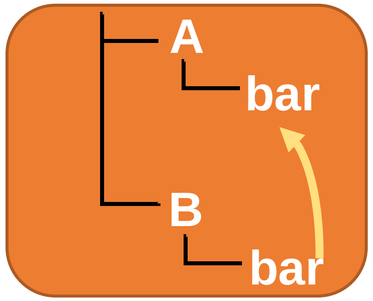

Memory

Storage




**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**

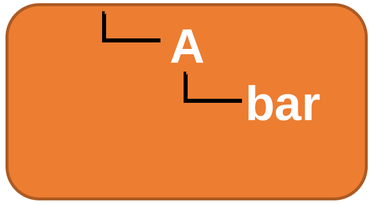
# Rename atomicity bug in btrfs




Memory

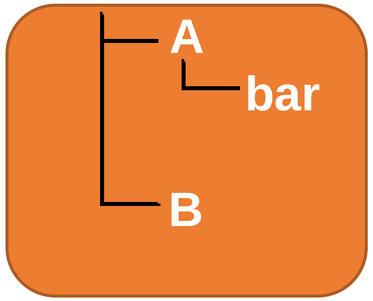
---

Storage




**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**

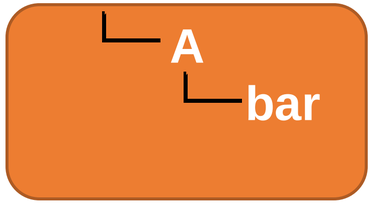
# Rename atomicity bug in btrfs




Memory

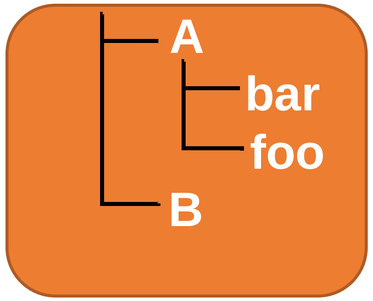
---

Storage




**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**
**rename (B/bar, A/bar)**

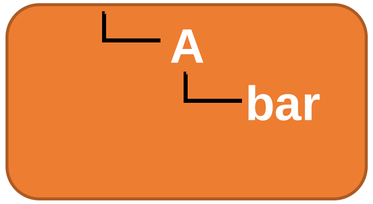
# Rename atomicity bug in btrfs




Memory

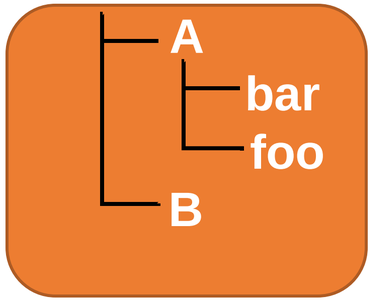
---

Storage




**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**
**rename (B/bar, A/bar)**
**touch (A/foo)**

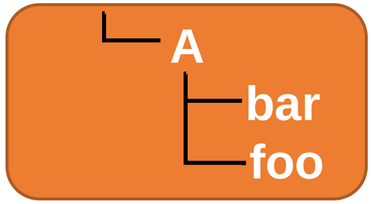
# Rename atomicity bug in btrfs




Memory

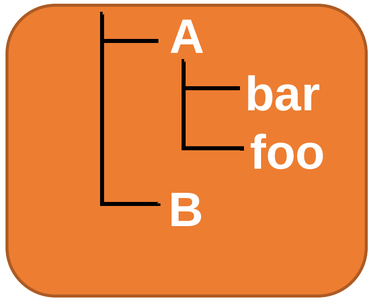
---

Storage




**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**
**rename (B/bar, A/bar)**
**touch (A/foo)**
**fsync (A/foo)**

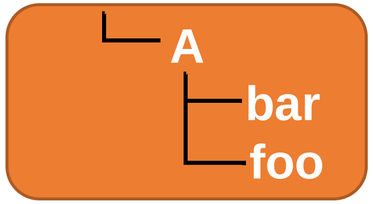
# Rename atomicity bug in btrfs




Memory

---

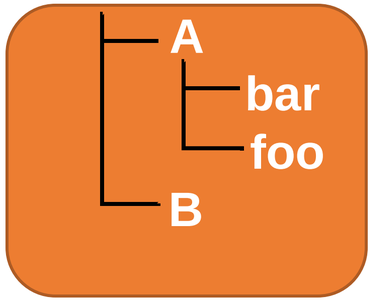
Storage




**Expected**

**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**
**rename (B/bar, A/bar)**
**touch (A/foo)**
**fsync (A/foo)**
**CRASH!**

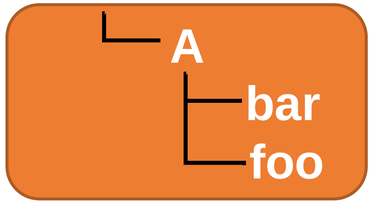
# Rename atomicity bug in btrfs



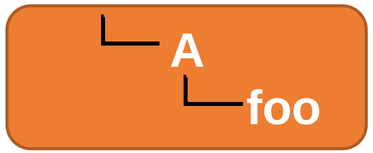

Memory

Storage

**Persisted file A/bar missing**



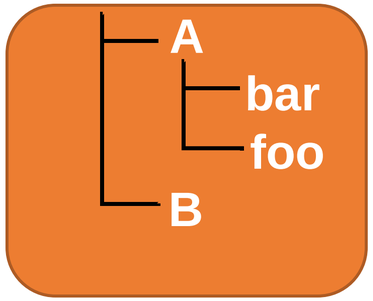

**Expected**




**Actual**

**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**
**rename (B/bar, A/bar)**
**touch (A/foo)**
**fsync (A/foo)**
**CRASH!**

# Rename atomicity bug in btrfs




Memory

Storage

**Persisted file A/bar missing**

**mkdir (A)**
**touch (A/bar)**
**fsync (A/bar)**
**mkdir (B)**
**touch (B/bar)**
**rename (B/bar, A/bar)**
**touch (A/foo)**
**fsync (A/foo)**
**CRASH!**

**Exists in the kernel since 2014!**
**Found by ACE and CrashMonkey**

# Testing Crash Consistency Today

- Build FS from scratch

**Verified Filesystems**

**Model Checking**

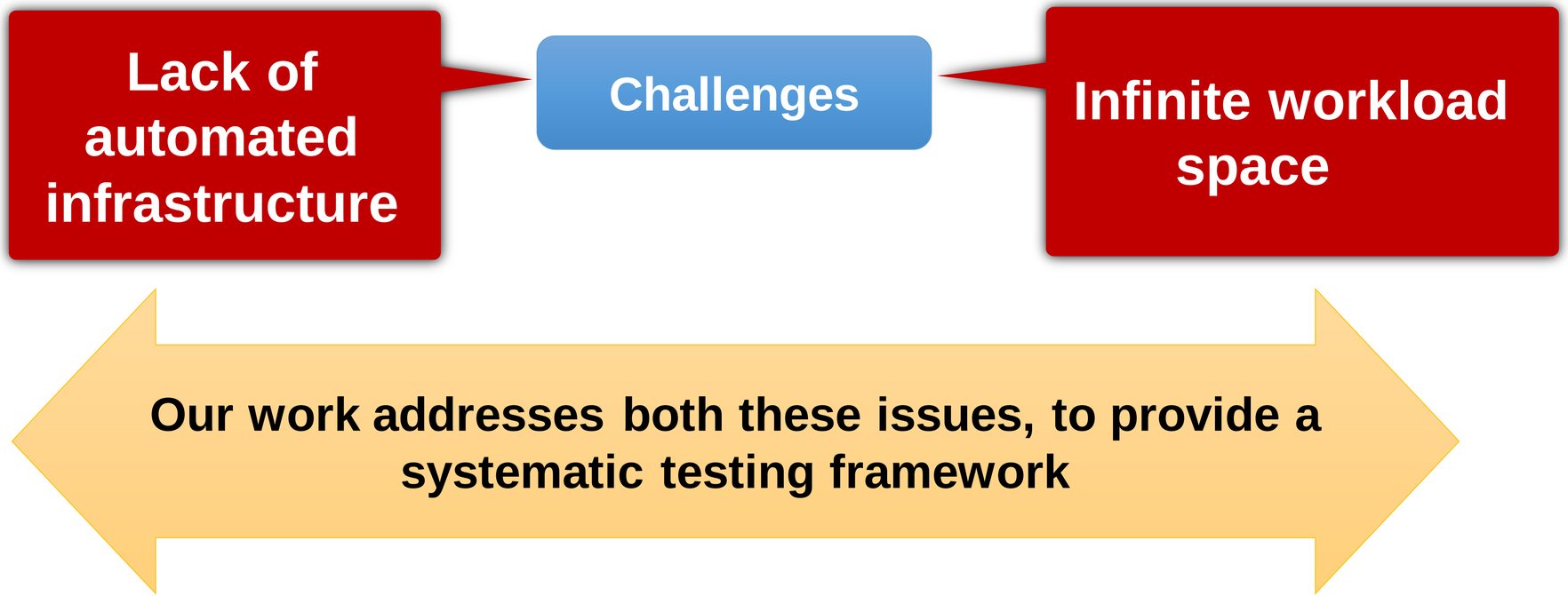
Annotate filesystems
Hard to do for existing FS

- State of the Art : xfstest suite
  - Collection of 482 regression tests

**Only 5% of tests in xfstest check for file system crash consistency**

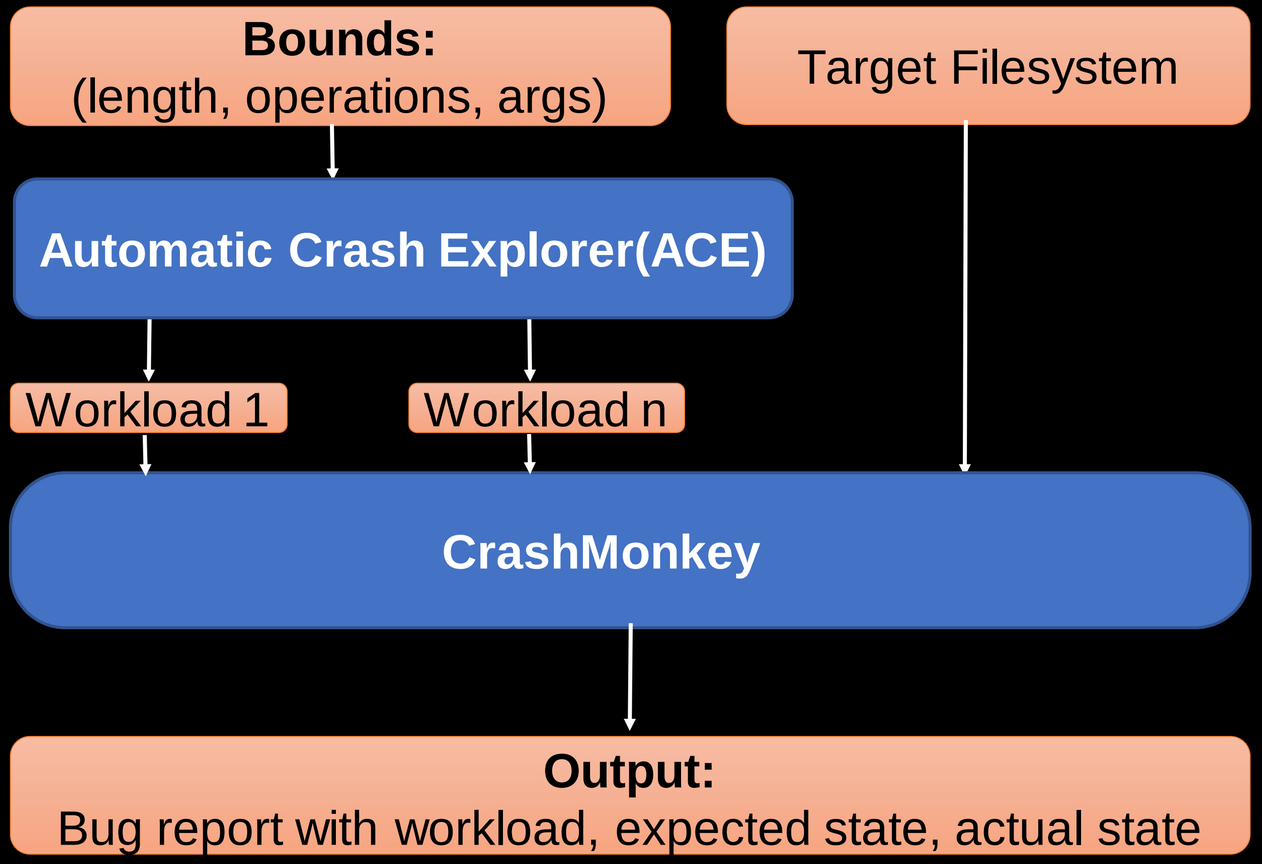
# Challenges with systematic testing

Systematically generate workloads



# Bounded Black-Box Crash Testing ($B^3$)

New approach to testing file-system crash consistency




- Focus on reproducible bugs resulting in metadata corruption, data loss.
- Found 10 new bugs across btrfs and F2FS;
- Found 1 bug in FSCQ (verified file system)

www.github.com/utsaslab/crashmonkey

# CrashMonkey and Ace : Features

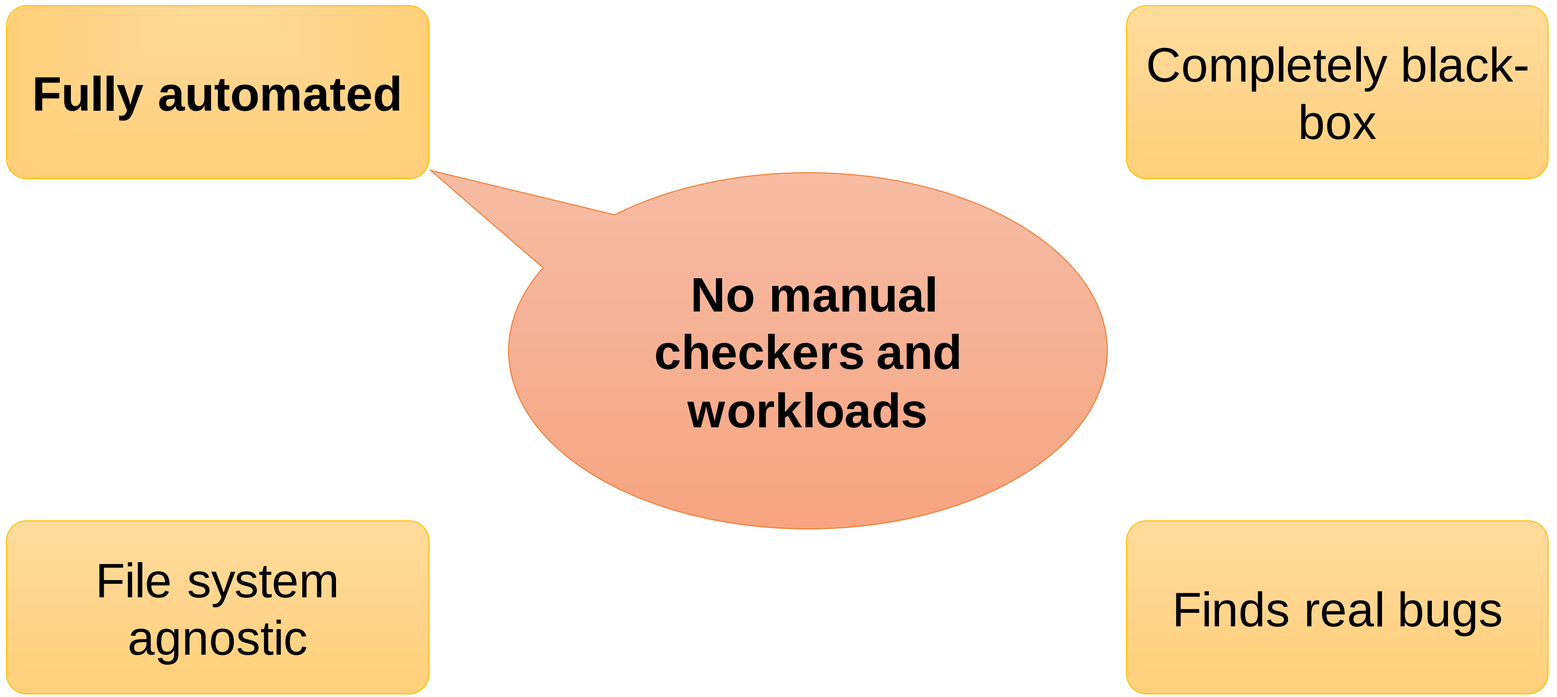
**Fully automated**

**Completely black-box**

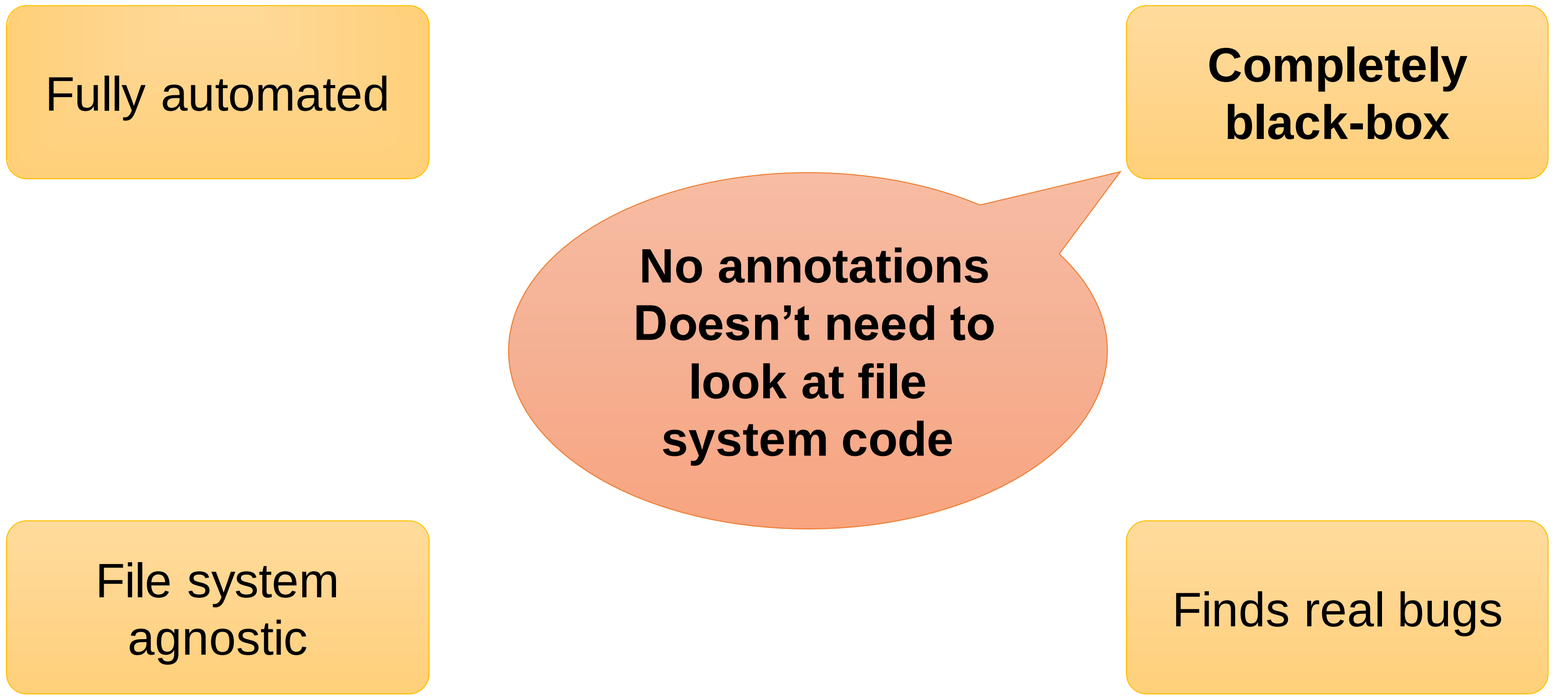
**File system agnostic**

**Finds real bugs**

# CrashMonkey and Ace : Features



# CrashMonkey and Ace : Features



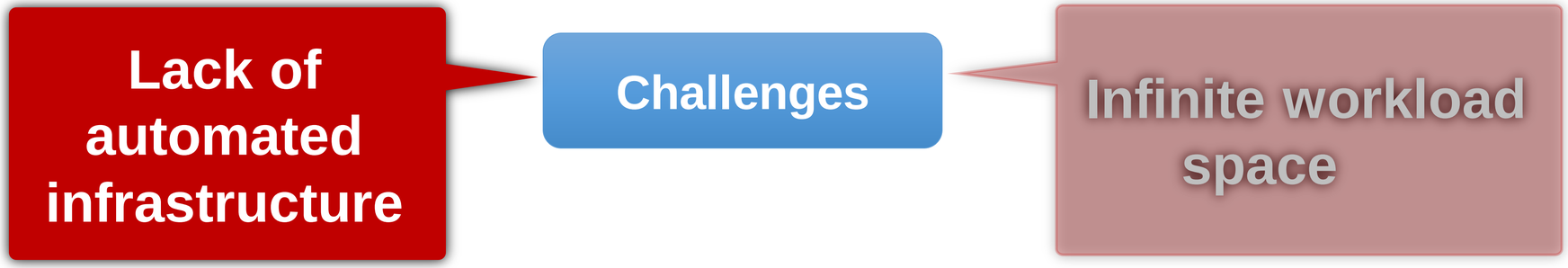
# CrashMonkey and Ace : Features

# CrashMonkey and Ace : Features

# Outline

- CrashMonkey
- Bounded Black Box Crash Testing
- Automatic Crash Explorer (ACE)
- Results

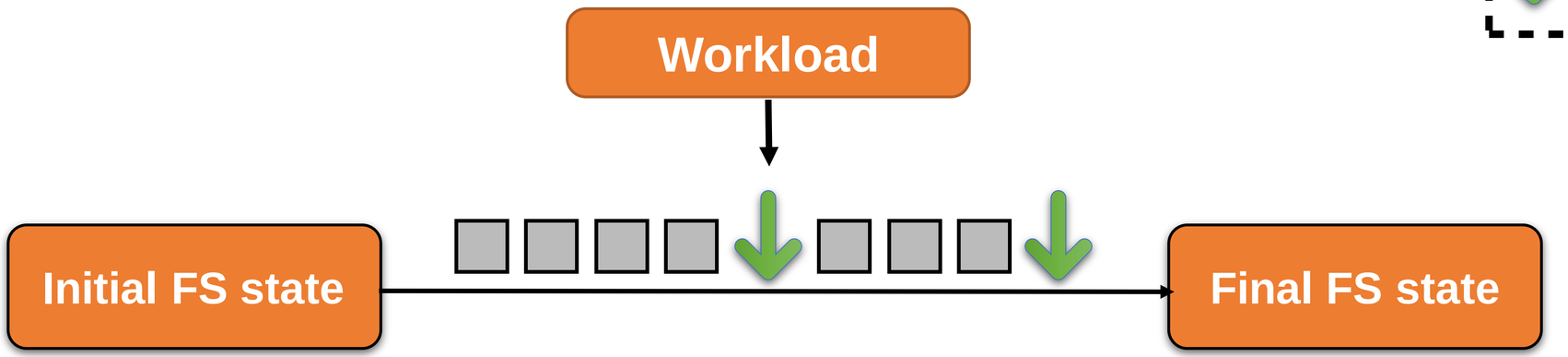
# Challenges with systematic testing

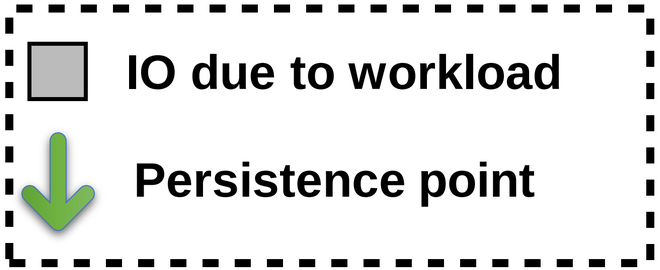


# CrashMonkey

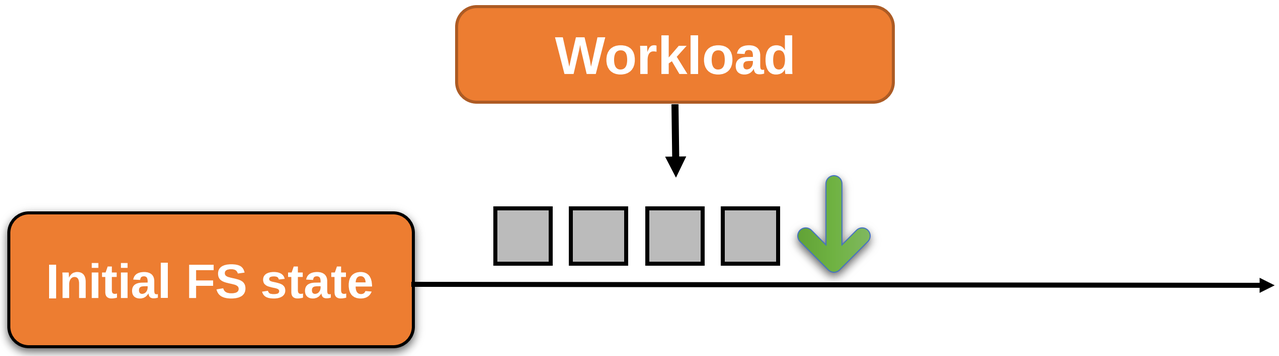
- Efficient infrastructure to record and replay block level IO requests
- Simulate crash at different points in the workload
- Automatically test for consistency after crash.
- Copy-on-write RAM block device

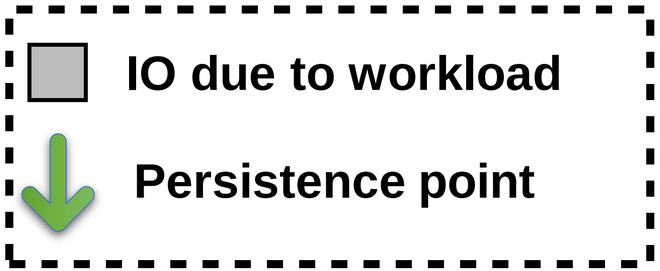
# CrashMonkey in Action



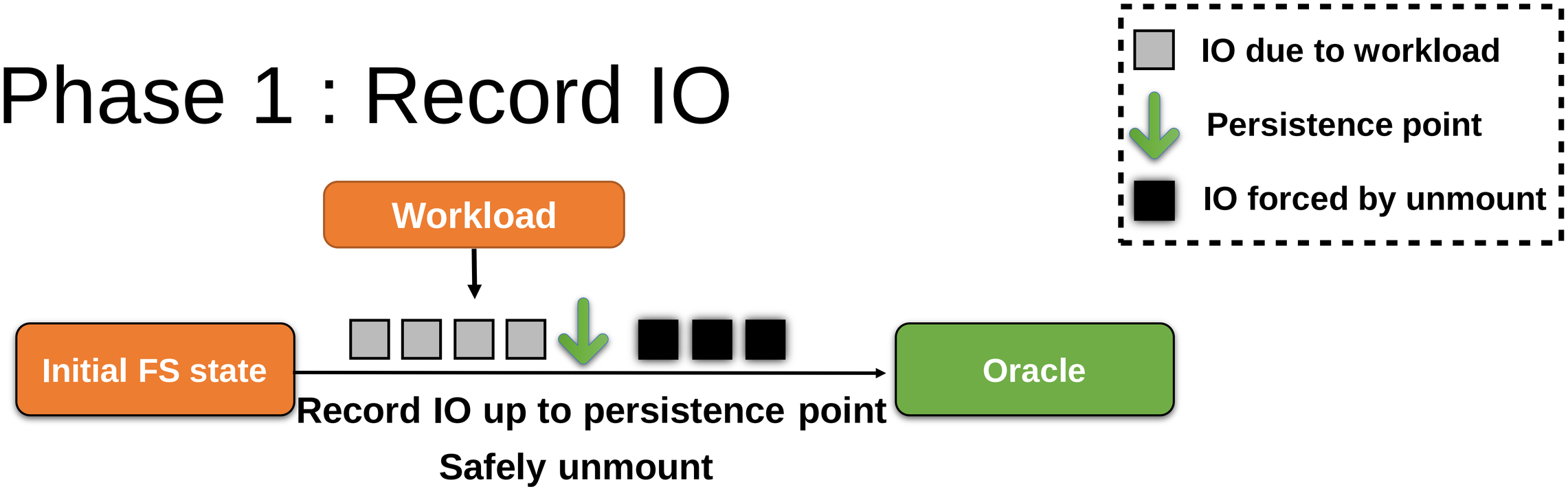



# CrashMonkey in Action

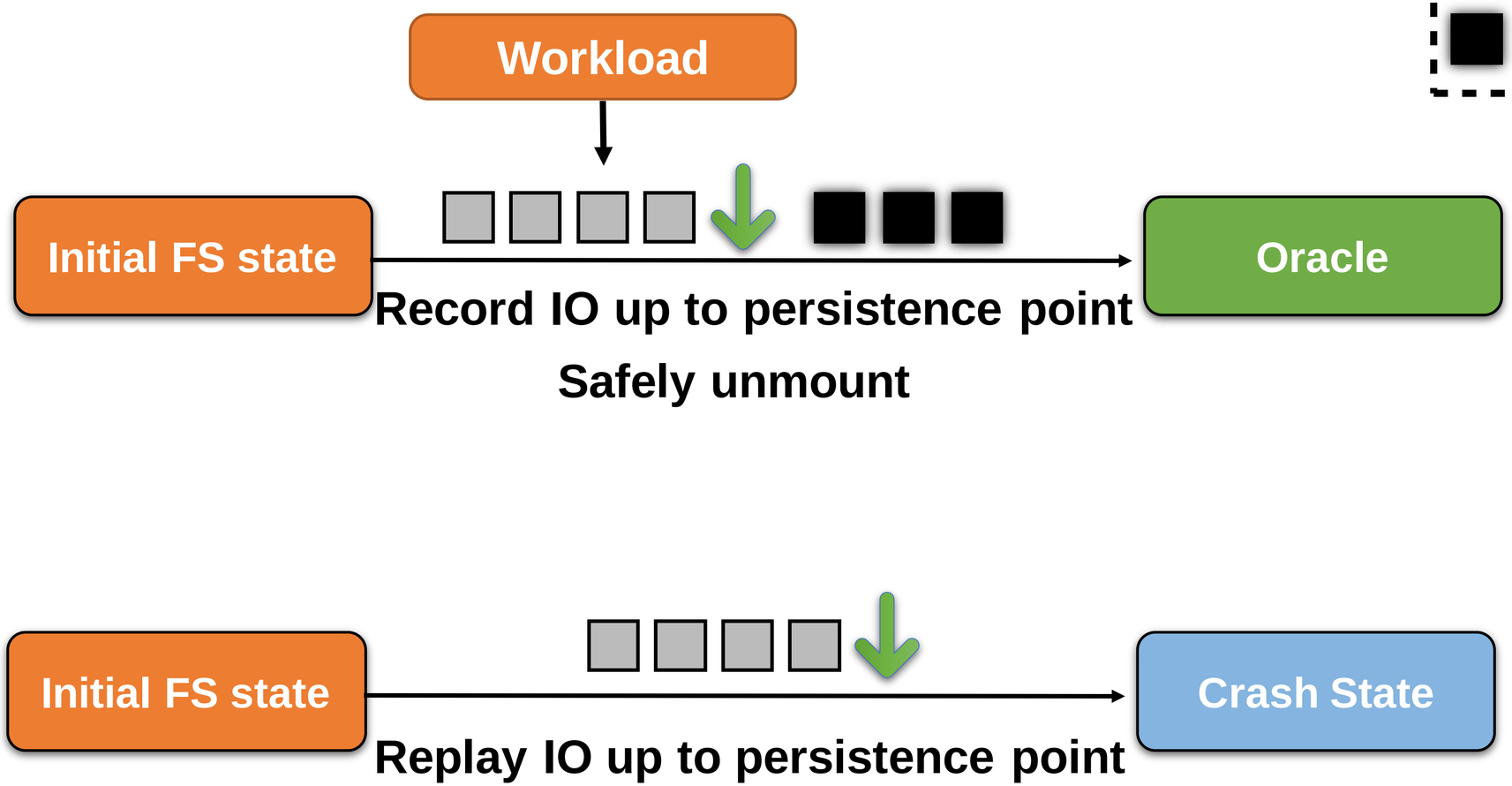


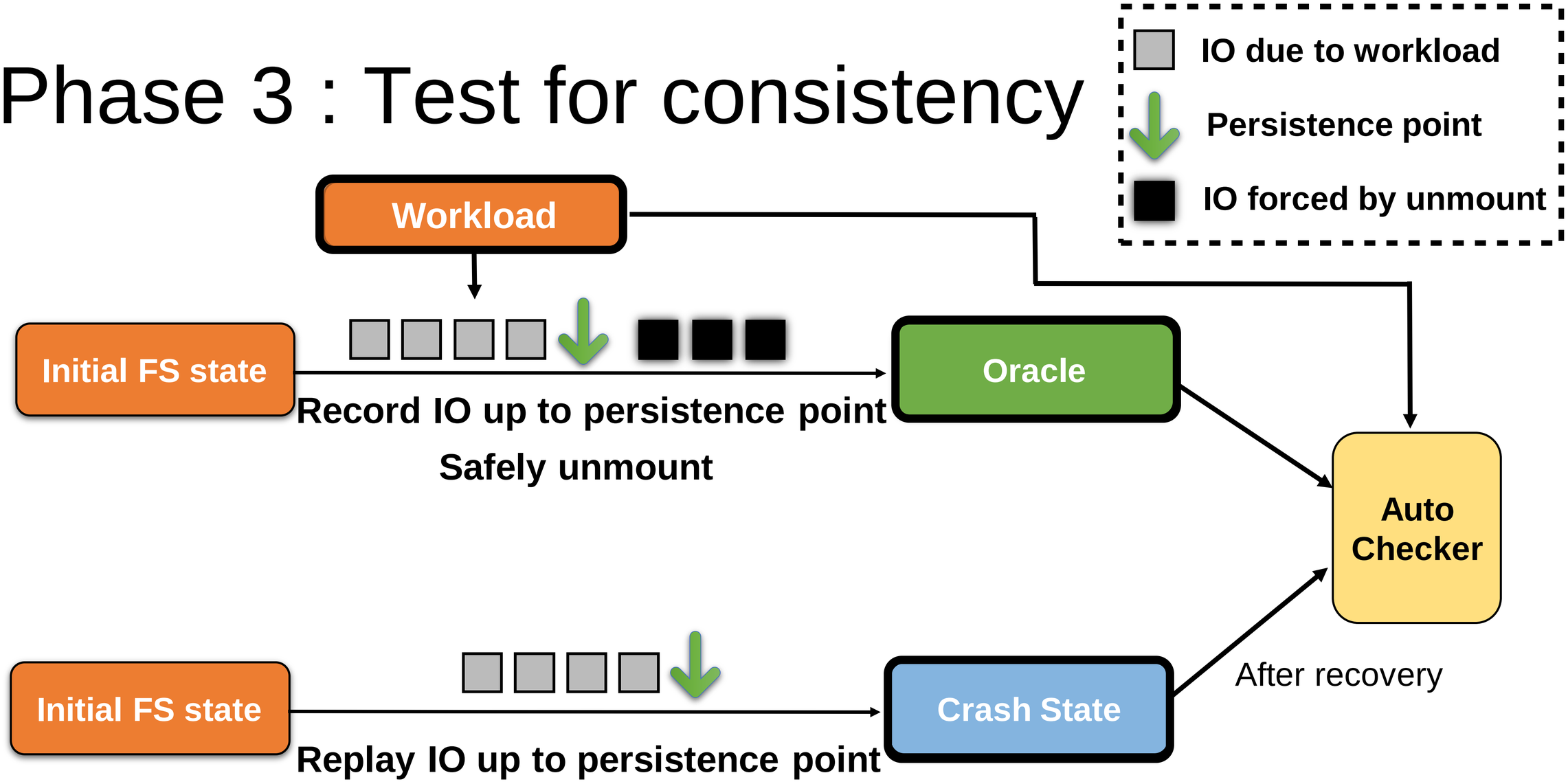
# Phase 1 : Record IO



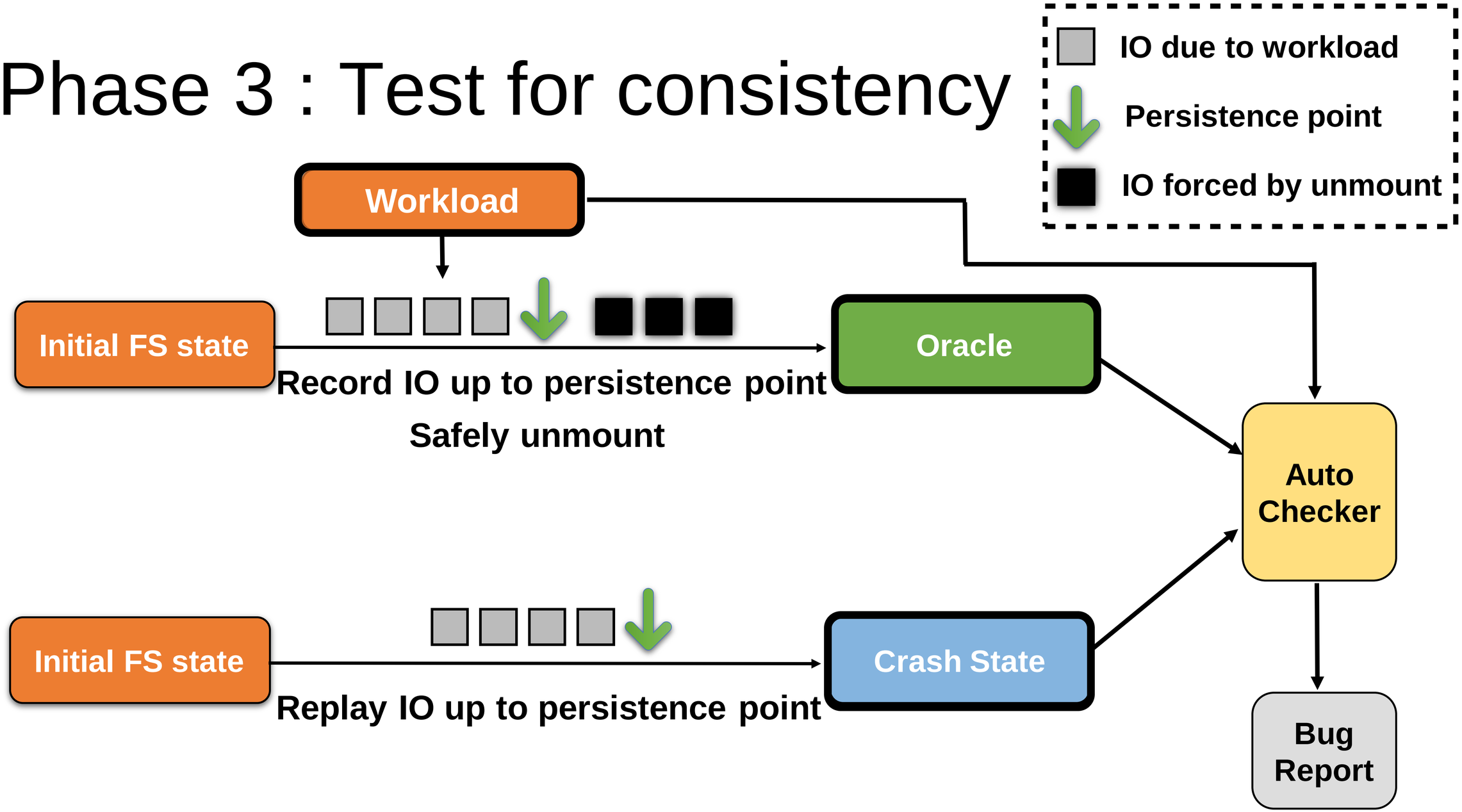
# Phase 2 : Replay IO



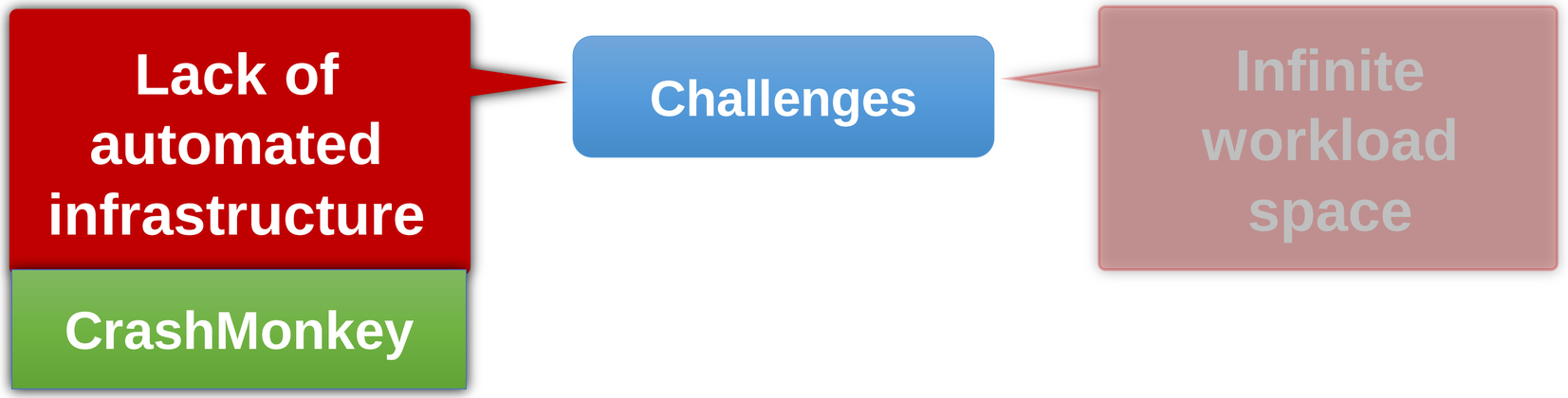
# Phase 3 : Test for consistency



# Phase 3 : Test for consistency



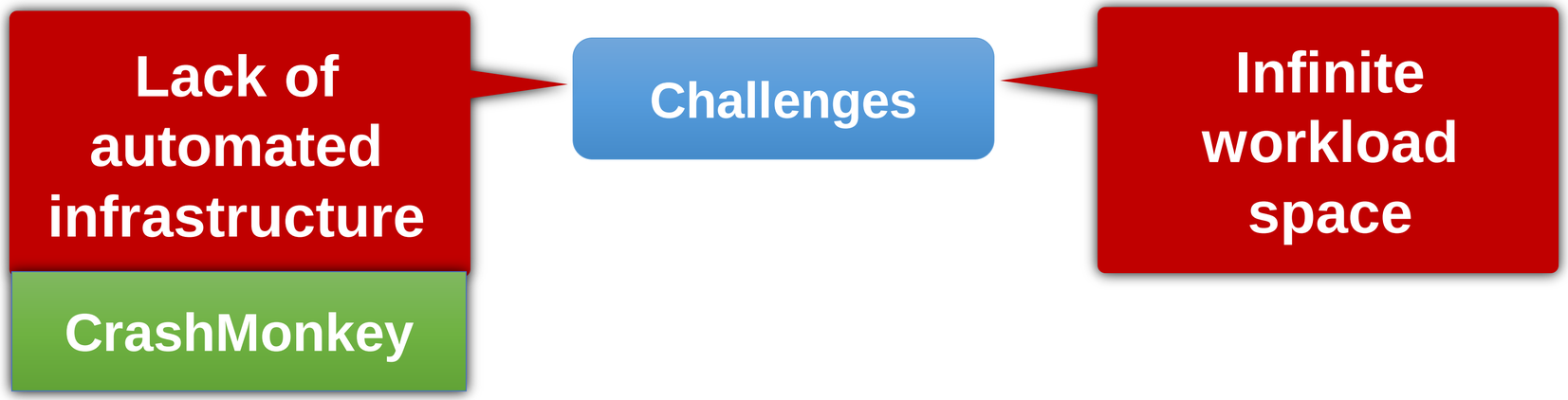
# Challenges with Systematic Testing




So Far…

- Given a workload compliant to POSIX API, we saw how CrashMonkey generates crash states and automatically tests for consistency

# Challenges with Systematic Testing




So Far…

- Given a workload compliant to POSIX API, we saw how CrashMonkey generates crash states and automatically tests for consistency

- Next question : How to automatically generate workloads in an the infinite workload space?

# Exploring the infinite workload space

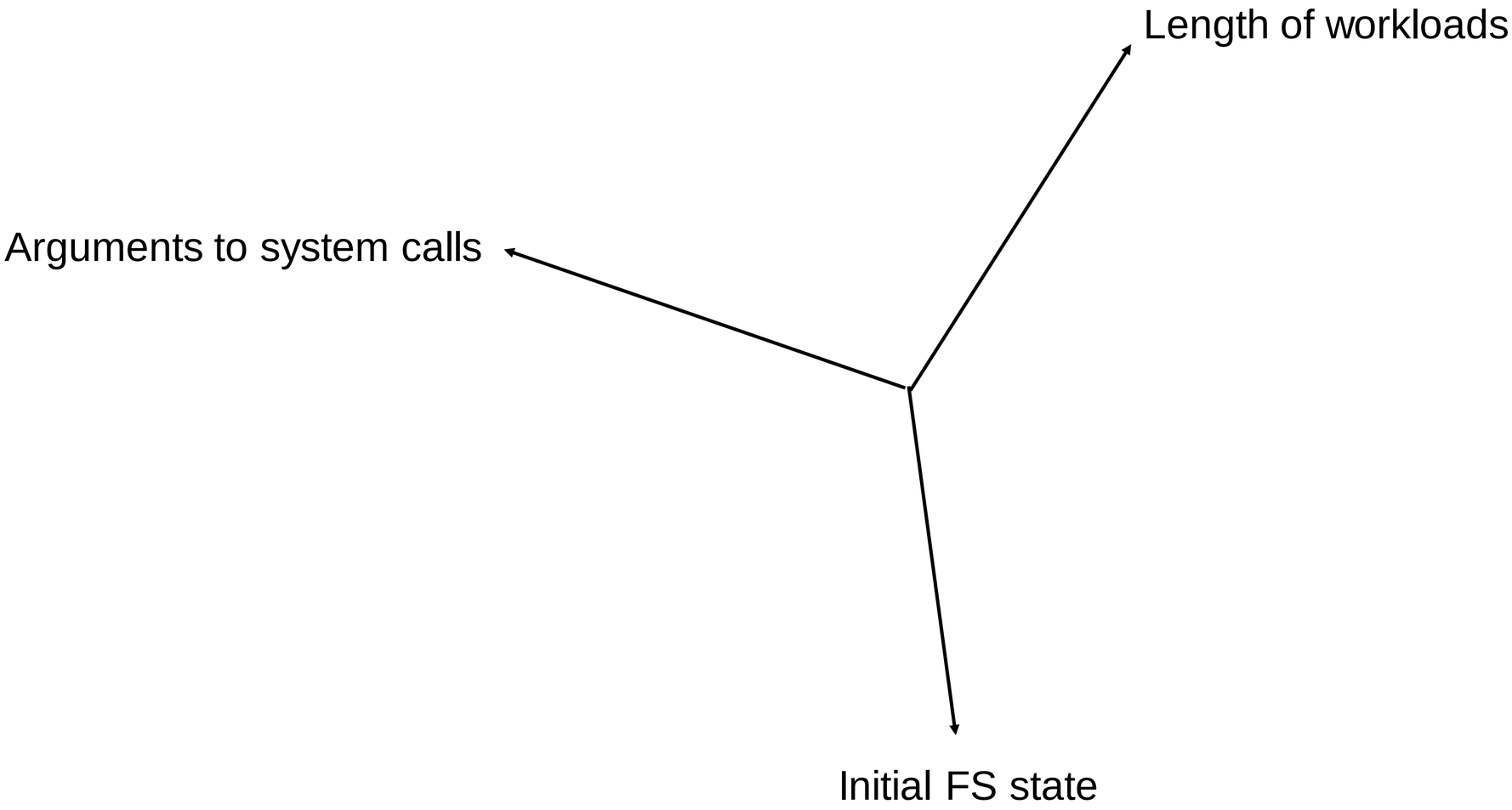
**Challenges:**

- Infinite length of workloads
- Large set of filesystem operations
- Infinite parameter options (file/directory names, depth)
- Infinite options for initial filesystem state
- When in the workload to simulate a crash?

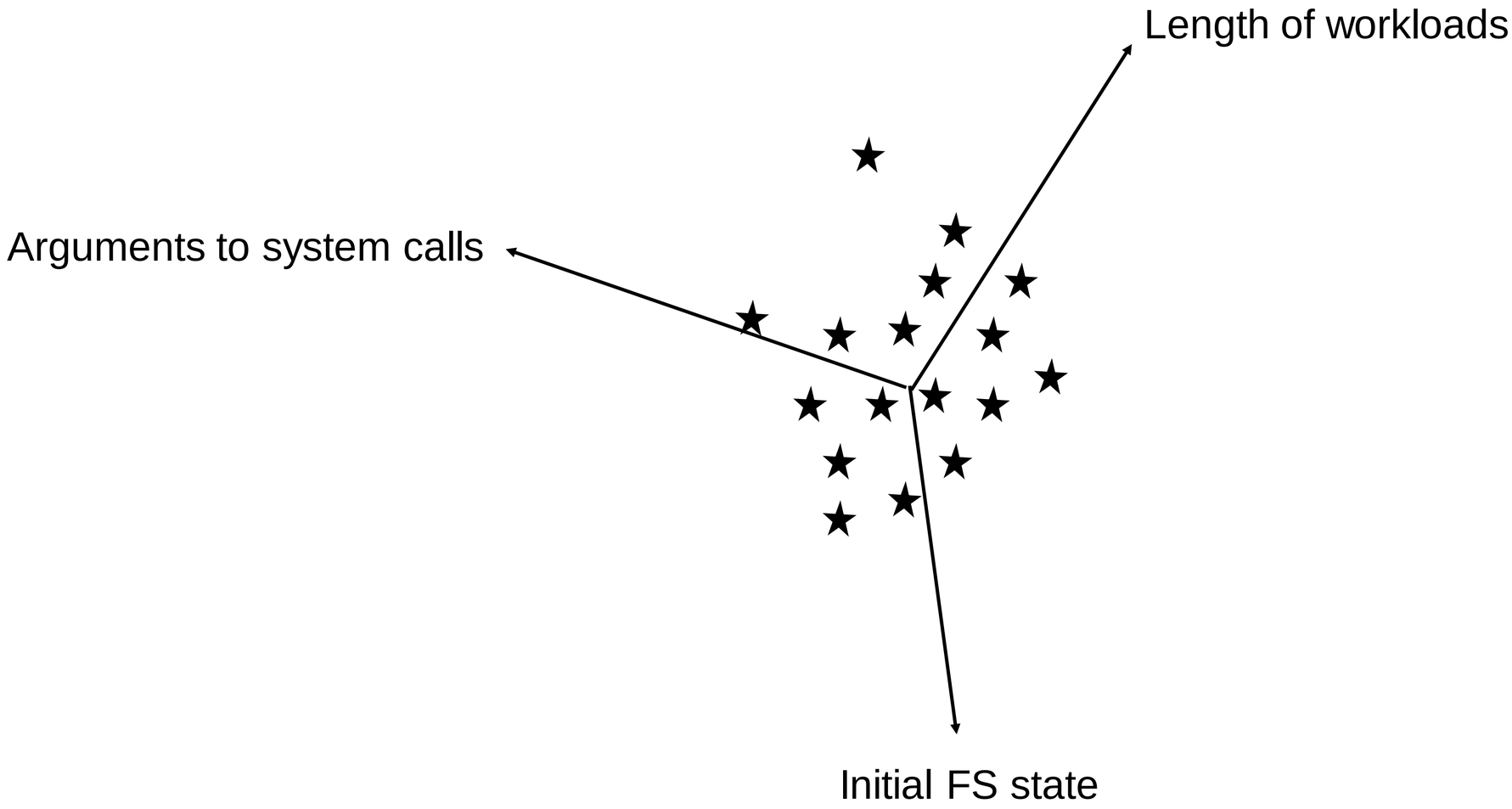
# Outline

- CrashMonkey
- **Bounded Black Box Crash Testing**
- Automatic Crash Explorer (ACE)
- Results

# $B^3$ : Bounded Black Box Crash Testing



# $B^3$ : Bounded Black Box Crash Testing



# $B^3$ : Bounded Black Box Crash Testing

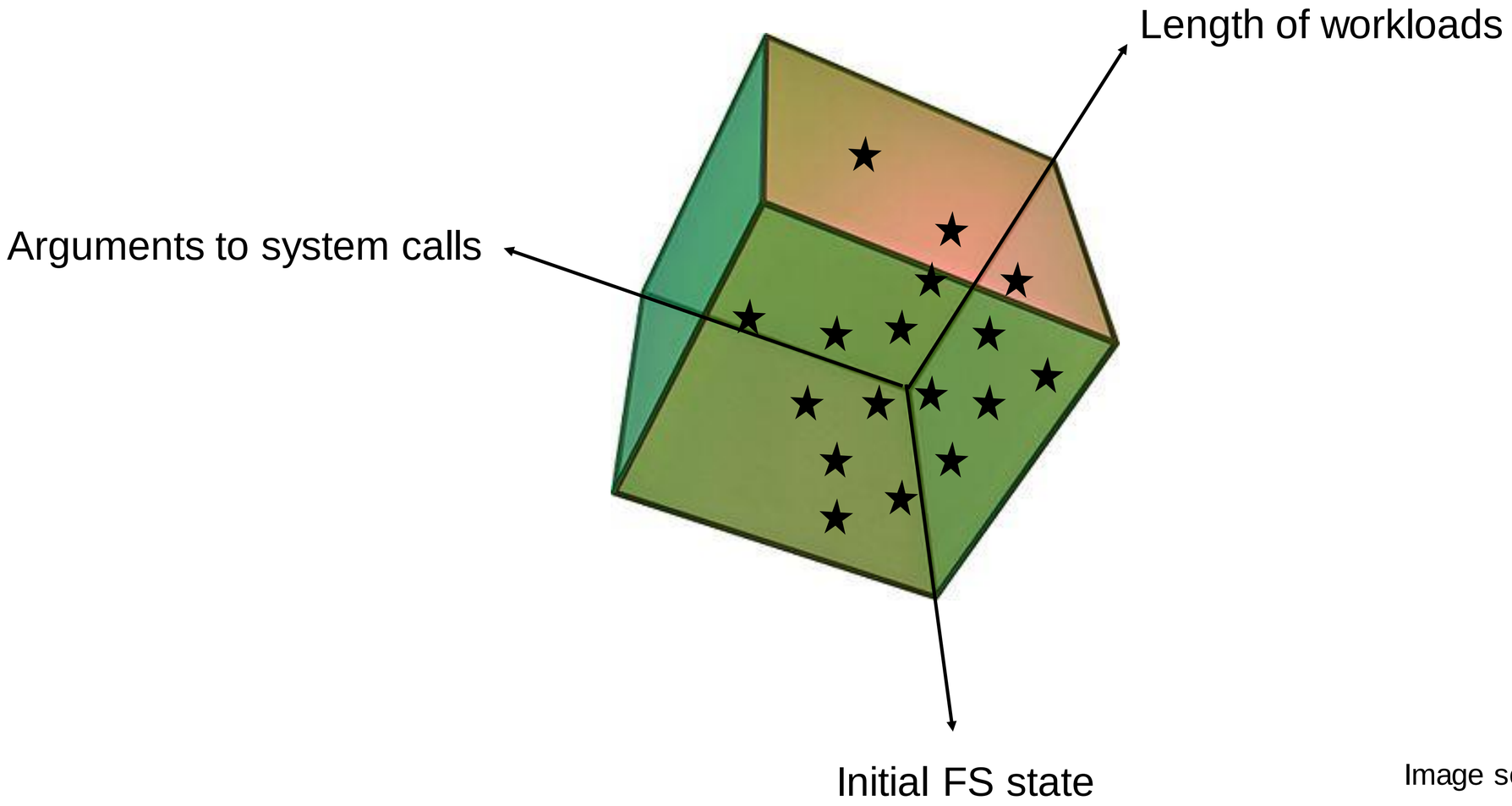
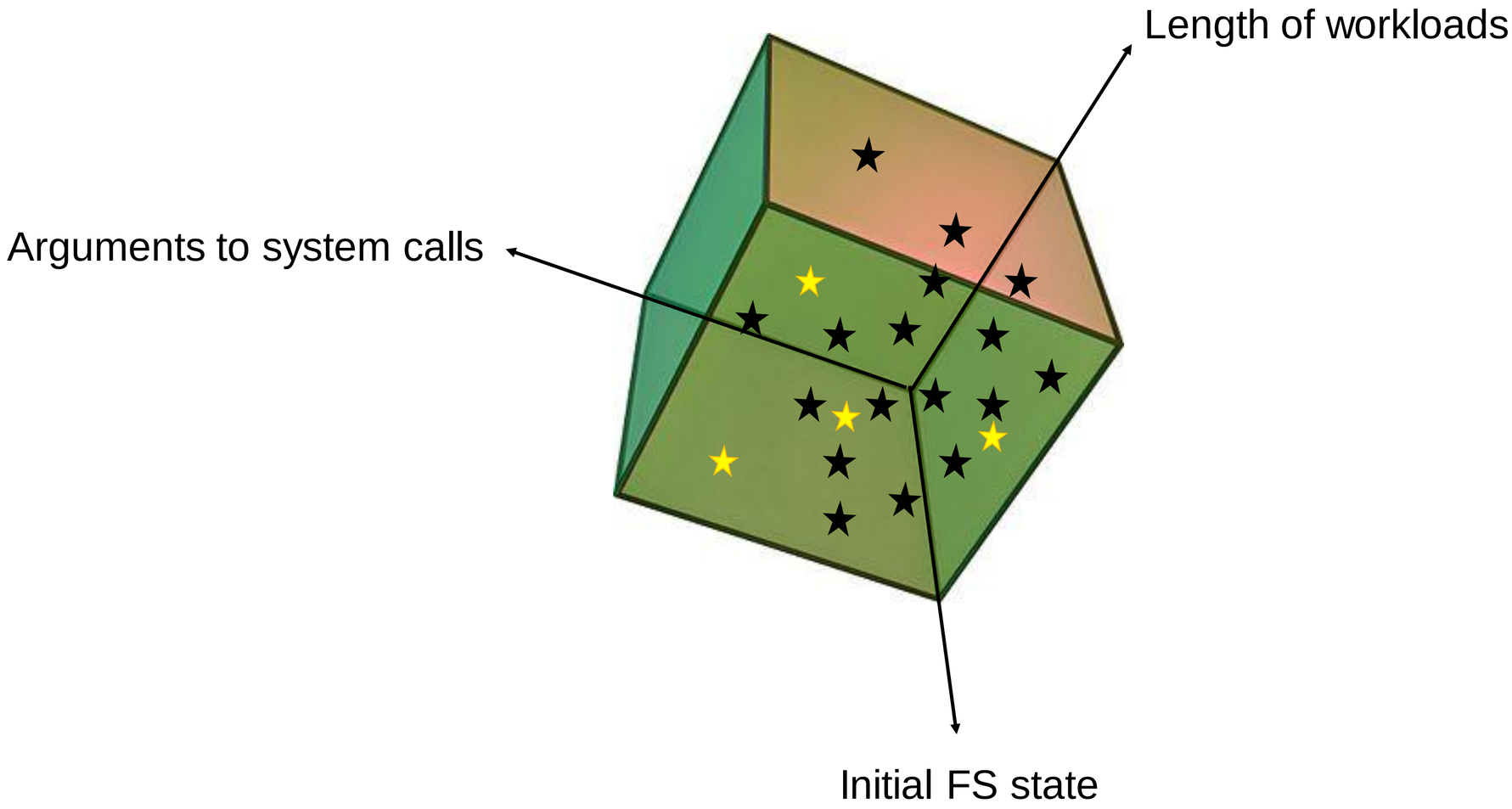
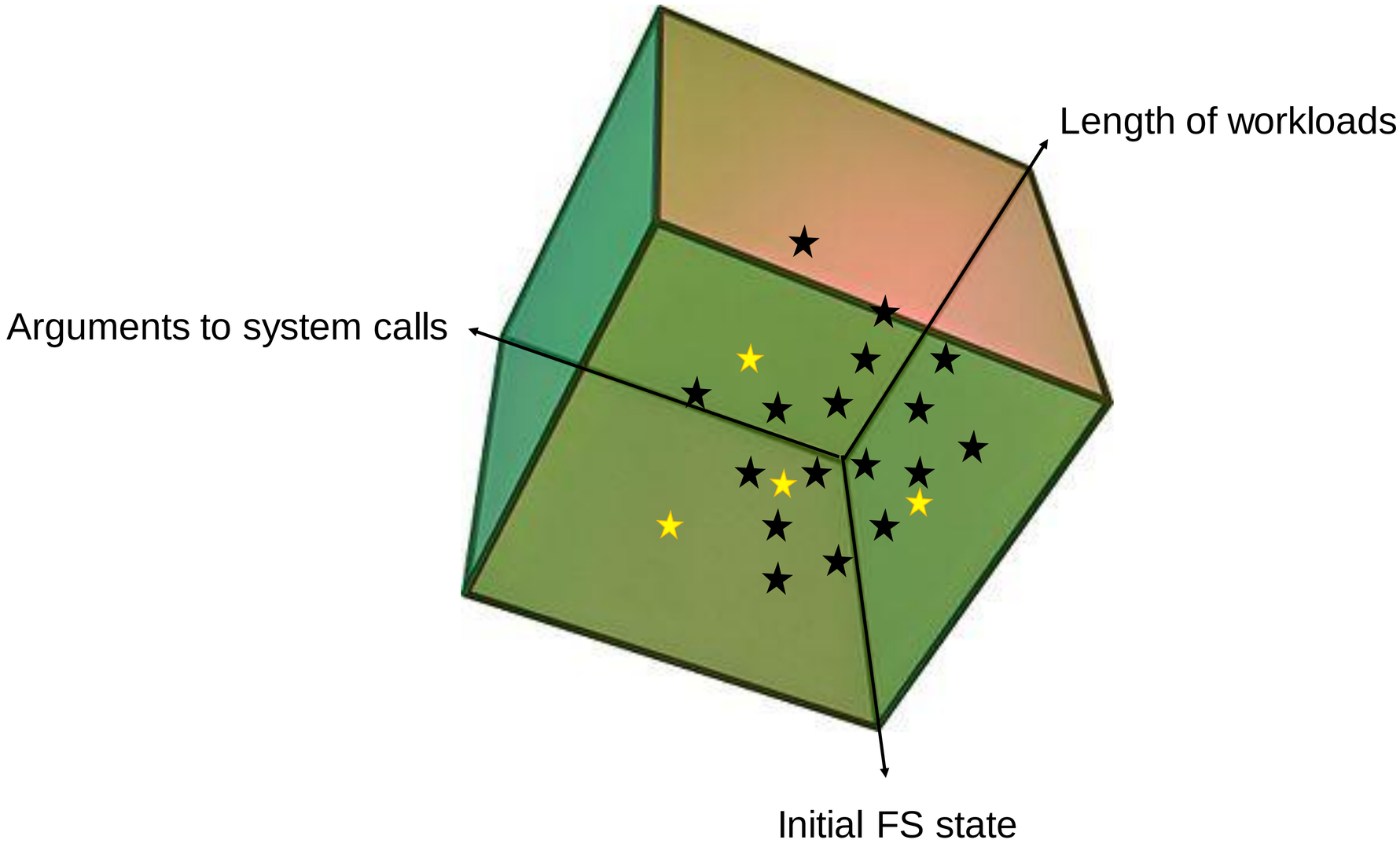



Image source: https://en.wikipedia.org/wiki/Cube

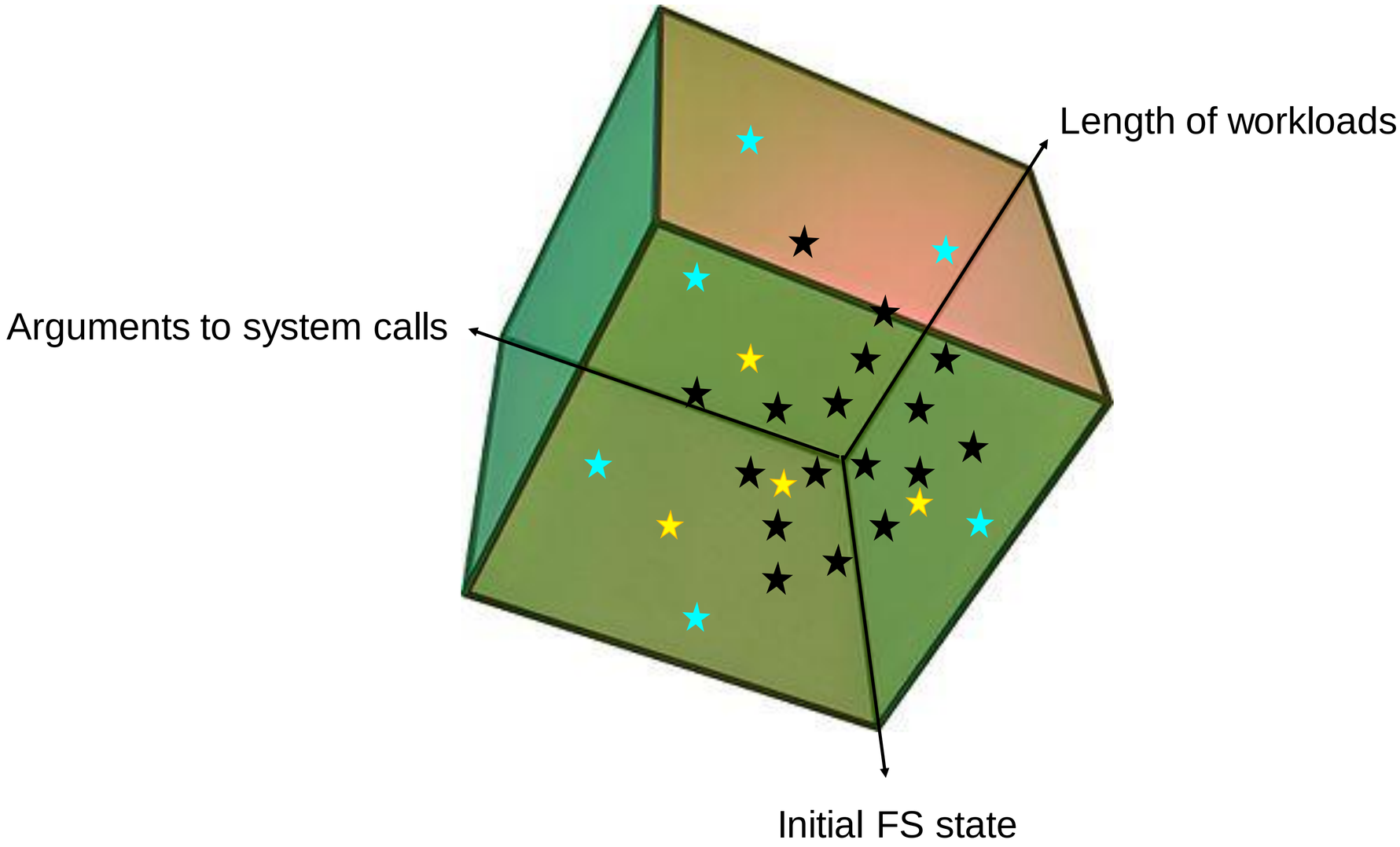
# $B^3$ : Bounded Black Box Crash Testing



# $B^3$ : Bounded Black Box Crash Testing



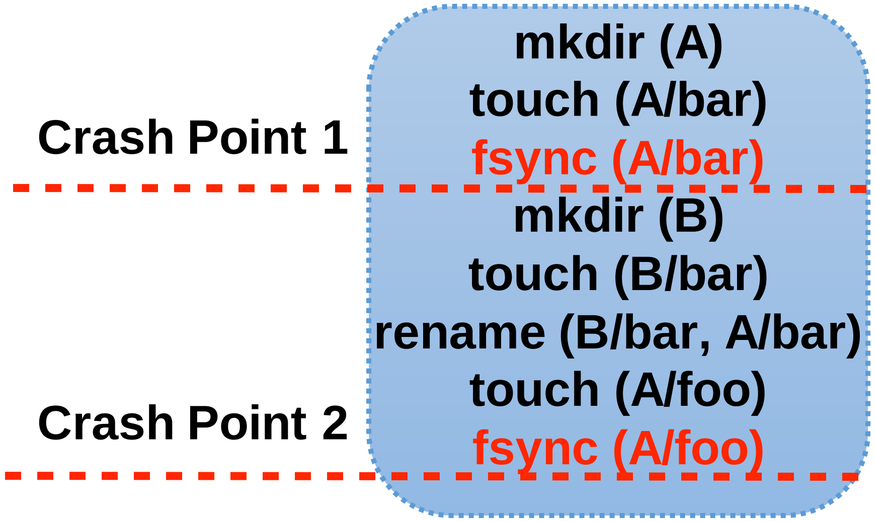
# $B^3$ : Bounded Black Box Crash Testing



# $B^3$ : Bounded Black Box Crash Testing

**Choice of crash point**

- Only after fsync(), fdatasync() or sync()
- Not in the middle of system call




- Developers are motivated to patch bugs that break semantics of persistence operations
- Crashing in the middle of system calls leads to exponentially large crash-states.

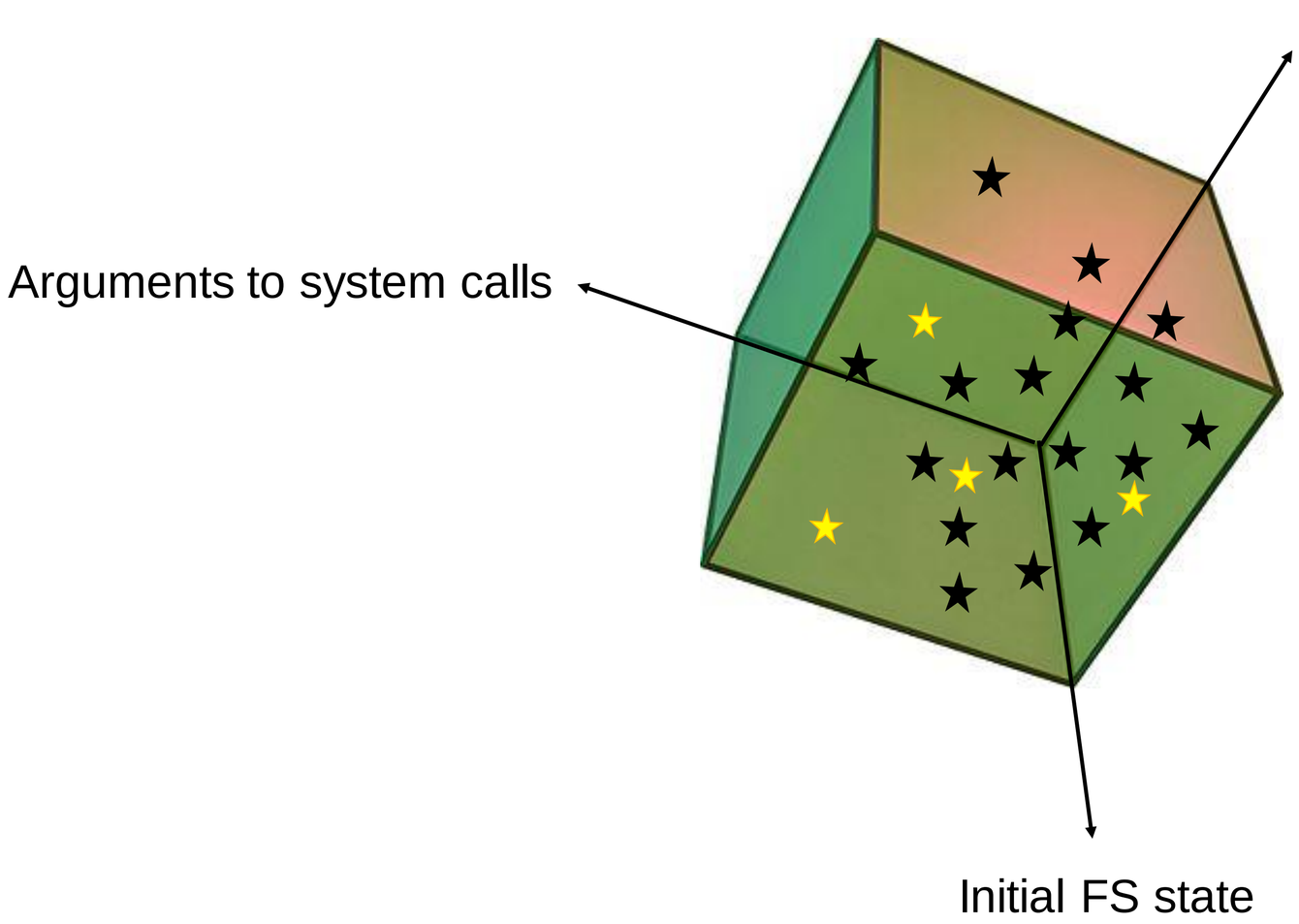
# Limitations of $B^3$

- No guarantee of finding **all** crash-consistency bugs in a filesystem
- Assumes the correct working of crash-consistency mechanism like journaling or CoW
  - Does not crash in the middle of system calls
- Can only reveal if a bug has occurred, not the reason or origin of bug.
- Needs larger compute to test higher sequence lengths

# Outline

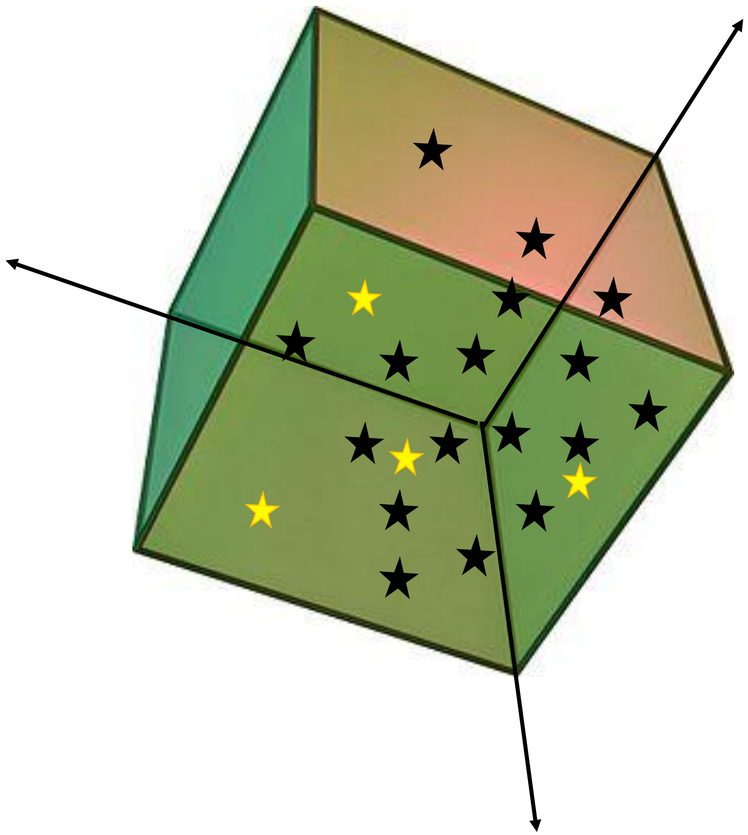
- CrashMonkey
- Bounded Black Box Crash Testing
- **Automatic Crash Explorer (ACE)**
- Results

# Bounds chosen by ACE




**Bounds picked based on insights from the study of crash-consistency bugs reported on Linux file systems over the last 5 years**

# Bounds chosen by ACE



Length of workloads

Maximum # core ops is 3

Arguments to system calls

Initial FS state

# Bounds chosen by ACE

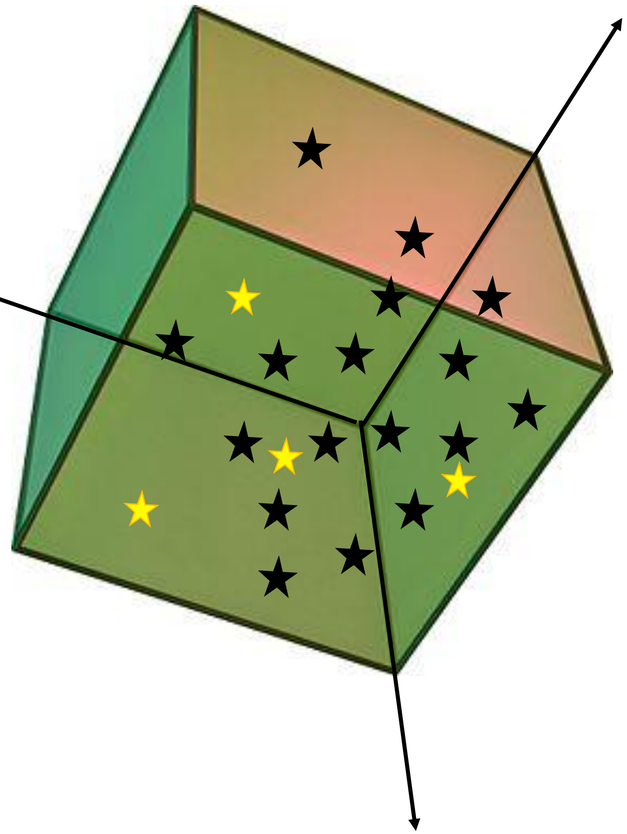
Length of workloads

**Maximum # core ops is 3**

**Arguments to system calls**

**Root**
- **A → (foo, bar)**
- **B → (foo, bar)**

**Overwrites to start, middle, end of a file and append**



Initial FS state

# Bounds chosen by ACE

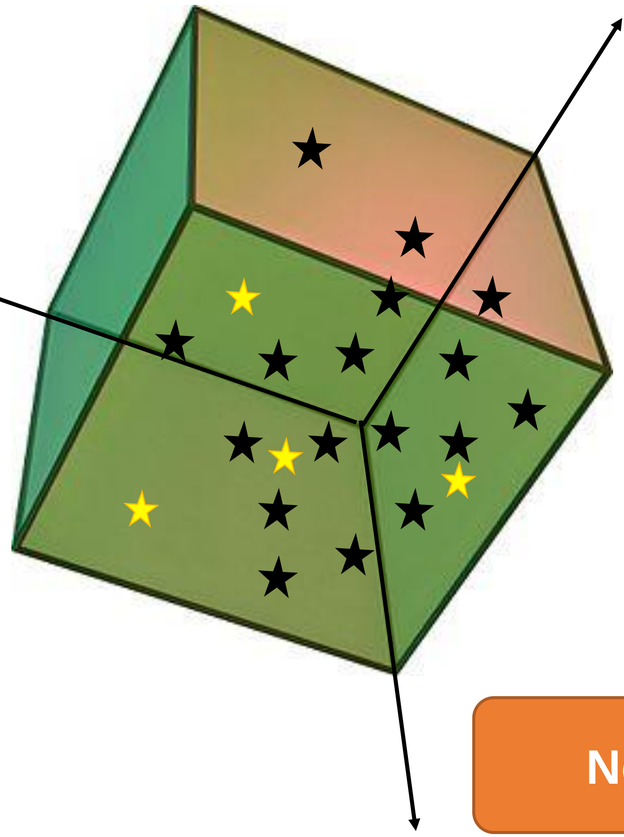
Length of workloads

**Maximum # core ops is 3**

Arguments to system calls

**Root**
- **A → (foo, bar)**
- **B → (foo, bar)**

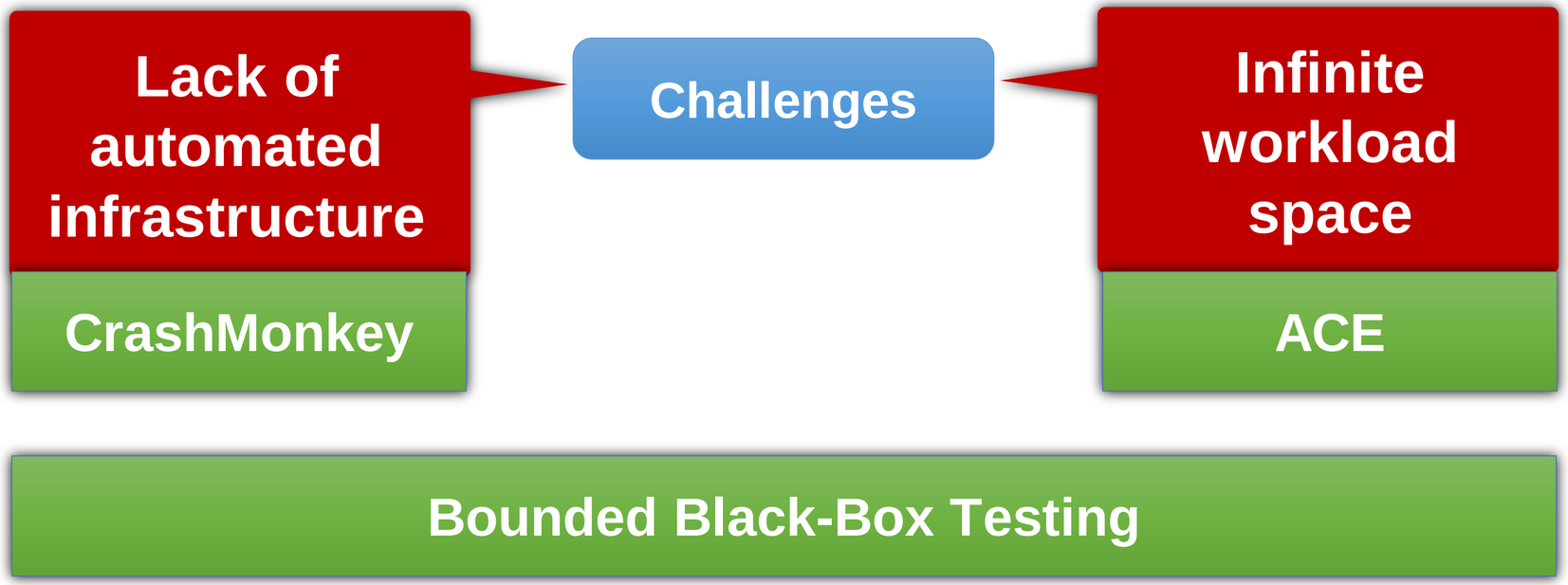
**Overwrites to start, middle, end and append**



**New, 100MB FS**

**Initial FS state**

# Challenges with Systematic Testing



# Outline

- CrashMonkey
- Bounded Black Box Crash Testing
- Automatic Crash Explorer (ACE)
- **Results**

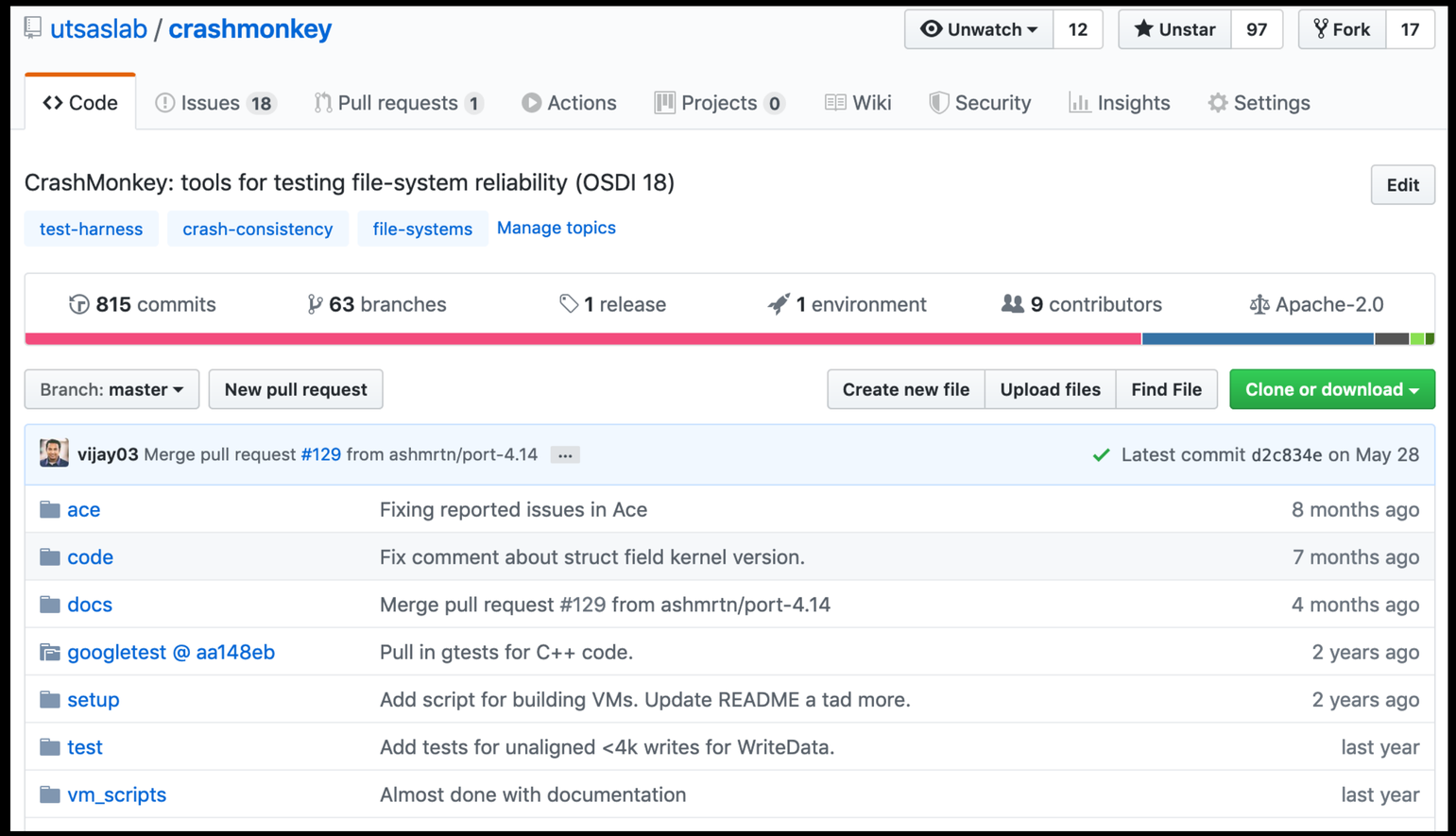
# Results

| Sequence Length | # workloads | # Bugs Reproduced | # Bugs found |
|---|---|---|---|
| 1, 2, 3 | 3.37M | 24 | 10 |

- Reproduced 24/26 known bugs across ext4, btrfs and F2FS
- Found 10 new bugs across btrfs and F2FS
- Found 1 bug in a verified file system, FSCQ

# Bounded Black-Box Crash Testing

- $B^3$ makes exhaustive testing feasible using informed bound selection
- Easily generalizable to test larger workloads if more compute is available
- Found 10 new bugs across btrfs and F2FS, most of which existed since 2014
- Found 1 bug in FSCQ




Thanks!
Questions?

Try our tools : https://github.com/utsaslab/crashmonkey