SKI: Exposing Kernel Concurrency Bugs through Systematic Schedule Exploration

Pedro Fonseca
(MPI-SWS)

Rodrigo Rodrigues
(NOVA University of Lisbon)

Björn Brandenburg
(MPI-SWS)

OSDI 2014
Kernel concurrency bugs

- Bugs that depend on the instruction interleavings
  - **Triggered only by a subset** of the interleavings
Kernel concurrency bugs

- Bugs that depend on the instruction interleavings
  - Triggered only by a subset of the interleavings
- Plenty of kernel concurrency bugs in kernels!
Kernel concurrency bugs

- Bugs that depend on the instruction interleavings
  - Triggered only by a subset of the interleavings
- Plenty of kernel concurrency bugs in kernels!

The bug is a race and not always easy to reproduce. [...] On my particular machine, [the test case] usually triggers [the bug] within 10 minutes but enabling debug options can change the timing such that it never hits. Once the bug is triggered, the machine is in trouble and needs to be rebooted.

Linux 3.0.41 change log
Kernel concurrency bugs

- Bugs that depend on the instruction interleavings
  - Triggered only by a subset of the interleavings
- Plenty of kernel concurrency bugs in kernels!

The bug is a race and not always easy to reproduce. [...] On my particular machine, [the test case] usually triggers [the bug] within 10 minutes but enabling debug options can change the timing such that it never hits. Once the bug is triggered, the machine is in trouble and needs to be rebooted.

[The bug] was quite hard to decode as the reproduction time is between 2 days and 3 weeks and intrusive tracing makes it less likely [...] Linux 3.4.41 change log
Kernel concurrency bugs

• Bugs that depend on the instruction interleavings
  - Triggered only by a subset of the interleavings
• Plenty of kernel concurrency bugs in kernels!

Three of the five 3.4.9 machines [...] locked up.
I've tried reproducing the issue, but so far I've been unsuccessful [...] 
Linux kernel mailing list (5/1/2013)

[The bug] was quite hard to decode as the reproduction time is between 2 days and 3 weeks and intrusive tracing makes it less likely [...] 
Linux 3.4.41 change log
Approaches to explore interleavings

- Stress testing approach
  - Hope to find the interleaving
Approaches to explore interleavings

• Stress testing approach
  - Hope to find the interleaving

• Systematic approach
  - Take full control of the interleavings
  - Existing tools focus on user-mode applications
Approaches to explore interleavings

• Stress testing approach
  - Hope to find the interleaving

• Systematic approach
  - Take full control of the interleavings
  - Existing tools focus on user-mode applications
Approaches to explore interleavings

• Stress testing approach
  – Hope to find the interleaving

• Systematic approach
  – Take full control of the interleavings
  – Existing tools focus on user mode applications

Focus on operating system kernels

This talk
SKI
Finding kernel concurrency bugs

- Testing applications versus kernels
- Our approach
- Implementation
- Evaluation
Existing user-mode tools

App

Kernel

Kernel-level abstractions
Threads and sync. objects

Existing user-mode systematic tools
LD_PRELOAD, ptrace
Existing user-mode tools

- App
- User-mode testing tool
- Kernel
  - Scheduler

Existing user-mode systematic tools
- LD_PRELOAD, ptrace
Kernel-mode challenges

• Kernel doesn't have a good instrumentation interface
Kernel-mode challenges

- Kernel doesn't have a good instrumentation interface
- An alternative would be to modify the kernel
  - But kernel modifications:
Kernel-mode challenges

- Kernel doesn't have a good instrumentation interface

- An alternative would be to modify the kernel
  - But kernel modifications:
    - Change the tested software
    - Are non-trivial
    - Hinder portability
Kernel-mode challenges

• Kernel doesn't have a good instrumentation interface

• An alternative would be to modify the kernel
  - But kernel modifications:
    • Change the tested software
    • Are non-trivial
    • Hinder portability

Avoid kernel modifications
User-mode versus kernel-mode

App

Kernel-level abstractions
Threads and sync. objects

Kernel

Scheduler

HW-level abstractions
mov, add, jmp, registers, APIC

Hardware

Existing user-mode systematic tools
LD_PRELOAD, ptrace

Our tool
(modified VMM)
User-mode versus kernel-mode

- App
- Kernel-level abstractions: Threads and sync. objects
- Kernel
- Kernel testing tool
- Hardware

Existing user-mode systematic tools: LD_PRELOAD, ptrace

Our tool (modified VMM)
SKI
Finding kernel concurrency bugs
SKI
Finding kernel concurrency bugs

Systematic
Full control of the kernel interleavings
SKI
Finding kernel concurrency bugs

Systematic
Full control of the kernel interleavings

Practical
No modifications to the kernel
Fast
SKI
Finding kernel concurrency bugs

- Challenges testing the kernel code
- SKI's approach
- Implementation
- Evaluation
SKI's approach

Challenges
1. How to control the schedules?
2. Which contexts are schedulable?
3. Which schedules to choose?
1. How to control the kernel schedules?
1. How to control the kernel schedules?

- Pin each tested thread to a different CPU (thread affinity)
1. How to control the kernel schedules?

- Pin each tested thread to a different CPU (thread affinity)
- Pause and resume CPUs to control schedules
1. How to control the kernel schedules?

- Pin each tested thread to a different CPU (thread affinity)
- Pause and resume CPUs to control schedules
1. How to control the kernel schedules?

- Pin each tested thread to a different CPU (thread affinity)
- Pause and resume CPUs to control schedules

Leverage thread affinity and control CPUs
2. Which contexts are schedulable?

- Execution of some instructions are good hints
2. Which contexts are schedulable?

- Execution of some instructions are good hints

![Diagram showing schedulable contexts]
2. Which contexts are schedulable?

- Execution of some instructions are good hints
- Memory access patterns can also provide hints

![Diagram showing schedulable contexts]
2. Which contexts are schedulable?

- Execution of some instructions are good hints
- Memory access patterns can also provide hints

Rely on VMM introspection
3. Which schedules to choose?

- PCT: User-mode scheduling algorithm [ASPLOS'10]
  - Run the highest priority live threads
  - Create schedule diversity
3. Which schedules to choose?

- PCT: User-mode scheduling algorithm [ASPLOS'10]
  - Run the highest priority live threads
  - Create schedule diversity
- Generalize with interrupt support
  - Detect arrival / end
  - Control dispatch
3. Which schedules to choose?

- **PCT**: User-mode scheduling algorithm [ASPLOS'10]
  - Run the highest priority live threads
  - Create schedule diversity
- **Generalize with interrupt support**
  - Detect arrival / end
  - Control dispatch
- **Reduce interleaving space**
3. Which schedules to choose?

- PCT: User-mode scheduling algorithm [ASPLOS'10]
  - Run the highest priority live threads
  - Create schedule diversity
- Generalize with interrupt support
  - Detect arrival / end
  - Control dispatch
- Reduce interleaving space

**Generalize user-mode systematic testing algorithms**
SKI
Finding kernel concurrency bugs

- Challenges testing kernel code
- SKI's approach
- **Implementation**
- Evaluation
Implementation

- Implemented SKI by modifying QEMU (VMM)
  - No kernel changes required
Implementation

• Implemented SKI by modifying QEMU (VMM)
  - No kernel changes required
• Built a user-mode library (VM)
  - Flags start/end of tests and sends results to VMM
  - Used library to implement several test-cases
    • e.g., file system tests
Implementation

• Implemented SKI by modifying QEMU (VMM)
  - No kernel changes required
• Built a user-mode library (VM)
  - Flags start/end of tests and sends results to VMM
  - Used library to implement several test-cases
    • e.g., file system tests
• Implemented several optimizations
Detecting and diagnosing bugs with SKI

- SKI supports different types of bug detectors
  - Crash and assertion violations
  - Data races
  - Semantic bugs (e.g. disk corruption)
Detecting and diagnosing bugs with SKI

• SKI supports different types of bug detectors
  - Crash and assertion violations
  - Data races
  - Semantic bugs (e.g. disk corruption)

• SKI produces detailed execution traces
SKI
Finding kernel concurrency bugs

• Challenges testing kernel code
• SKI's approach
• Implementation
• **Evaluation**

1. Regression testing
2. Finding previously unknown bugs
1. Regression testing: setup

• Searched for previously reported bugs
  – In kernel bugzilla, mailing lists, git logs
  – Well documented reports and diverse set of bugs

• Created SKI test suites for these bugs
  – By adapting the stress tests in the bug reports
1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
</tr>
</tbody>
</table>
1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
</tr>
</tbody>
</table>

Diverse properties
## 1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
</tr>
</tbody>
</table>
1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
</tr>
</tbody>
</table>

SKI is portable
1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
<th>Schedules</th>
<th>Throughput (sched/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
<td>28</td>
<td>302,000</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
<td>53</td>
<td>169,300</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
<td>51</td>
<td>218,700</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
<td>3519</td>
<td>501,400</td>
</tr>
</tbody>
</table>
1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
<th>Schedules</th>
<th>Throughput (sched/h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
<td>28</td>
<td>302,000</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
<td>53</td>
<td>169,300</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
<td>51</td>
<td>218,700</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
<td>3519</td>
<td>501,400</td>
</tr>
</tbody>
</table>

SKI can expose bugs in seconds
### 1. Regression testing: results

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
<th>Schedules</th>
<th>Throughput (sched/h)</th>
<th>Schedules</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
<td>28</td>
<td>302,000</td>
<td>NA (&gt;24h)</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
<td>53</td>
<td>169,300</td>
<td>200,000 (4h)</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
<td>51</td>
<td>218,700</td>
<td>800 (1 min)</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
<td>3519</td>
<td>501,400</td>
<td>NA (&gt;24h)</td>
</tr>
</tbody>
</table>
1. Regression testing: results

Some stress tests were ineffective

<table>
<thead>
<tr>
<th>Bug</th>
<th>Kernel</th>
<th>Component</th>
<th>Detector</th>
<th>Schedules</th>
<th>Throughput (sched/h)</th>
<th>Schedules</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Linux 2.6.28</td>
<td>Anonymous pipes</td>
<td>Crash</td>
<td>28</td>
<td>302,000</td>
<td>NA (&gt;24h)</td>
</tr>
<tr>
<td>B</td>
<td>Linux 3.2</td>
<td>Inotify + FAT32</td>
<td>Crash</td>
<td>53</td>
<td>169,300</td>
<td>200,000 (4h)</td>
</tr>
<tr>
<td>C</td>
<td>Linux 3.6.1</td>
<td>Proc + Ext4</td>
<td>Semantic</td>
<td>51</td>
<td>218,700</td>
<td>800 (1 min)</td>
</tr>
<tr>
<td>D</td>
<td>FreeBSD 8.0</td>
<td>Sockets</td>
<td>Semantic</td>
<td>3519</td>
<td>501,400</td>
<td>NA (&gt;24h)</td>
</tr>
</tbody>
</table>
2. Finding previously unknown bugs

- Created a SKI test suit for file systems
  - Adapted the existing fsstress test suit
  - Tested several file systems
- Bug detectors
  - Crashes, warnings, data races, semantic errors (fsck)
- Tested recent versions of Linux
## 2. Finding previously unknown bugs

<table>
<thead>
<tr>
<th>Bug</th>
<th>Linux</th>
<th>FS</th>
<th>Detector / Failure</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>2</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer) + Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>3</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>4</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Fsck (References not found)</td>
<td>Reported</td>
</tr>
<tr>
<td>5</td>
<td>3.11.1+p</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>6</td>
<td>3.12.2</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>7</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Null-pointer)</td>
<td>Reported</td>
</tr>
<tr>
<td>8</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Invalid paging)</td>
<td>Reported</td>
</tr>
<tr>
<td>9</td>
<td>3.13.5</td>
<td>Jfs</td>
<td>Crash (Assertion violation)</td>
<td>Reported</td>
</tr>
<tr>
<td>10</td>
<td>3.13.5</td>
<td>Ext4</td>
<td>Data race</td>
<td>Fixed</td>
</tr>
<tr>
<td>11</td>
<td>3.13.5</td>
<td>VFS</td>
<td>Data race</td>
<td>Reported</td>
</tr>
</tbody>
</table>
2. Finding previously unknown bugs

<table>
<thead>
<tr>
<th>Bug</th>
<th>Linux</th>
<th>FS</th>
<th>Detector / Failure</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>2</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer + Warning)</td>
<td>Fixed</td>
</tr>
<tr>
<td>3</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>4</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Fsck (References not found)</td>
<td>Reported</td>
</tr>
<tr>
<td>5</td>
<td>3.11.1+p</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>6</td>
<td>3.12.2</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>7</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Null-pointer)</td>
<td>Reported</td>
</tr>
<tr>
<td>8</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Invalid paging)</td>
<td>Reported</td>
</tr>
<tr>
<td>9</td>
<td>3.13.5</td>
<td>Jfs</td>
<td>Crash (Assertion violation)</td>
<td>Reported</td>
</tr>
<tr>
<td>10</td>
<td>3.13.5</td>
<td>Ext4</td>
<td>Data race</td>
<td>Fixed</td>
</tr>
<tr>
<td>11</td>
<td>3.13.5</td>
<td>VFS</td>
<td>Data race</td>
<td>Reported</td>
</tr>
</tbody>
</table>

Official Linux releases

SKI: Exposing Kernel Concurrency Bugs  
Pedro Fonseca
## 2. Finding previously unknown bugs

<table>
<thead>
<tr>
<th>Bug</th>
<th>Linux</th>
<th>FS</th>
<th>Detector / Failure</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>2</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer) + Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>3</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>4</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Fsck (References not found)</td>
<td>Reported</td>
</tr>
<tr>
<td>5</td>
<td>3.11.1+p</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>6</td>
<td>3.12.2</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>7</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Null-pointer)</td>
<td>Reported</td>
</tr>
<tr>
<td>8</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Invalid paging)</td>
<td>Reported</td>
</tr>
<tr>
<td>9</td>
<td>3.13.5</td>
<td>Jfs</td>
<td>Crash (Assertion violation)</td>
<td>Reported</td>
</tr>
<tr>
<td>10</td>
<td>3.13.5</td>
<td>Ext4</td>
<td>Data race</td>
<td>Fixed</td>
</tr>
<tr>
<td>11</td>
<td>3.13.5</td>
<td>VFS</td>
<td>Data race</td>
<td>Reported</td>
</tr>
</tbody>
</table>

**Requested by developers**
## 2. Finding previously unknown bugs

<table>
<thead>
<tr>
<th>Bug</th>
<th>Linux</th>
<th>FS</th>
<th>Detector / Failure</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>2</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer) + Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>3</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>4</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Fsck (References not found)</td>
<td>Reported</td>
</tr>
<tr>
<td>5</td>
<td>3.11.1+p</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>6</td>
<td>3.12.2</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>7</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Null-pointer)</td>
<td>Reported</td>
</tr>
<tr>
<td>8</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Invalid paging)</td>
<td>Reported</td>
</tr>
<tr>
<td>9</td>
<td>3.13.5</td>
<td>Jfs</td>
<td>Crash (Assertion violation)</td>
<td>Reported</td>
</tr>
<tr>
<td>10</td>
<td>3.13.5</td>
<td>Ext4</td>
<td>Data race</td>
<td>Fixed</td>
</tr>
<tr>
<td>11</td>
<td>3.13.5</td>
<td>VFS</td>
<td>Data race</td>
<td>Reported</td>
</tr>
</tbody>
</table>

**Important file systems:**

- Btrfs
- Logfs
- Ext4
- VFS
## 2. Finding previously unknown bugs

<table>
<thead>
<tr>
<th>Bug</th>
<th>Linux</th>
<th>FS</th>
<th>Detector / Failure</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>2</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Crash (Null-pointer) + Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>3</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>4</td>
<td>3.11.1</td>
<td>Btrfs</td>
<td>Fsck (References not found)</td>
<td>Reported</td>
</tr>
<tr>
<td>5</td>
<td>3.11.1+p</td>
<td>Btrfs</td>
<td>Crash (Null-pointer)</td>
<td>Fixed</td>
</tr>
<tr>
<td>6</td>
<td>3.12.2</td>
<td>Btrfs</td>
<td>Warning</td>
<td>Fixed</td>
</tr>
<tr>
<td>7</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Null-pointer)</td>
<td>Reported</td>
</tr>
<tr>
<td>8</td>
<td>3.13.5</td>
<td>Logfs</td>
<td>Crash (Invalid paging)</td>
<td>Reported</td>
</tr>
<tr>
<td>9</td>
<td>3.13.6</td>
<td>Jfs</td>
<td>Crash (Assertion violation)</td>
<td>Reported</td>
</tr>
<tr>
<td>10</td>
<td>3.13.5</td>
<td>Ext4</td>
<td>Data race</td>
<td>Fixed</td>
</tr>
<tr>
<td>11</td>
<td>3.13.5</td>
<td>VFS</td>
<td>Data race</td>
<td>Reported</td>
</tr>
</tbody>
</table>

**Data loss**
Current limitations and future work
Current limitations and future work

- Bugs in kernel scheduler code
  - SKI pins tested threads
    → Represent a small set of bugs
Current limitations and future work

- Bugs in kernel scheduler code
  - SKI pins tested threads
    → Represent a small set of bugs

- Bugs in device drivers
  - SKI supports a large set of devices but not all
    → Implement SKI with binary instrumentation techniques
Current limitations and future work

- Bugs in kernel scheduler code
  - SKI pins tested threads
    → Represent a small set of bugs

- Bugs in device drivers
  - SKI supports a large set of devices but not all
    → Implement SKI with binary instrumentation techniques

- Bugs that depend on weak memory models
  - SKI currently implements a strong memory model
    → Generalize SKI to also expose these bugs
Conclusion
Conclusion

SKI is Systematic

Full control of the kernel interleavings
Conclusion

**SKI is Systematic**
Full control of the kernel interleavings

**SKI is Practical**
No modifications to the kernel
Fast
Conclusion

**SKI is Systematic**
Full control of the kernel interleavings

**SKI is Practical**
No modifications to the kernel
Fast

**SKI is Effective**
Finds and reproduces real-world kernel concurrency bugs