Fine-Grained Fault Tolerance using Device Checkpoints

Asim Kadav
with Matthew Renzelmann and Michael M. Swift
University of Wisconsin-Madison
The (old) elephant in the room

3rd party developers

+ device drivers
(majority of kernel code)

OS kernel
The (old) elephant in the room

3rd party developers

+ device drivers
  (majority of kernel code)

OS kernel
The (old) elephant in the room

3rd party developers + device drivers (majority of kernel code) = Recipe for disaster
### Extensive past work on reliability research

<table>
<thead>
<tr>
<th>Improvement</th>
<th>System</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Drivers</td>
</tr>
<tr>
<td><strong>Isolation</strong></td>
<td>Nooks [SOSP 03]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>XFI [OSDI 06]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CuriOS [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td><strong>Type Safety</strong></td>
<td>SafeDrive [OSDI 06]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Singularity [Eurosys 06]</td>
<td>1</td>
</tr>
<tr>
<td><strong>Specification</strong></td>
<td>Nexus [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Termite [SOSP 09]</td>
<td>2</td>
</tr>
<tr>
<td><strong>Recovery</strong></td>
<td>Shadow Drivers [OSDI 04]</td>
<td>13</td>
</tr>
<tr>
<td><strong>Static analysis tools</strong></td>
<td>Windows SDV [Eurosys 06]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Coverity [CACM 10]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Cocinelle [Eurosys 08]</td>
<td>All</td>
</tr>
</tbody>
</table>
Extensive past work on reliability research

<table>
<thead>
<tr>
<th>Improvement</th>
<th>System</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Drivers</td>
</tr>
<tr>
<td>Isolation</td>
<td>Nooks [SOSP 03]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>XFI [OSDI 06]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CuriOS [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td>Type Safety</td>
<td>SafeDrive [OSDI 06]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Singularity [Eurosys 06]</td>
<td>1</td>
</tr>
<tr>
<td>Specification</td>
<td>Nexus [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Termite [SOSP 09]</td>
<td>2</td>
</tr>
<tr>
<td>Recovery</td>
<td>Shadow Drivers [OSDI 04]</td>
<td>13</td>
</tr>
<tr>
<td>Static analysis tools</td>
<td>Windows SDV [Eurosys 06]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Coverity [CACM 10]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Cocinelle [Eurosys 08]</td>
<td>All</td>
</tr>
</tbody>
</table>

Note: The table shows the number of drivers, buses, and classes for various systems and improvement areas. The last row indicates static analysis tools and their comprehensive coverage.
### Extensive past work on reliability research

<table>
<thead>
<tr>
<th>Improvement</th>
<th>System</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Drivers</td>
</tr>
<tr>
<td>Isolation</td>
<td>Nooks [SOSP 03]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>XFI [OSDI 06]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CuriOS [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td>Type Safety</td>
<td>SafeDrive [OSDI 06]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Singularity [Eurosys 06]</td>
<td>1</td>
</tr>
<tr>
<td>Specification</td>
<td>Nexus [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Termite [SOSP 09]</td>
<td>2</td>
</tr>
<tr>
<td>Recovery</td>
<td>Shadow Drivers [OSDI 04]</td>
<td>13</td>
</tr>
<tr>
<td>Static analysis tools</td>
<td>Windows SDV [Eurosys 06]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Coverity [CACM 10]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Cocinelle [Eurosys 08]</td>
<td>All</td>
</tr>
</tbody>
</table>
Observation 1: Solutions that limit changes to kernel and apply to lots of drivers have real impact

<table>
<thead>
<tr>
<th>Improvement</th>
<th>System</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Drivers</td>
</tr>
<tr>
<td>Isolation</td>
<td>Nooks [SOSP 03]</td>
<td>6</td>
</tr>
<tr>
<td>Type Safety</td>
<td>SafeDrive [OSDI 06]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Singularity [Eurosys 06]</td>
<td>1</td>
</tr>
<tr>
<td>Specification</td>
<td>Nexus [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Termite [SOSP 09]</td>
<td>2</td>
</tr>
<tr>
<td>Recovery</td>
<td>Shadow Drivers [OSDI 04]</td>
<td>13</td>
</tr>
<tr>
<td>Static analysis tools</td>
<td>Windows SDV [Eurosys 06]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Coverity [CACM 10]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Cocinelle [Eurosys 08]</td>
<td>All</td>
</tr>
</tbody>
</table>

Extensive past work on reliability research
Extensive past work on reliability research

<table>
<thead>
<tr>
<th>Improvement</th>
<th>System</th>
<th>Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Drivers</td>
</tr>
<tr>
<td>Isolation</td>
<td>Nooks [SOSP 03]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>XFI [OSDI 06]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>CuriOS [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td>Type Safety</td>
<td>SafeDrive [OSDI 06]</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Singularity [Eurosys 06]</td>
<td>1</td>
</tr>
<tr>
<td>Specification</td>
<td>Nexus [OSDI 08]</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Termite [SOSP 09]</td>
<td>2</td>
</tr>
<tr>
<td>Recovery</td>
<td>Shadow Drivers [OSDI 04]</td>
<td>13</td>
</tr>
<tr>
<td>Static analysis tools</td>
<td>Windows SDV [Eurosys 06]</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Coverity [CACM 10]</td>
<td>All</td>
</tr>
</tbody>
</table>

**Observation 2:** Most systems focus on improving isolation and detection and not on recovery.
Driver failure recovery limited to driver restart

- Restart driver upon failure
  - Safedrive and MINIX approach
  - Can break applications
Driver failure recovery limited to driver restart

- Restart driver upon failure
  - Safedrive and MINIX approach
  - Can break applications
Driver failure recovery limited to driver restart

★ Restart driver upon failure
  ★ Safedrive and MINIX approach
  ★ Can break applications

★ Restart and replay upon failure
  ★ Shadow driver approach
  ★ Always record state of driver
  ★ Perform restart and log replay upon failure
  ★ Transparent to applications
Problem 1: Restart based driver recovery is slow

The problem is that the restart times are significantly longer than expected. The chart shows the following restart times for different devices:

- **8139too**: 500ms
- **e1000**: 1,500ms
- **ens1371**: 1,000ms
- **psmouse**: 0ms

These restart times are much slower than what is typically expected, indicating a potential issue with the driver recovery process.
Problem 1: Restart based driver recovery is slow

Shadow drivers restart the driver upon failure which can be slow.

Restart times

- 0ms
- 500ms
- 1,000ms
- 1,500ms
- 2,000ms

Devices:
- 8139too (net)
- e1000 (net)
- ens1371 (sound)
- psmouse (input)
Driver re-initialization probes hardware again

Allocate device structures

Map BAR and I/O ports

Register device operations

Detect chipset capabilities

Cold boot device

Device ready

Configure device

Set chipset specific ops

Device self test

Verify EEPROM checksum

Device ready
Driver re-initialization probes hardware again

- **Allocate device structures**
- Map BAR and I/O ports
- Register device operations
- Detect chipset capabilities
- Cold boot device

- **Device ready**
- Configure device
- Set chipset specific ops
- Device self test
- Verify EEPROM checksum

Device ready
Driver re-initialization probes hardware again

Allocate device structures
- Map BAR and I/O ports
- Register device operations
- Detect chipset capabilities
- Cold boot device

Device ready
- Configure device
- Set chipset specific ops
- Device self test
- Verify EEPROM checksum

★ What does slow device re-initialization hurt?
★ Fault tolerance: Driver recovery
★ Virtualization: Live migration
★ OS functions: Fast reboot
Problem 2: Shadow drivers assume drivers follow class behavior

Class definition includes:

* Callbacks registered with the bus, device and kernel subsystem
Problem 2: Shadow drivers assume drivers follow class behavior

Class definition includes:

- Callbacks registered with the bus, device and kernel subsystem

How many drivers follow class behavior and how much code does this add and
Problem 2(a): Drivers do behave outside class definitions

★ Non-class behavior that affects recovery:
- procfs/sysfs interactions and unique ioctls

Windows WLAN card config via private ioctls

$ echo 1 > /sys/class/sound/mixer/device/enable

Linux sound card config via sysfs
Problem 2(a): Drivers do behave outside class definitions

★ Non-class behavior that affects recovery:
- procfs/sysfs interactions and unique ioctls

Windows WLAN card config via private ioctl

At least 16% of drivers have non-class behavior and may not recover correctly using shadow drivers

Linux sound card config via sysfs

$ echo 1 > /sys/class/sound/mixer/device/enable
Problem 2(b): Too many classes

“Understanding Modern Device Drivers” ASPLOS 2012
Problem 2(b): Too many classes

Class-specific driver recovery leads to a large kernel recovery subsystem

★ “Understanding Modern Device Drivers” ASPLOS 2012
Fine-Grained Fault Tolerance (FGFT)
Fine-grained Isolation

★ Runs driver entry points like transactions
★ Relies on code generation to limit new code in kernel
**Fine-Grained Fault Tolerance (FGFT)**

### Fine-grained Isolation
- Runs driver entry points like transactions
- Relies on code generation to limit new code in kernel

### Checkpoint-based recovery
- Provides fast and correct recovery semantics
Fine-Grained Fault Tolerance (FGFT)

**Fine-grained Isolation**
- Runs driver entry points like transactions
- Relies on code generation to limit new code in kernel
- Requires incremental overhead/changes to drivers
- Shifts burden of fault tolerance to faulty code

**Checkpoint-based recovery**
- Provides fast and correct recovery semantics
Outline

Introduction

Fine-grained isolation

Checkpoint-based recovery

Evaluation and Conclusions
Unit of fault tolerance: Driver entry point
Unit of fault tolerance: **Driver entry point**

**whole driver isolation**

- probe
- xmit
- config

**network driver**

**network card**
Unit of fault tolerance: Driver entry point

network
driver

probe
xmit
config

network
card
Unit of fault tolerance: Driver entry point
Provide fault tolerance to specific driver entry points
Unit of fault tolerance: Driver entry point

- Provide fault tolerance to specific driver entry points
- Can be applied to untested code or code marked suspicious by static or runtime tools
Transactional support through code generation

```
netdev
get ringparam
network
driver
```
Transactional support through code generation

get ringparam

network driver

SFI network driver
Transactional support through code generation

**network driver**

get ringparam

**SFI network driver**
Transactional support through code generation

**Range Table**

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>

**Diagram**

- *netdev*
- *network driver*
- *SFI network driver*
- *get ringparam*
- *Range Table*
Transactional support through code generation

- Detects and recovers from:
  - Memory errors like invalid pointer accesses
  - Structural errors like malformed structures
  - Processor exceptions like divide by zero, stack corruption

### Range Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>
Transactional support through code generation

Detection and recovery from:
- Memory errors like invalid pointer accesses
- Structural errors like malformed structures
- Processor exceptions like divide by zero, stack corruption

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xfffffa00</td>
<td>Read</td>
</tr>
<tr>
<td>0xfffffa08</td>
<td>Write</td>
</tr>
<tr>
<td>0xfffffa0a</td>
<td>Read</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- Fine-grained isolation
- Checkpoint-based recovery
- Conclusion
Checkpointing drivers is hard

★ Easy to capture memory state
Checkpointing drivers is hard

★ Easy to capture memory state

checkpoint

network driver

network card
Checkpointing drivers is hard

★ Easy to capture memory state

★ Device state is not captured
  ★ Device configuration space
Checkpointing drivers is hard

★ Easy to capture memory state

Device state is not captured
★ Device configuration space
★ Internal device registers and counters
Checkpointing drivers is hard

★ Easy to capture memory state

★ Device state is not captured
  ★ Device configuration space
  ★ Internal device registers and counters
  ★ Memory buffer addresses used for DMA
Checkpointing drivers is hard

- Easy to capture memory state

- Device state is not captured
  - Device configuration space
  - Internal device registers and counters
  - Memory buffer addresses used for DMA
  - Unique for every device
Checkpointing drivers is hard

- Easy to capture memory state

Intuition: Operating systems already capture device state during power management

- Device state is not captured
  - Device configuration space
  - Internal device registers and counters
  - Memory buffer addresses used for DMA
- Unique for every device
Intuition with power management

- Refactor power management code for device checkpoints
  - Correct: Developer captures unique device semantics
  - Fast: Avoids probe and latency critical for applications

- Ask developers to export checkpoint/restore in their drivers
Device checkpoint/restore from PM code

**Suspend**
- Save config state
- Save register state
- Disable device
- Save DMA state
- Suspend device

**Resume**
- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready
Device checkpoint/restore from PM code

Suspend

- Save config state
- Save register state
- Save DMA state
- Suspend device

Resume

- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready
Device checkpoint/restore from PM code

**Suspend**
- Save config state
- Save register state
- Save DMA state

**Resume**
- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready
Device checkpoint/restore from PM code

**Suspend**
- Save config state
- Save register state
- Save DMA state

**Resume**
- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready
Device checkpoint/restore from PM code

**Checkpoint**
- Save config state
- Save register state
- Save DMA state

**Resume**
- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
- Device Ready
Device checkpoint/restore from PM code

**Checkpoint**
- Save config state
- Save register state
- Save DMA state

**Resume**
- Restore config state
- Restore register state
- Restore or reset DMA state
- Re-attach/Enable device
Device checkpoint/restore from PM code

**Checkpoint**

- Save config state
- Save register state
- Save DMA state

**Resume**

- Restore config state
- Restore register state
- Restore or reset DMA state
Device checkpoint/restore from PM code

**Checkpoint**
- Save config state
- Save register state
- Save DMA state

**Restore**
- Restore config state
- Restore register state
- Restore or reset DMA state
Device checkpoint/restore from PM code

**Checkpoint**
- Save config state
- Save register state
- Save DMA state

**Restore**
- Restore config state
- Restore register state
- Restore or reset DMA state

Suspend/resume code provides device checkpoint functionality
Synergy of isolation and fast checkpoints

netdev

network
driver
Synergy of isolation and fast checkpoints

*netdev*

*network driver*

*xmit*
Synergy of isolation and fast checkpoints

- `netdev`
- `network driver`
- `get ringparam`
Synergy of isolation and fast checkpoints

C

get ringparam

netdev

network driver
Synergy of isolation and fast checkpoints

- netdev
- network driver
- SFI network driver

get ringparam
Synergy of isolation and fast checkpoints
Synergy of isolation and fast checkpoints
Synergy of isolation and fast checkpoints

get ringparam

network driver

SFI network driver

C

netdev

netdev

stubs

stubs
Synergy of isolation and fast checkpoints

![Diagram showing network driver and SFI connections]

Range Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>
Synergy of isolation and fast checkpoints

Range Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>

get ringparam
Synergy of isolation and fast checkpoints

Range Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>
Synergy of isolation and fast checkpoints

Range Table

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>

get ringparam
Synergy of isolation and fast checkpoints

FGFT provides transactional execution of driver entry points

<table>
<thead>
<tr>
<th>Address</th>
<th>Access rights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffffa000</td>
<td>Read</td>
</tr>
<tr>
<td>0xffffa008</td>
<td>Write</td>
</tr>
<tr>
<td>0xffffa00a</td>
<td>Read</td>
</tr>
</tbody>
</table>
How does this give us transactional execution?
How does this give us transactional execution?

- Atomicity: All or nothing execution
  - Driver state: Run code in SFI module
  - Device state: Explicitly checkpoint/restore state
How does this give us transactional execution?

- Atomicity: All or nothing execution
  - Driver state: Run code in SFI module
  - Device state: Explicitly checkpoint/restore state

- Isolation: Serialization to hide incomplete transactions
  - Re-use existing device locks to lock driver
  - Two phase locking
How does this give us transactional execution?

- **Atomicity:** All or nothing execution
  - **Driver state:** Run code in SFI module
  - **Device state:** Explicitly checkpoint/restore state

- **Isolation:** Serialization to hide incomplete transactions
  - **Re-use existing device locks to lock driver**
  - **Two phase locking**

- **Consistency:** Only valid (kernel, driver and device) states
  - **Higher level mechanisms to rollback external actions**
  - **At most once device action guarantee to applications**
Outline

- Introduction
- Fine-grained isolation
- Checkpoint-based recovery
- Evaluation & Conclusions
Evaluation platform

★ Criterion :
★ Latency of recovery: How fast is it?
★ Correctness of recovery: How well does it work?
★ Incremental effort: How much work is it?
★ Performance: How much does it cost?
Evaluation platform

★ Criterion:
★ Latency of recovery: How fast is it?
★ Correctness of recovery: How well does it work?
★ Incremental effort: How much work is it?
★ Performance: How much does it cost?

★ Platform:
★ Implemented in Linux 2.6.29
★ 2.5 GHz Intel Core 2 Quad core w/ 4 GB DDR2 DRAM
★ Six drivers across three classes

<table>
<thead>
<tr>
<th>Driver</th>
<th>Class</th>
<th>Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>net</td>
<td>PCI</td>
</tr>
<tr>
<td>e1000</td>
<td>net</td>
<td>PCI</td>
</tr>
<tr>
<td>r8169</td>
<td>net</td>
<td>PCI</td>
</tr>
<tr>
<td>pegasus</td>
<td>net</td>
<td>USB</td>
</tr>
<tr>
<td>psmouse</td>
<td>sound</td>
<td>PCI</td>
</tr>
<tr>
<td>ens1371</td>
<td>input</td>
<td>serio</td>
</tr>
</tbody>
</table>
Recovery speedup

<table>
<thead>
<tr>
<th>Recovery times</th>
<th>0ms</th>
<th>500ms</th>
<th>1,000ms</th>
<th>1,500ms</th>
<th>2,000ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>310.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e1000</td>
<td>150.00</td>
<td>120.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pegasus</td>
<td>120.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r8169</td>
<td>1030.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ens1371</td>
<td>680.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>psmouse</td>
<td>1800.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Restart recovery
FGFT recovery
Recovery speedup

![Recovery speeds comparison chart]

- **Restart recovery**
- **FGFT recovery**

- **Recovery times**
  - 0ms: 310.00, 295.00, 150.00, 120.00, 115.00, 115.00
  - 500ms: 1800.00, 1600.00, 1050.00, 1030.00, 1030.00, 1030.00
  - 1000ms: 680.00, 680.00, 680.00, 680.00, 680.00, 680.00
  - 1500ms: 1030.00, 1030.00, 1030.00, 1030.00, 1030.00, 1030.00
  - 2000ms: 680.00, 680.00, 680.00, 680.00, 680.00, 680.00

- **Devices**
  - 8139too
  - e1000
  - pegasus
  - r8169
  - ens1371
  - psmouse
FGFT provides significant speedup in driver recovery and improves system availability.
## Static and dynamic fault injection

<table>
<thead>
<tr>
<th>Driver</th>
<th>Injected Faults</th>
<th>Native Crashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>e1000</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>r8169</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>pegasus</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>ens1371</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>psmouse</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>258</strong></td>
<td><strong>256</strong></td>
</tr>
</tbody>
</table>
## Static and dynamic fault injection

<table>
<thead>
<tr>
<th>Driver</th>
<th>Injected Faults</th>
<th>Native Crashes</th>
<th>FGFT Crashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>43</td>
<td>43</td>
<td>NONE</td>
</tr>
<tr>
<td>e1000</td>
<td>47</td>
<td>47</td>
<td>NONE</td>
</tr>
<tr>
<td>r8169</td>
<td>36</td>
<td>36</td>
<td>NONE</td>
</tr>
<tr>
<td>pegasus</td>
<td>34</td>
<td>33</td>
<td>NONE</td>
</tr>
<tr>
<td>ens1371</td>
<td>22</td>
<td>21</td>
<td>NONE</td>
</tr>
<tr>
<td>psmouse</td>
<td>46</td>
<td>46</td>
<td>NONE</td>
</tr>
<tr>
<td>TOTAL</td>
<td>258</td>
<td>256</td>
<td>NONE</td>
</tr>
</tbody>
</table>
## Static and dynamic fault injection

<table>
<thead>
<tr>
<th>Driver</th>
<th>Injected Faults</th>
<th>Native Crashes</th>
<th>FGFT Crashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>43</td>
<td>43</td>
<td>NONE</td>
</tr>
<tr>
<td>e1000</td>
<td>47</td>
<td>47</td>
<td>NONE</td>
</tr>
<tr>
<td>r8169</td>
<td>36</td>
<td>36</td>
<td>NONE</td>
</tr>
<tr>
<td>pegasus</td>
<td>34</td>
<td>33</td>
<td>NONE</td>
</tr>
<tr>
<td>ens1371</td>
<td>22</td>
<td>21</td>
<td>NONE</td>
</tr>
<tr>
<td>psmouse</td>
<td>46</td>
<td>46</td>
<td>NONE</td>
</tr>
<tr>
<td>TOTAL</td>
<td>258</td>
<td>256</td>
<td>NONE</td>
</tr>
</tbody>
</table>

**FGFT recovers from multiple failures:** 1) restores non-class state and 2) does not affect other threads.
## Programming effort

<table>
<thead>
<tr>
<th>Driver</th>
<th>LOC</th>
<th>Isolation annotations</th>
<th>Recovery additions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Driver annotations</td>
<td>Kernel annotations</td>
</tr>
<tr>
<td>8139too</td>
<td>1,904</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>e1000</td>
<td>13,973</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>r8169</td>
<td>2,993</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>pegasus</td>
<td>1,541</td>
<td>26</td>
<td>12</td>
</tr>
<tr>
<td>ens1371</td>
<td>2,110</td>
<td>23</td>
<td>66</td>
</tr>
<tr>
<td>psmouse</td>
<td>2,448</td>
<td>11</td>
<td>19</td>
</tr>
</tbody>
</table>
Programming effort

<table>
<thead>
<tr>
<th>Driver</th>
<th>LOC</th>
<th>Isolation annotations</th>
<th>Recovery additions</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>1,904</td>
<td>15</td>
<td>26</td>
</tr>
<tr>
<td>e1000</td>
<td>13,973</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>r8169</td>
<td>2,993</td>
<td>10</td>
<td>17</td>
</tr>
<tr>
<td>pegasus</td>
<td>1,541</td>
<td>26</td>
<td>22</td>
</tr>
<tr>
<td>ens1371</td>
<td>2,110</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>psmouse</td>
<td>2,448</td>
<td>11</td>
<td>19</td>
</tr>
</tbody>
</table>

FGFT requires a loadable kernel module (1200 LOC) and 38 lines of kernel changes to trap processor exceptions
Throughput with isolation and recovery

- Native
- FGFT-I/O-all
- FGFT-off-I/O
- FGFT-I/O-1/2

netperf on Intel quad-core machines
Throughput with isolation and recovery

Throughput %age (Baseline 844 Mbps)

- Native
- FGFT-I/O-all
- FGFT-off-I/O
- FGFT-I/O-1/2

e1000 Network Card

netperf on Intel quad-core machines
Throughput with isolation and recovery

**CPU:** 2.4%

Throughput % (Baseline 844 Mbps)

Native
FGFT-I/O-all
FGFT-off-I/O
FGFT-I/O-1/2

*netperf on Intel quad-core machines*
Throughput with isolation and recovery

Throughput % (Baseline 844 Mbps)

**CPU**: 2.4% 2.4%

Native
FGFT-I/O-all
FGFT-off-I/O
FGFT-I/O-1/2

*netperf on Intel quad-core machines*
Throughput with isolation and recovery

Throughput %age (Baseline 844 Mbps)

CPU: 2.4% 2.4% 3.4%

100 93 100

0 25 50 75 100

Native FGFT-I/O-all FGFT-off-I/O FGFT-I/O-1/2

e1000 Network Card

netperf on Intel quad-core machines
Throughput with isolation and recovery

Throughput %age (Baseline 844 Mbps)

CPU: 2.4%  2.4%  3.4%  2.9%

100  93  100  96

0  25  50  75  100

e1000 Network Card

netperf on Intel quad-core machines
Throughput with isolation and recovery

FGFT can isolate and recover high bandwidth devices at low overhead without adding kernel subsystems

netperf on Intel quad-core machines
Summary

- FGFT runs driver code as transactions
  - Provides fault tolerance at incremental performance and programmer efforts

- Introduced device checkpoints
  - Provides fast and complete recovery semantics

- Fast device checkpoints should be explored in other domains like fast reboot, upgrade etc.
Questions

Asim Kadav
★ http://cs.wisc.edu/~kadav
★ kadav@cs.wisc.edu
★ Graduating in spring!
Extra slides

- Unlike suspend, devices continue to be accessed after a checkpoint
  - Rely on drivers following ACPI specifications for correctness
## Latency for device checkpoint/restore

<table>
<thead>
<tr>
<th>Driver</th>
<th>Class</th>
<th>Bus</th>
<th>Checkpoint Times</th>
<th>Restore Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>8139too</td>
<td>net</td>
<td>PCI</td>
<td>33μs</td>
<td>62μs</td>
</tr>
<tr>
<td>e1000</td>
<td>net</td>
<td>PCI</td>
<td>32μs</td>
<td>280ms</td>
</tr>
<tr>
<td>r8169</td>
<td>net</td>
<td>PCI</td>
<td>26μs</td>
<td>30μs</td>
</tr>
<tr>
<td>pegasus</td>
<td>net</td>
<td>USB</td>
<td>0μs</td>
<td>4ms</td>
</tr>
<tr>
<td>ens1371</td>
<td>sound</td>
<td>PCI</td>
<td>33μs</td>
<td>111ms</td>
</tr>
<tr>
<td>psmouse</td>
<td>input</td>
<td>serio</td>
<td>0μs</td>
<td>390ms</td>
</tr>
</tbody>
</table>

**Fast checkpoint/restore using suspend/resume**
Transforming drivers to run as FGFT

```c
if (c==0) {
  print("Driver init");
}
```

Driver with annotations

Static modifications
Transforming drivers to run as FGFT

If (c==0) {
    print ("Driver init");
}

Driver with annotations

Source transformation (adds driver transactions)

User supplied annotations

Static modifications
Transforming drivers to run as FGFT

Source transformation (adds driver transactions)

Main driver module

SFI driver module

SFI = software fault isolated

Driver with annotations

User supplied annotations

Static modifications
Transforming drivers to run as FGFT

Static modifications

Driver with annotations

User supplied annotations

Source transformation (adds driver transactions)

Main driver module

SFI driver module

SFI = software fault isolated

Run-time support
Transforming drivers to run as FGFT

Source transformation (adds driver transactions)

User supplied annotations

Driver with annotations

Main driver module

SFI driver module

SFI = software fault isolated

Object tracking
Marshaling/Demarshaling
Kernel undo log
Communication and recovery support

Static modifications

Run-time support

If (c==0) {
    print ("Driver init");
    
}