Interrupt filtering

Paolo Pisati

Universita' Statale di Milano

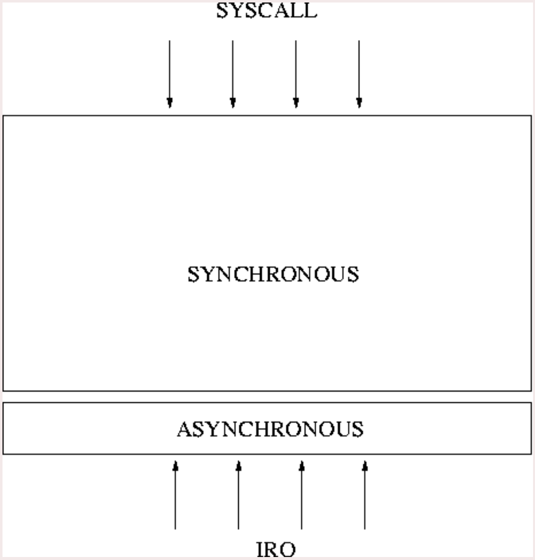
December 1, 2006

What's this talk about?

- Interrupt handling mechanism
- An interrupt is an asynchronous signal from hardware indicating the need for attention
- Key aspect: Synchronization policy

What's this talk about?

- Interrupt handling mechanism
- An interrupt is an asynchronous signal from hardware indicating the need for attention
- Key aspect: Synchronization policy


SYSCALL
SYNCHRONOUS
ASYNCHRONOUS
IRQ

What's this talk about?

- Interrupt handling mechanism
- An interrupt is an asynchronous signal from hardware indicating the need for attention
- Key aspect: Synchronization policy

Outline

1 FreeBSD 4x

2 FreeBSD SMPng

3 Interrupt filtering

4 Performance

5 Implementation details

Outline

1 **FreeBSD 4x**

2 FreeBSD SMPng

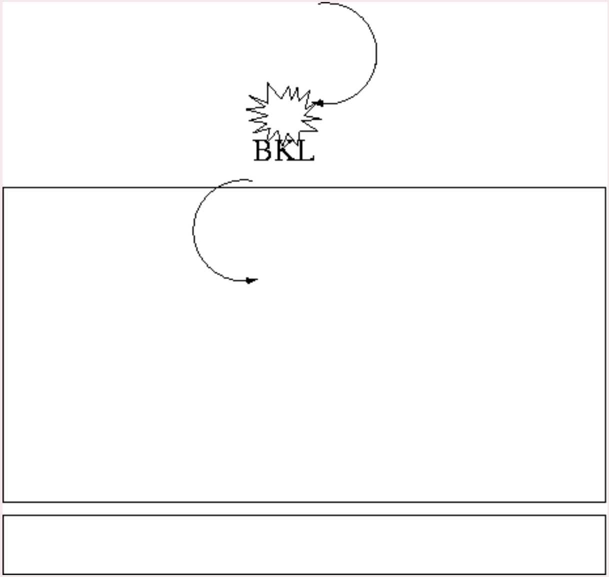
3 Interrupt filtering

4 Performance

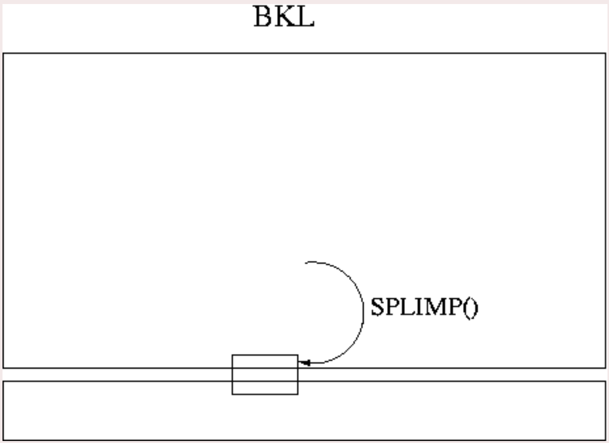
5 Implementation details

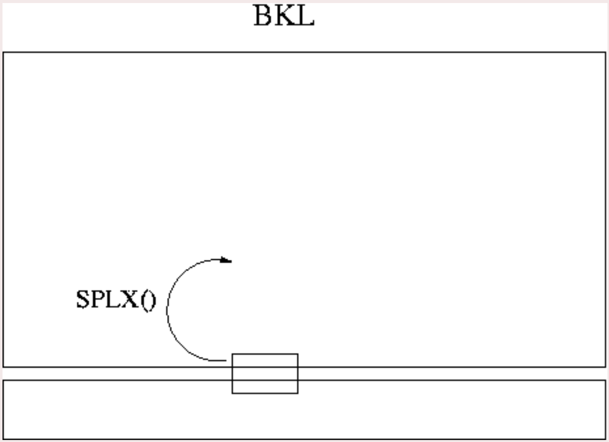
- Monothread kernel
- BKL ruled access to kernel space
- SPL calls used to synchronize top and bottom halves

- Monothread kernel
- BKL ruled access to kernel space
- SPL calls used to synchronize top and bottom halves


BKL

- Monothread kernel
- BKL ruled access to kernel space
- SPL calls used to synchronize top and bottom halves


BKL
SPLIMP()


BKL
SPLX()

Outline

1 FreeBSD 4x

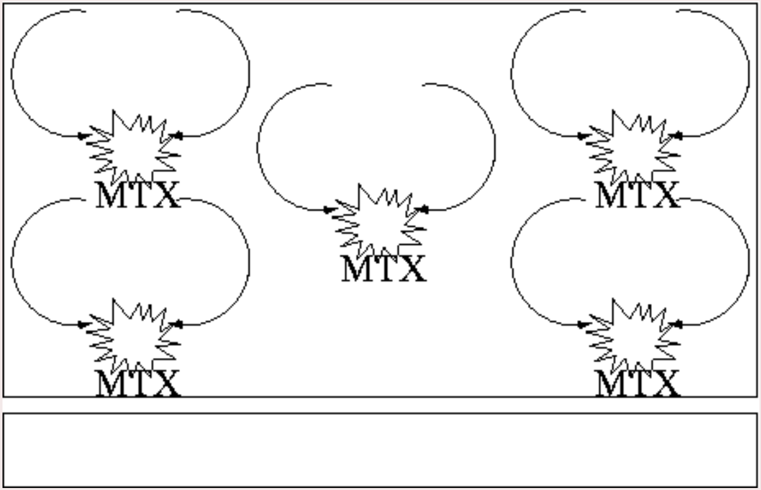
2 **FreeBSD SMPng**

3 Interrupt filtering

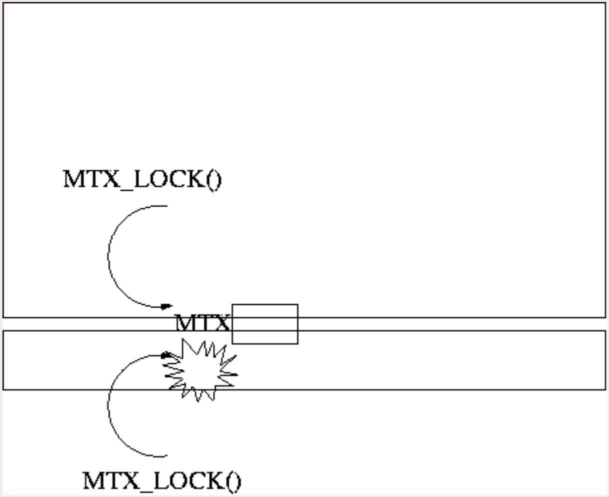
4 Performance

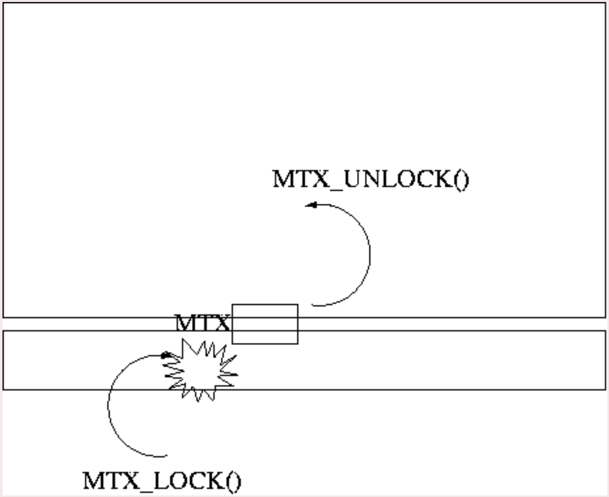
5 Implementation details

- Multithreaded kernel
- Interrupt handler got a private context: ithread
- Time critical handlers that don't block, can run in the context of the interrupted process (FAST)


MTX
MTX
MTX
MTX
MTX

- Multithreaded kernel
- Interrupt handler got a private context: ithread
- Time critical handlers that don't block, can run in the context of the interrupted process (FAST)


MTX_LOCK()
MTX
MTX_LOCK()


MTX_UNLOCK()
MTX
MTX_LOCK()

- Multithreaded kernel
- Interrupt handler got a private context: ithread
- Time critical handlers that don’t block, can run in the context of the interrupted process (FAST)

The problems

- Interrupt latency
- ...even worse with shared irq
- Interrupt masking at controller level

The problems

- Interrupt latency
- ...even worse with shared irq
- Interrupt masking at controller level

The problems

- Interrupt latency
- ...even worse with shared irq
- Interrupt masking at controller level

The problems

- Interrupt latency
- ...even worse with shared irq
- Interrupt masking at controller level

Outline

1 FreeBSD 4x

2 FreeBSD SMPng

3 **Interrupt filtering**

4 Performance

5 Implementation details

- Divide the interrupt handler in 2 logical pieces:

  FILTER runs in interrupt context, checks the received interrupt, serves it/delegates more work

  ITHREAD runs in ithread context, can block

- ack the interrupt after filter execution

- Divide the interrupt handler in 2 logical pieces:

  FILTER runs in interrupt context, checks the received interrupt, serves it/delegates more work

  ITHREAD runs in ithread context, can block

- ack the interrupt after filter execution

- Divide the interrupt handler in 2 logical pieces:

  FILTER runs in interrupt context, checks the received interrupt, serves it/delegates more work

  ITHREAD runs in ithread context, can block

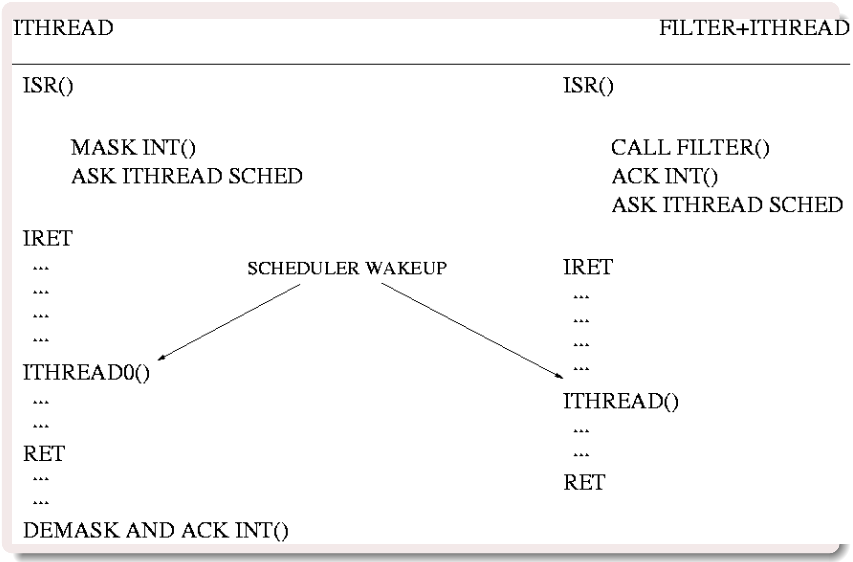
- ack the interrupt after filter execution

- Divide the interrupt handler in 2 logical pieces:

  FILTER runs in interrupt context, checks the received interrupt, serves it/delegates more work

  ITHREAD runs in ithread context, can block

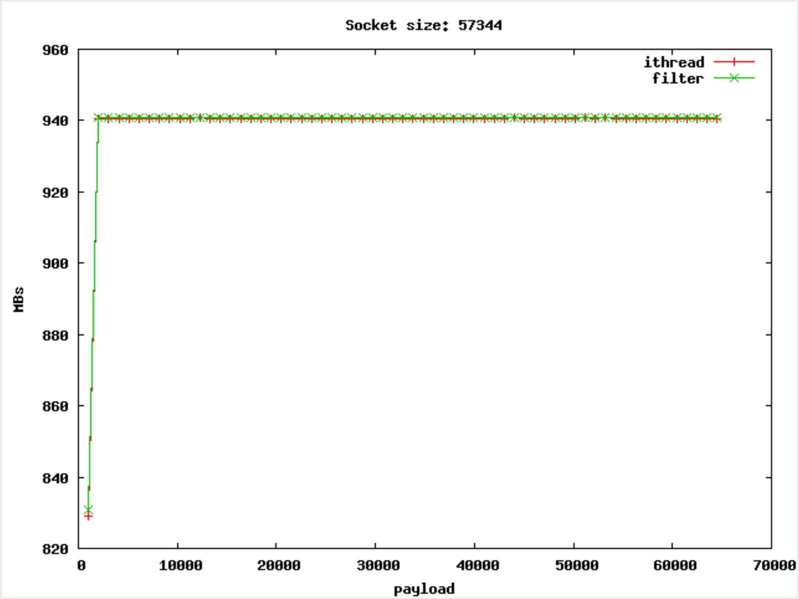
- ack the interrupt after filter execution

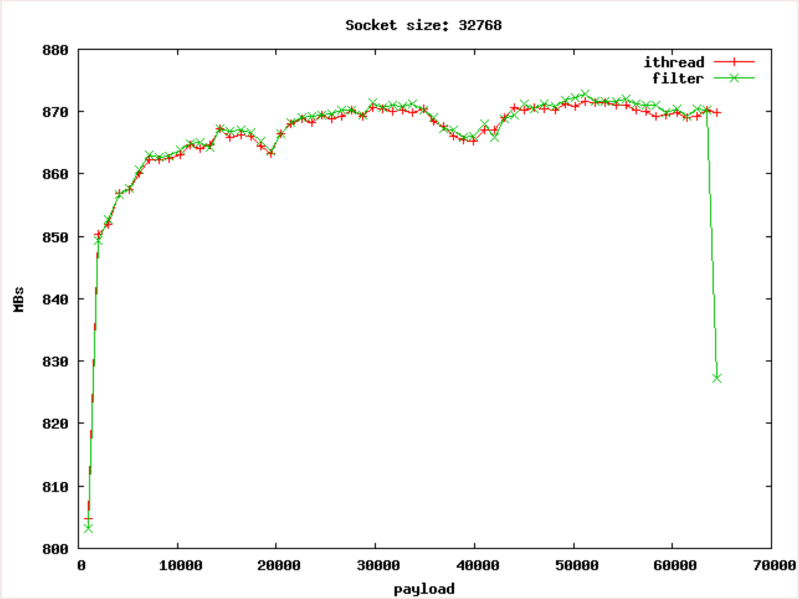

ITHREAD
FILTER+ITHREAD
ISR()
MASK INT()
ASK ITHREAD SCHED
IRET
...
...
...
...
ITHREAD0()
...
...
RET
...
...
DEMASK AND ACK INT()
SCHEDULER WAKEUP
ISR()
CALL FILTER()
ACK INT()
ASK ITHREAD SCHED
IRET
...
...
...
...
ITHREAD()
...
...
RET

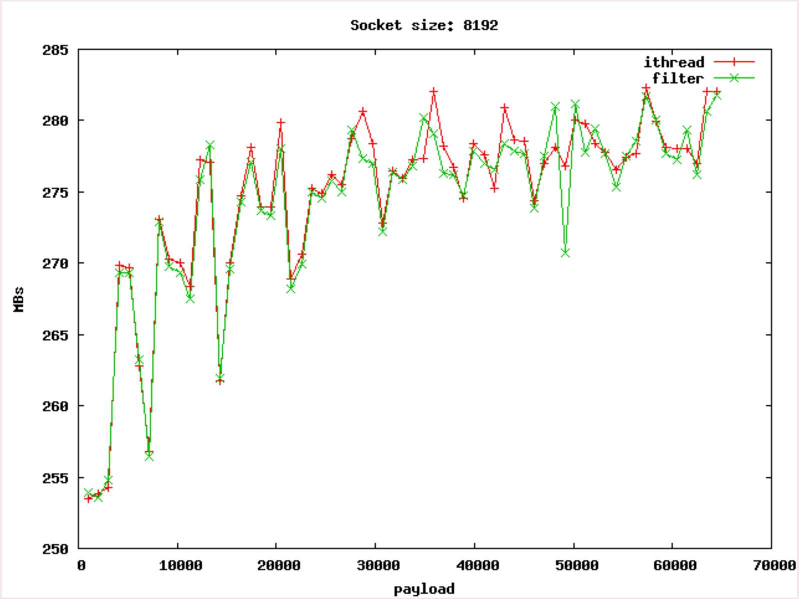
Let’s see some code: bfe, em, re, aac, xl

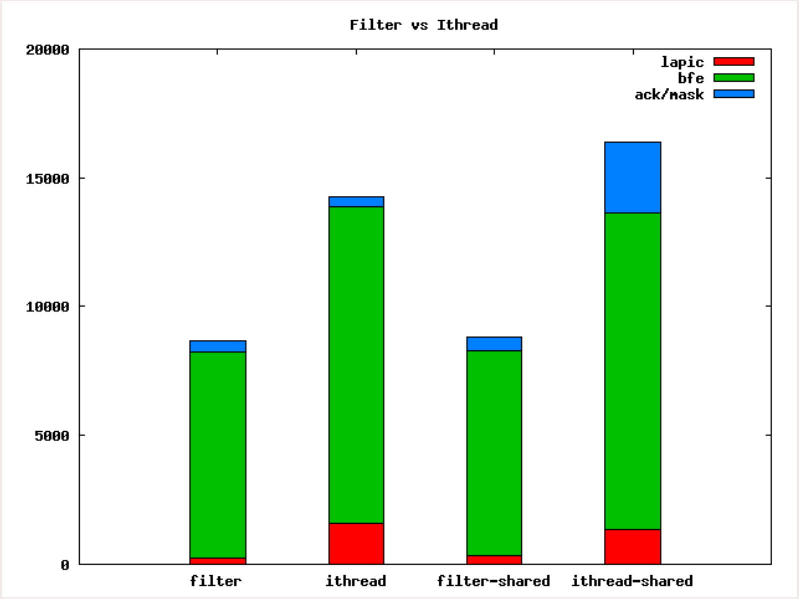
Outline

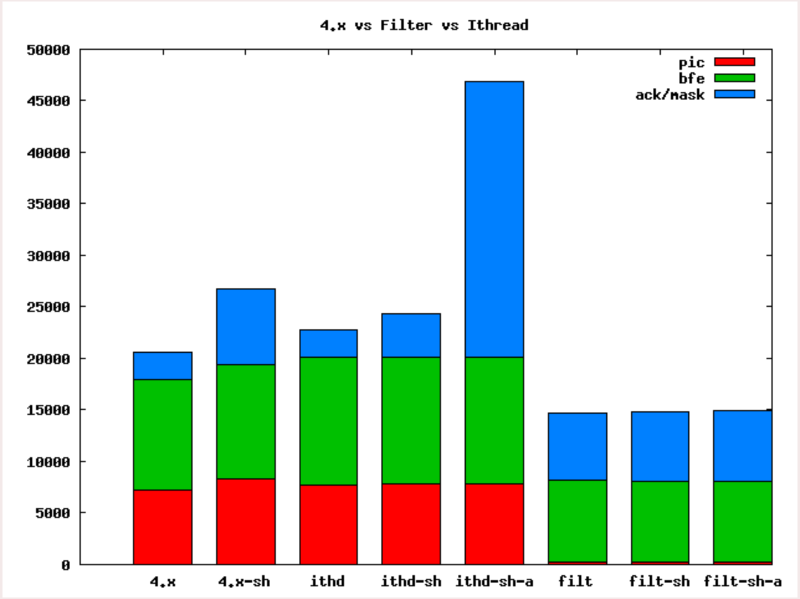
1 FreeBSD 4x
2 FreeBSD SMPng
3 Interrupt filtering
4 **Performance**
5 Implementation details


Socket size: 57344
ithread
filter
MBs
payload
960
940
920
900
880
860
840
820
0
10000
20000
30000
40000
50000
60000
70000


Socket size: 32768
ithread
filter
MBs
payload
880
870
860
850
840
830
820
810
800
0
10000
20000
30000
40000
50000
60000
70000


Socket size: 8192
ithread
filter
MBs
payload
285
280
275
270
265
260
255
250
0
10000
20000
30000
40000
50000
60000
70000


Filter vs Ithread
20000
15000
10000
5000
0
lapic
bfe
ack/mask
filter
ithread
filter-shared
ithread-shared


4.x vs Filter vs Ithread
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
pic
bfe
ack/mask
4.x
4.x-sh
ithd
ithd-sh
ithd-sh-a
filt
filt-sh
filt-sh-a

Outline

1. FreeBSD 4x
2. FreeBSD SMPng
3. Interrupt filtering
4. Performance
5. **Implementation details**

Implementation details

- the total patch against HEAD is about 300kb
- newbus API change: bus_setup_intr()
- 3 different ways to handle interrupts:

Implementation details

- the total patch against HEAD is about 300kb
- newbus API change: bus_setup_intr()
- 3 different ways to handle interrupts:

-
  ```
  int
  bus_setup_intr(
      device_t dev,
      struct resource *r,
      int flags,
      driver_filter_t filter,
      driver_intr_t handler,
      void *arg,
      void **cookiep
  );
  ```
- `int driver_filter_t(void*);`

-
  ```
  int
  bus_setup_intr(
         device_t dev,
         struct resource *r,
         int flags,
         driver_filter_t filter,
         driver_intr_t handler,
         void *arg,
         void **cookiep
  );
  ```

- `int driver_filter_t(void*);`

Implementation details

- the total patch against HEAD is about 300kb
- newbus API change: bus_setup_intr()
- 3 different ways to handle interrupts:

FILTER (aka FAST) bus_setup_intr(dev, r, flags, filter, NULL, arg, cookiep);

ITHREAD bus_setup_intr(dev, r, flags, NULL, ithread, arg, cookiep);

FILTER+ITHREAD bus_setup_intr(dev, r, flags, filter, ithread, arg, cookiep);

Implementation details

- the total patch against HEAD is about 300kb
- newbus API change: bus_setup_intr()
- 3 different ways to handle interrupts:

FILTER (aka FAST) bus_setup_intr(dev, r, flags, filter, NULL, arg, cookiep);

ITHREAD bus_setup_intr(dev, r, flags, NULL, ithread, arg, cookiep);

FILTER+ITHREAD bus_setup_intr(dev, r, flags, filter, ithread, arg, cookiep);

Implementation details

- the total patch against HEAD is about 300kb
- newbus API change: bus_setup_intr()
- 3 different ways to handle interrupts:

FILTER (aka FAST) bus_setup_intr(dev, r, flags, filter, NULL, arg, cookiep);

ITHREAD bus_setup_intr(dev, r, flags, NULL, ithread, arg, cookiep);

FILTER+ITHREAD bus_setup_intr(dev, r, flags, filter, ithread, arg, cookiep);

Implementation details

- the total patch against HEAD is about 300kb
- newbus API change: bus_setup_intr()
- 3 different ways to handle interrupts:

FILTER (aka FAST) bus_setup_intr(dev, r, flags, filter, NULL, arg, cookiep);

ITHREAD bus_setup_intr(dev, r, flags, NULL, ithread, arg, cookiep);

FILTER+ITHREAD bus_setup_intr(dev, r, flags, filter, ithread, arg, cookiep);

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(2)

- filter have 3 returns code:

FILTER_STRAY	event not recognized
FILTER_HANDLED	interrupt is acked/turned off
FILTER_SCHEDULE_THREAD	schedule the ithread

- no other value can be returned with FILTER_STRAY
- if a filter wants to schedule an ithread, it returns FILTER_HANDLED | FILTER_SCHEDULE_THREAD
- if all filters returned FILTER_STRAY...
- or no handlers were registered on that line...
- a new interrupt mitigation mechanism will kick in

Implementation details(3)

- interrupt.h::struct intr_event was modified
- kern_intr.c::intr_event_create() was modified too
- int (*ie_pending)(void *);
- PowerPC MD code was modified to accept more than one FAST handler per line(previouslys INTR_FAST implied INTR_EXCL)

Implementation details(3)

- interrupt.h::struct intr_event was modified
- kern_intr.c::intr_event_create() was modified too
- int (*ie_pending)(void *);
- PowerPC MD code was modified to accept more than one FAST handler per line(previouslys INTR_FAST implied INTR_EXCL)

Implementation details(3)

- interrupt.h::struct intr_event was modified
- kern_intr.c::intr_event_create() was modified too
- int (*ie_pending)(void *);
- PowerPC MD code was modified to accept more than one FAST handler per line(previouslys INTR_FAST implied INTR_EXCL)

Implementation details(3)

- interrupt.h::struct intr_event was modified
- kern_intr.c::intr_event_create() was modified too
- int (*ie_pending)(void *);
- PowerPC MD code was modified to accept more than one FAST handler per line(previouslys INTR_FAST implied INTR_EXCL)

TODO

- Check the MD code that turn off/mask interrupts (arm, ia64, powerpc and sparc64)
- Check drivers that override newbus generic_bus_setup_intr() (dev/puc/puc.c)
- Much of the MD code in intr_execute_handlers() could be turned into MI
- Testing

TODO

- Check the MD code that turn off/mask interrupts (arm, ia64, powerpc and sparc64)
- Check drivers that override newbus generic_bus_setup_intr() (dev/puc/puc.c)
- Much of the MD code in intr_execute_handlers() could be turned into MI
- Testing

TODO

- Check the MD code that turn off/mask interrupts (arm, ia64, powerpc and sparc64)
- Check drivers that override newbus generic_bus_setup_intr() (dev/puc/puc.c)
- Much of the MD code in intr_execute_handlers() could be turned into MI
- Testing

TODO

- Check the MD code that turn off/mask interrupts (arm, ia64, powerpc and sparc64)
- Check drivers that override newbus generic_bus_setup_intr() (dev/puc/puc.c)
- Much of the MD code in intr_execute_handlers() could be turned into MI
- Testing

TODO(2)

- Do more measurements
- Develop some tools to measure interrupt latency
- Take a look at lock-free/wait-free data struct

TODO(2)

- Do more measurements
- Develop some tools to measure interrupt latency
- Take a look at lock-free/wait-free data struct

TODO(2)

- Do more measurements
- Develop some tools to measure interrupt latency
- Take a look at lock-free/wait-free data struct

Conclusion

- No performance regression against previous model
- Lower interrupt latency
- Less interrupt problems (“storms”)
- I wish to make it part of 7.x

Conclusion

- No performance regression against previous model
- Lower interrupt latency
- Less interrupt problems ("storms")
- I wish to make it part of 7.x

Conclusion

- No performance regression against previous model
- Lower interrupt latency
- Less interrupt problems (“storms”)
- I wish to make it part of 7.x

Conclusion

- No performance regression against previous model
- Lower interrupt latency
- Less interrupt problems (“storms”)
- I wish to make it part of 7.x

Conclusion

- No performance regression against previous model
- Lower interrupt latency
- Less interrupt problems (“storms”)
- I wish to make it part of 7.x