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

UNIVERSIDADE NOVA DE LISBOA

Max
Planck
Institute
for
Software Systems

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!

The bug is a race ... particular machi... within **10 minut**... **timing such that** ... machine is in trouble and needs to be rebooted.

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

This talk

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

android

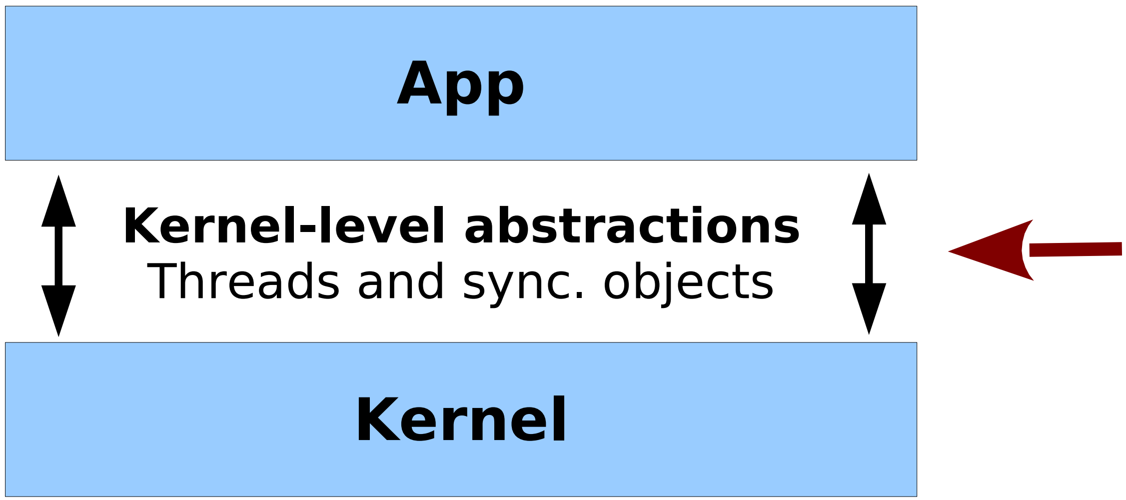
This talk

SKI

Finding kernel concurrency bugs

- **Testing applications versus kernels**
- Our approach
- Implementation
- Evaluation

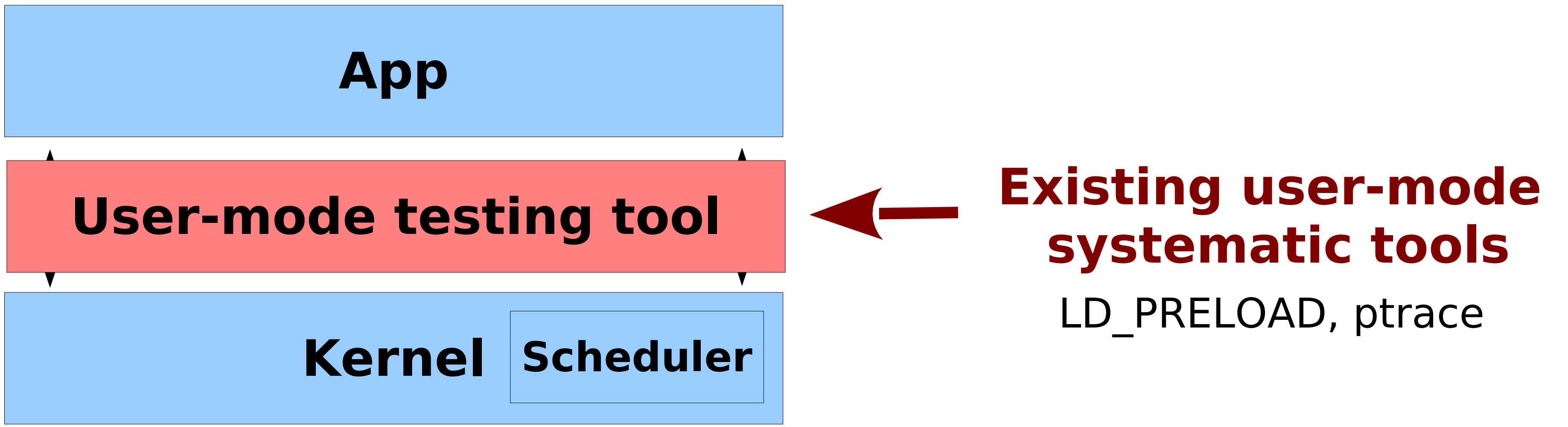
Existing user-mode tools



Existing user-mode systematic tools

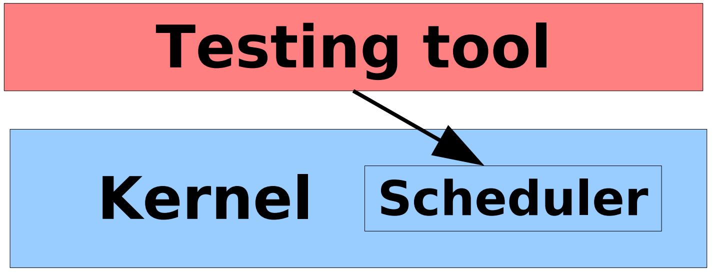
LD_PRELOAD, ptrace

Existing user-mode tools



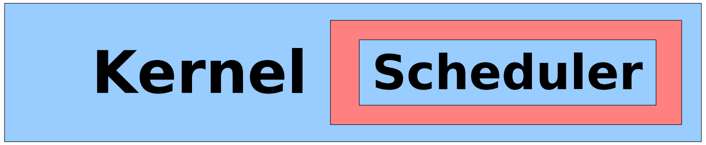
Kernel-mode challenges

- Kernel doesn't have a good instrumentation interface



Kernel-mode challenges

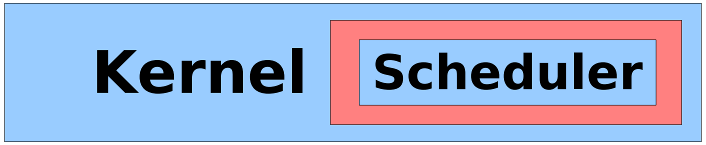
- Kernel doesn't have a good instrumentation interface



- An alternative would be to modify the kernel
  - But kernel modifications:

Kernel-mode challenges

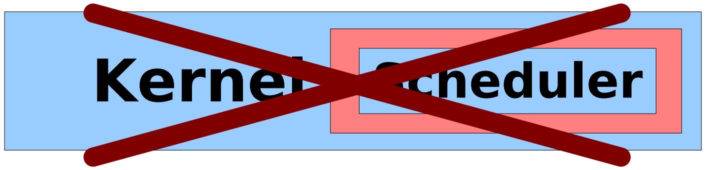
- Kernel doesn't have a good instrumentation interface



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

Kernel-mode challenges

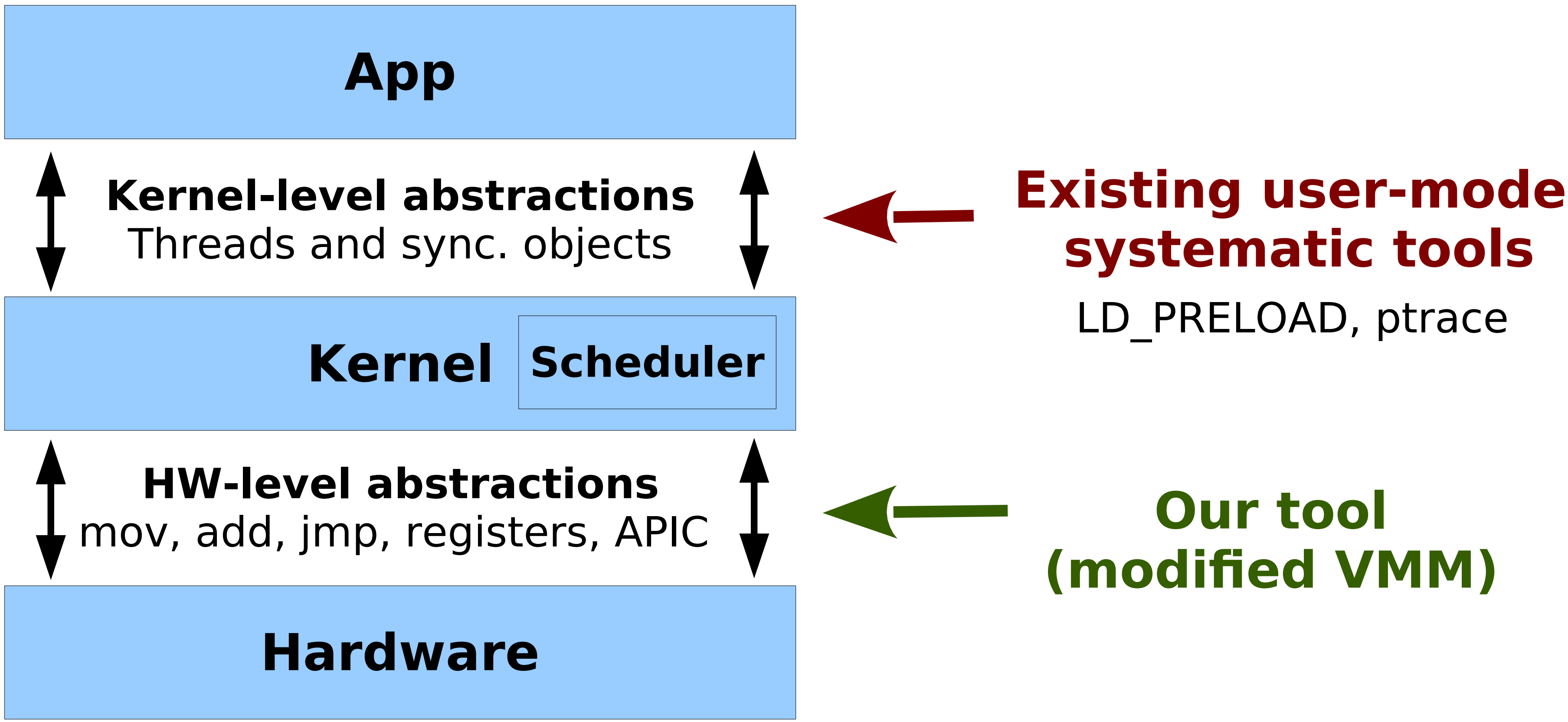
- Kernel doesn't have a good instrumentation interface



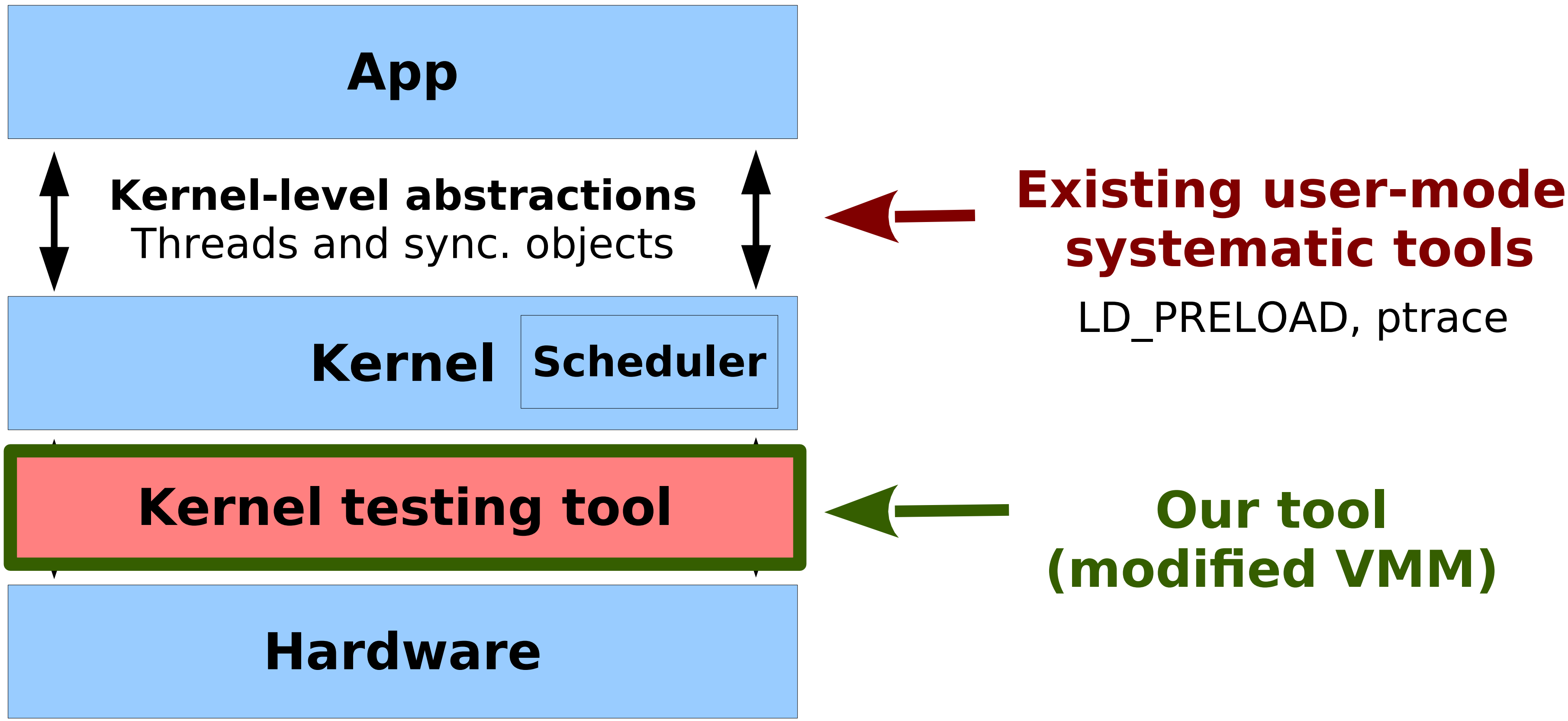
- An alternative would be to modify the kernel
  - But kernel modifications:
    - Change the tested software
    - Are non-trivial
    - Hinder portability

Avoid kernel modifications

User-mode *versus* kernel-mode



User-mode *versus* kernel-mode

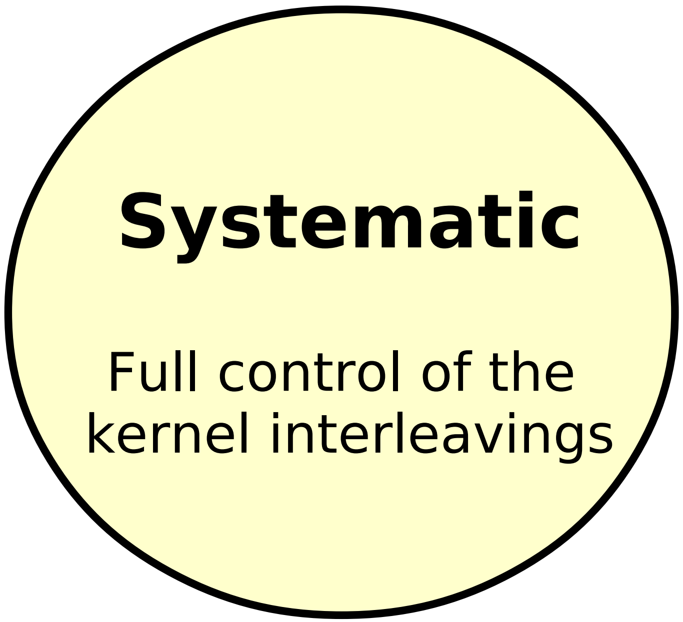


SKI

Finding kernel concurrency bugs

SKI

Finding kernel concurrency bugs



SKI

Finding kernel concurrency bugs

Systematic

Full control of the kernel interleavings

+

Practical

No modifications to the kernel

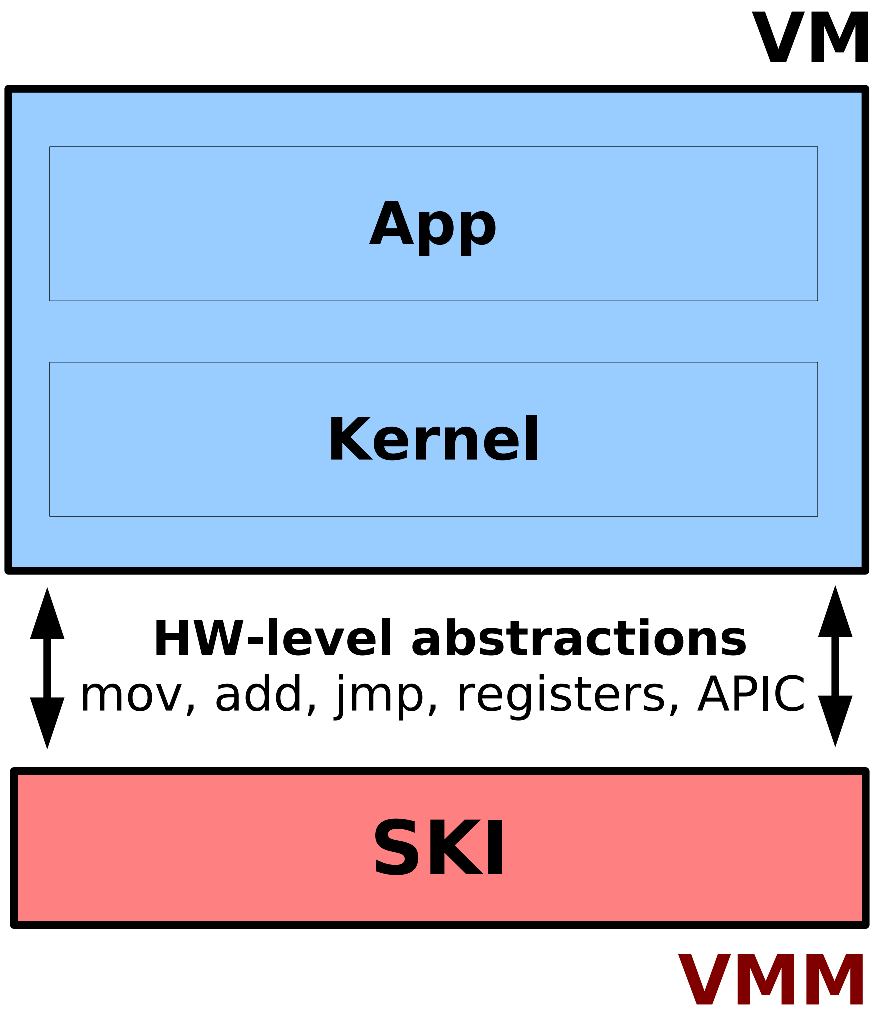
Fast

SKI

Finding kernel concurrency bugs

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

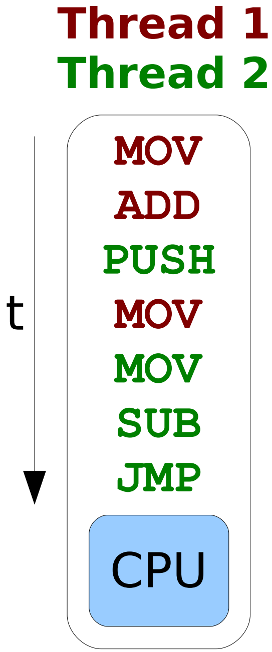
SKI's approach



Challenges

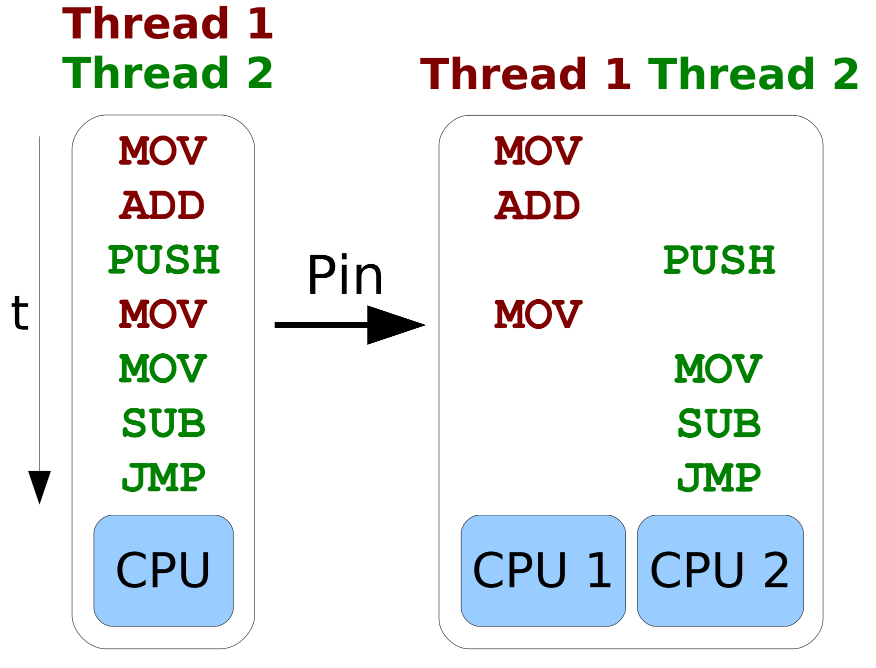
1. How to control the schedules?
2. Which contexts are schedulable?
3. Which schedules to choose?

1. How to control the kernel schedules?



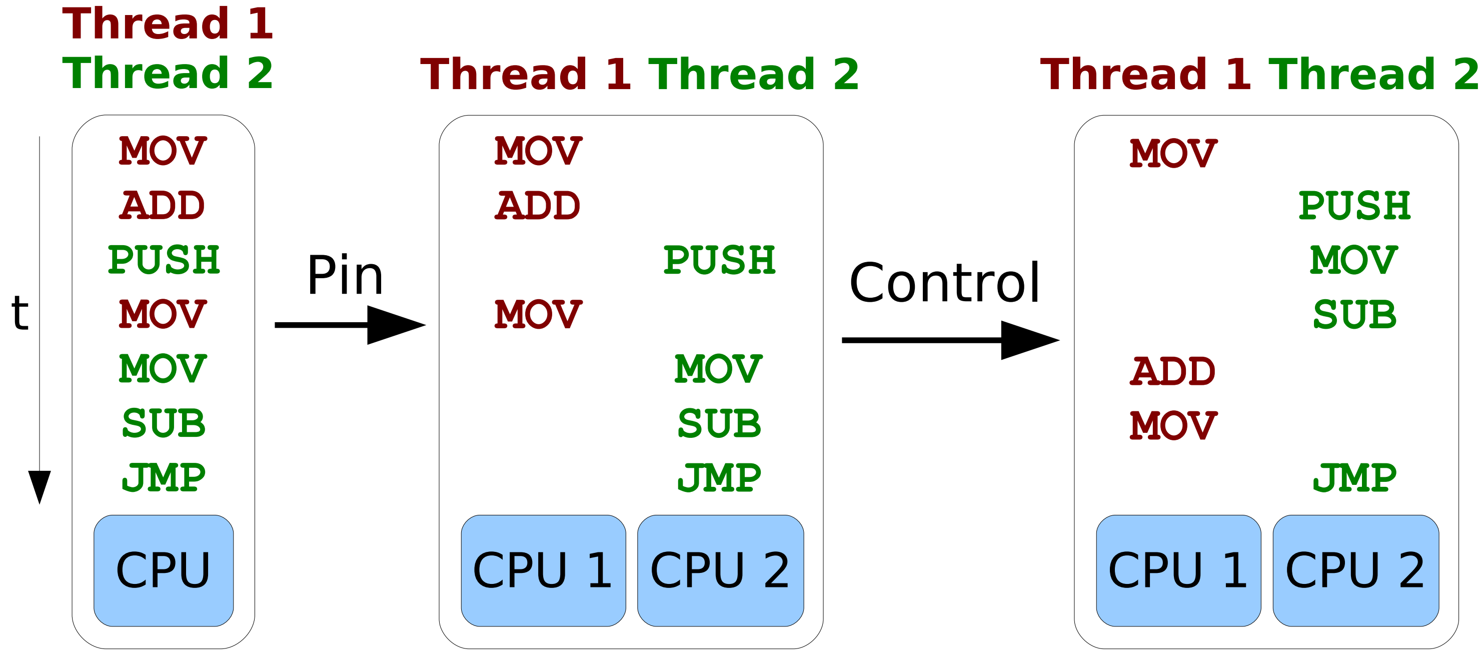
1. How to control the kernel schedules?

- Pin each tested thread to a different CPU (thread affinity)



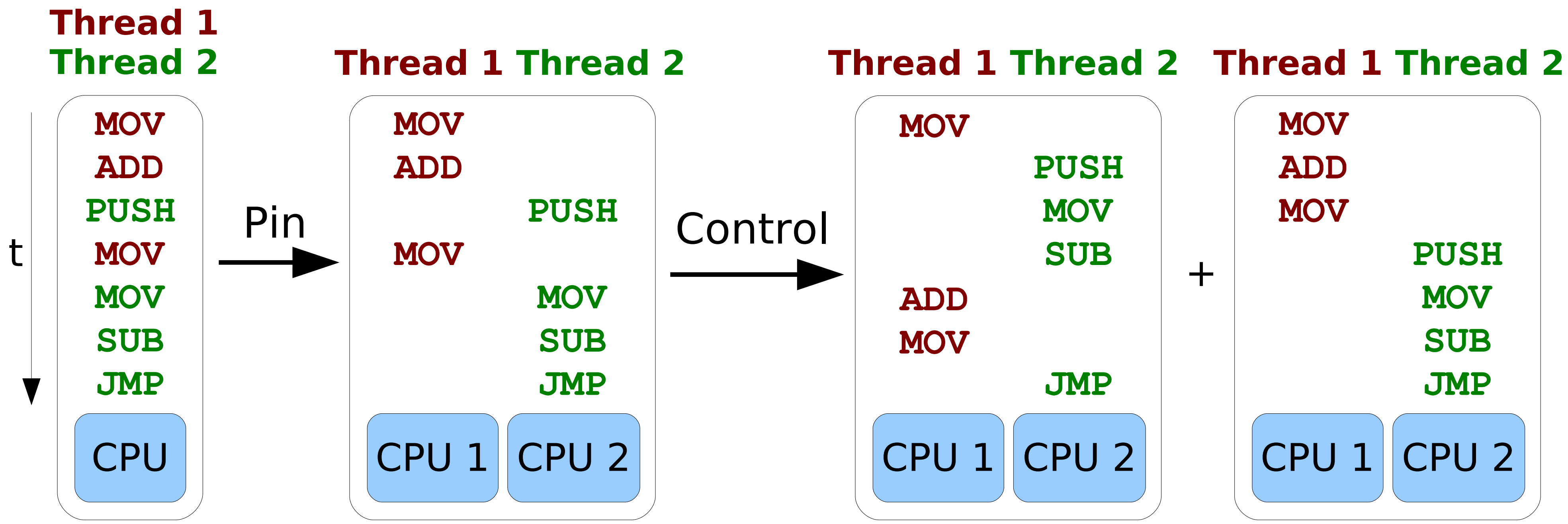
1. How to control the kernel schedules?

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



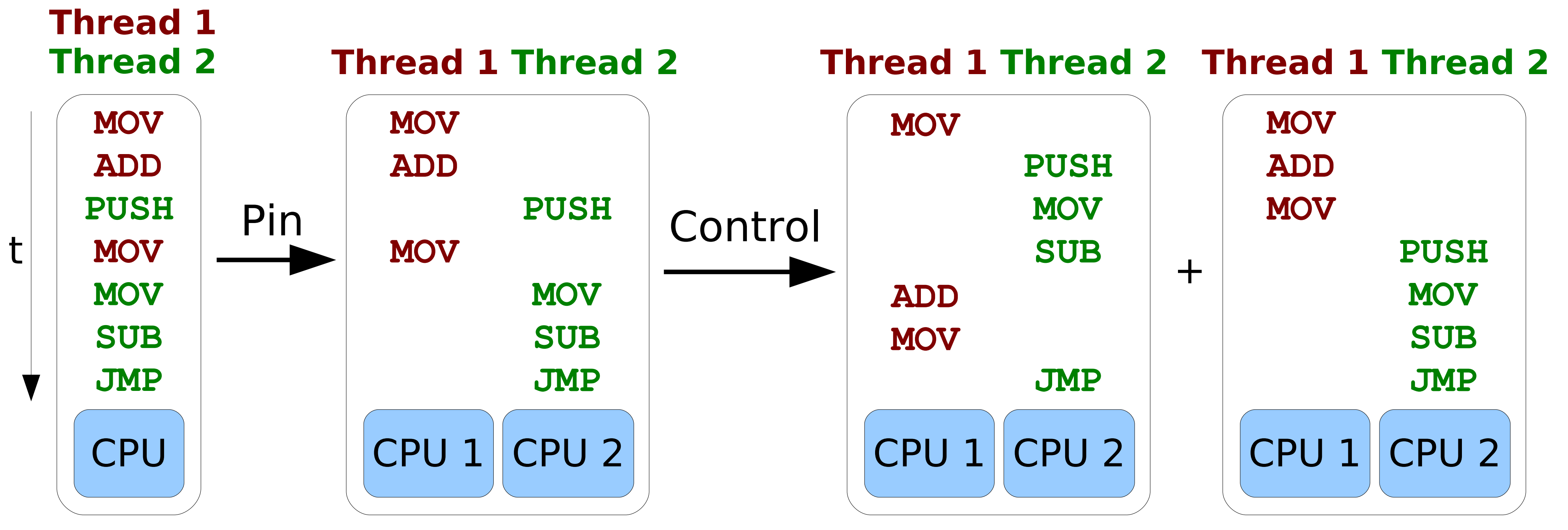
1. How to control the kernel schedules?

- 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



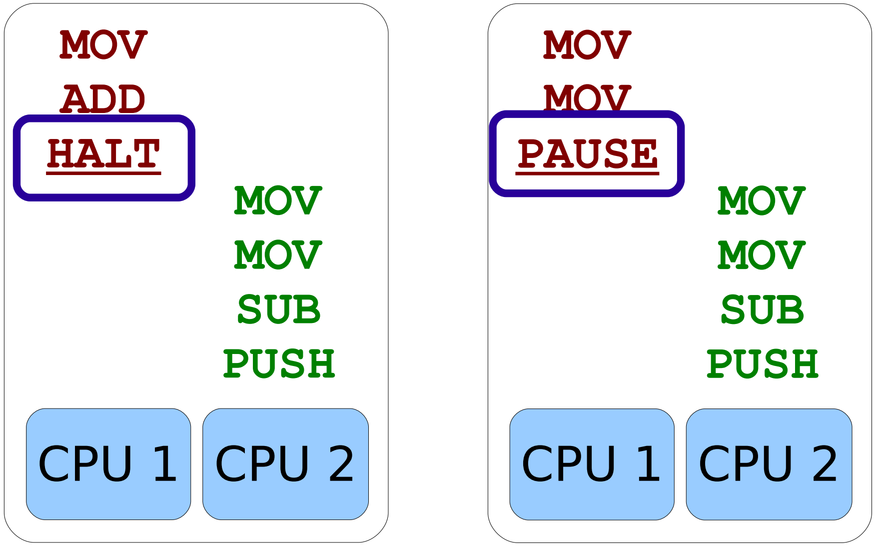
Leverage thread affinity and control CPUs

2. Which contexts are schedulable?

- Execution of some instructions are good hints

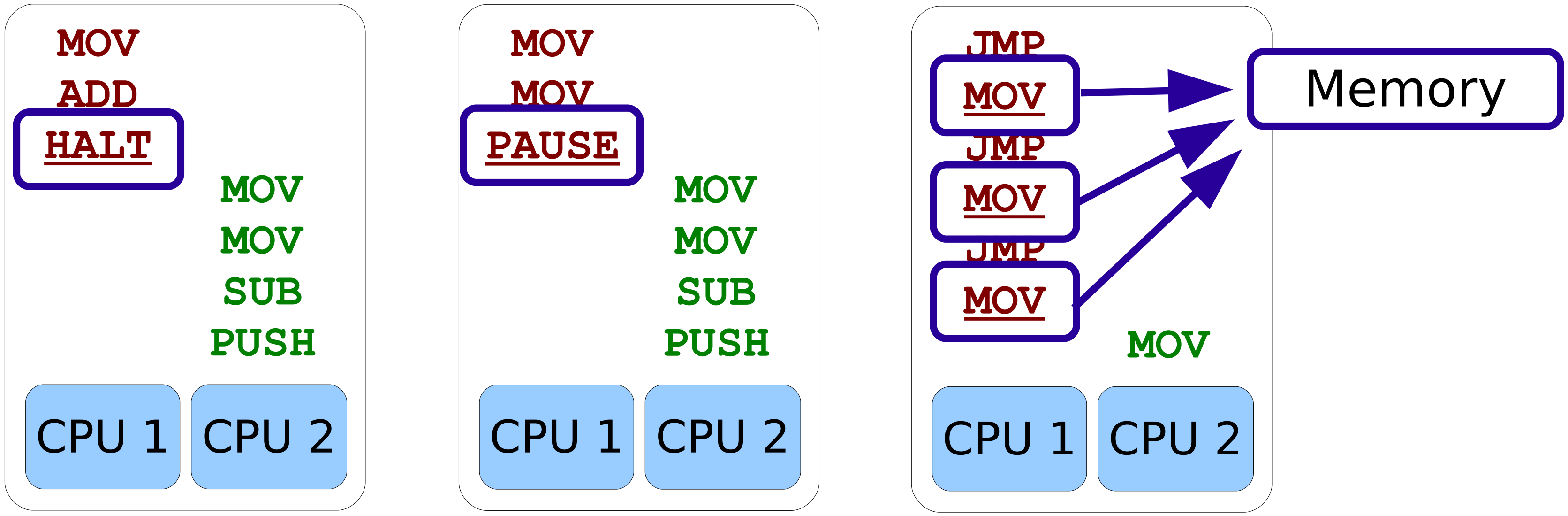
2. Which contexts are schedulable?

- Execution of some instructions are good hints



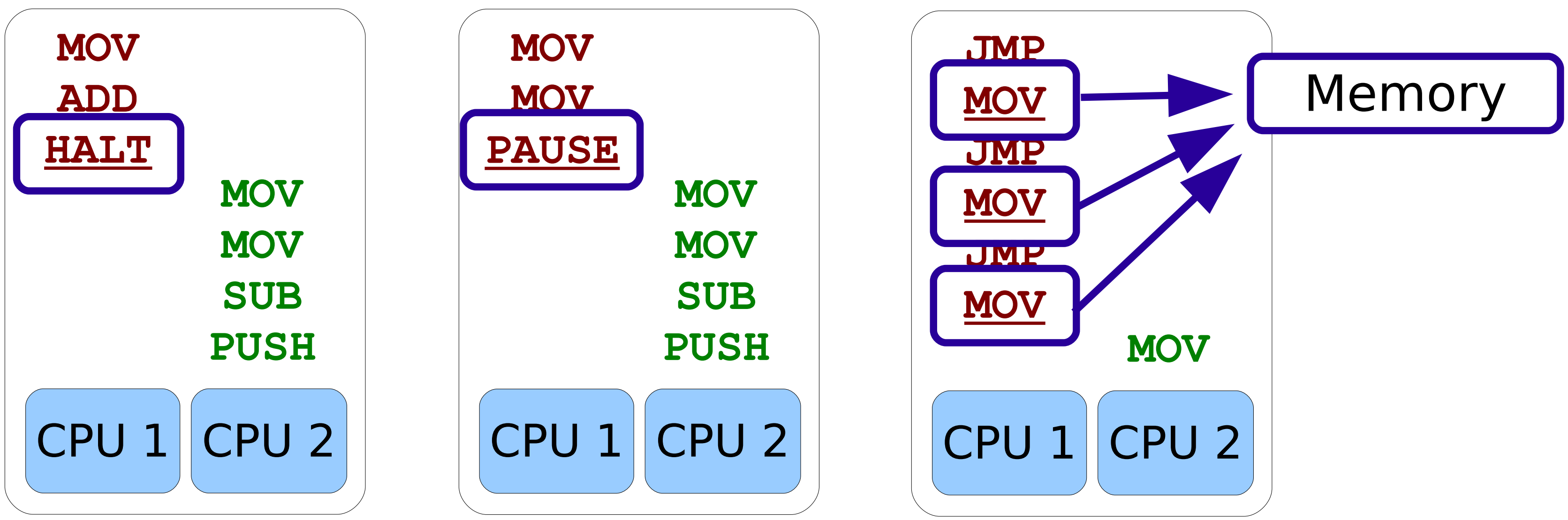
2. Which contexts are schedulable?

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



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

Bug	Kernel	Component	Detector
A	Linux 2.6.28	Anonymous pipes	Crash
B	Linux 3.2	Inotify + FAT32	Crash
C	Linux 3.6.1	Proc + Ext4	Semantic
D	FreeBSD 8.0	Sockets	Semantic

1. Regression testing: results

Bug	Kernel	Component	Detector
A	Linux 2.6.28	Anonymous pipes	Crash
B	Linux 3.2	Inotify + FAT32	Crash
C	Linux 3.6.1	Proc + Ext4	Semantic
D	FreeBSD 8.0	Sockets	Semantic

Diverse properties

1. Regression testing: results

Bug	Kernel	Component	Detector
A	Linux 2.6.28	Anonymous pipes	Crash
B	Linux 3.2	Inotify + FAT32	Crash
C	Linux 3.6.1	Proc + Ext4	Semantic
D	FreeBSD 8.0	Sockets	Semantic

1. Regression testing: results

Bug	Kernel	Component	Detector
A	Linux 2.6.28	Anonymous pipes	Crash
B	Linux 3.2	Inotify + FAT32	Crash
C	Linux 3.6.1	Proc + Ext4	Semantic
D	FreeBSD 8.0	Sockets	Semantic

SKI is portable

1. Regression testing: results

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

1. Regression testing: results

SKI can expose bugs in seconds

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

1. Regression testing: results

				SKI		Stress tests
Bug	**Kernel**	**Component**	**Detector**	**Schedules**	**Throughput (sched/h)**	**Schedules**
A	Linux 2.6.28	Anonymous pipes	Crash	28	302,000	NA (>24h)
B	Linux 3.2	Inotify + FAT32	Crash	53	169,300	200,000 (4h)
C	Linux 3.6.1	Proc + Ext4	Semantic	51	218,700	800 (1 min)
D	FreeBSD 8.0	Sockets	Semantic	3519	501,400	NA (>24h)

1. Regression testing: results

Some stress tests were ineffective

				SKI		Stress tests
Bug	**Kernel**	**Component**	**Detector**	**Schedules**	**Throughput (sched/h)**	**Schedules**
A	Linux 2.6.28	Anonymous pipes	Crash	28	302,000	NA (>24h)
B	Linux 3.2	Inotify + FAT32	Crash	53	169,300	200,000 (4h)
C	Linux 3.6.1	Proc + Ext4	Semantic	51	218,700	800 (1 min)
D	FreeBSD 8.0	Sockets	Semantic	3519	501,400	NA (>24h)

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

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

2. Finding previously unknown bugs

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

Official Linux releases

2. Finding previously unknown bugs

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

Requested by developers

2. Finding previously unknown bugs

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

Important file systems

2. Finding previously unknown bugs

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

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