FreeBSD on IBM PowerNV

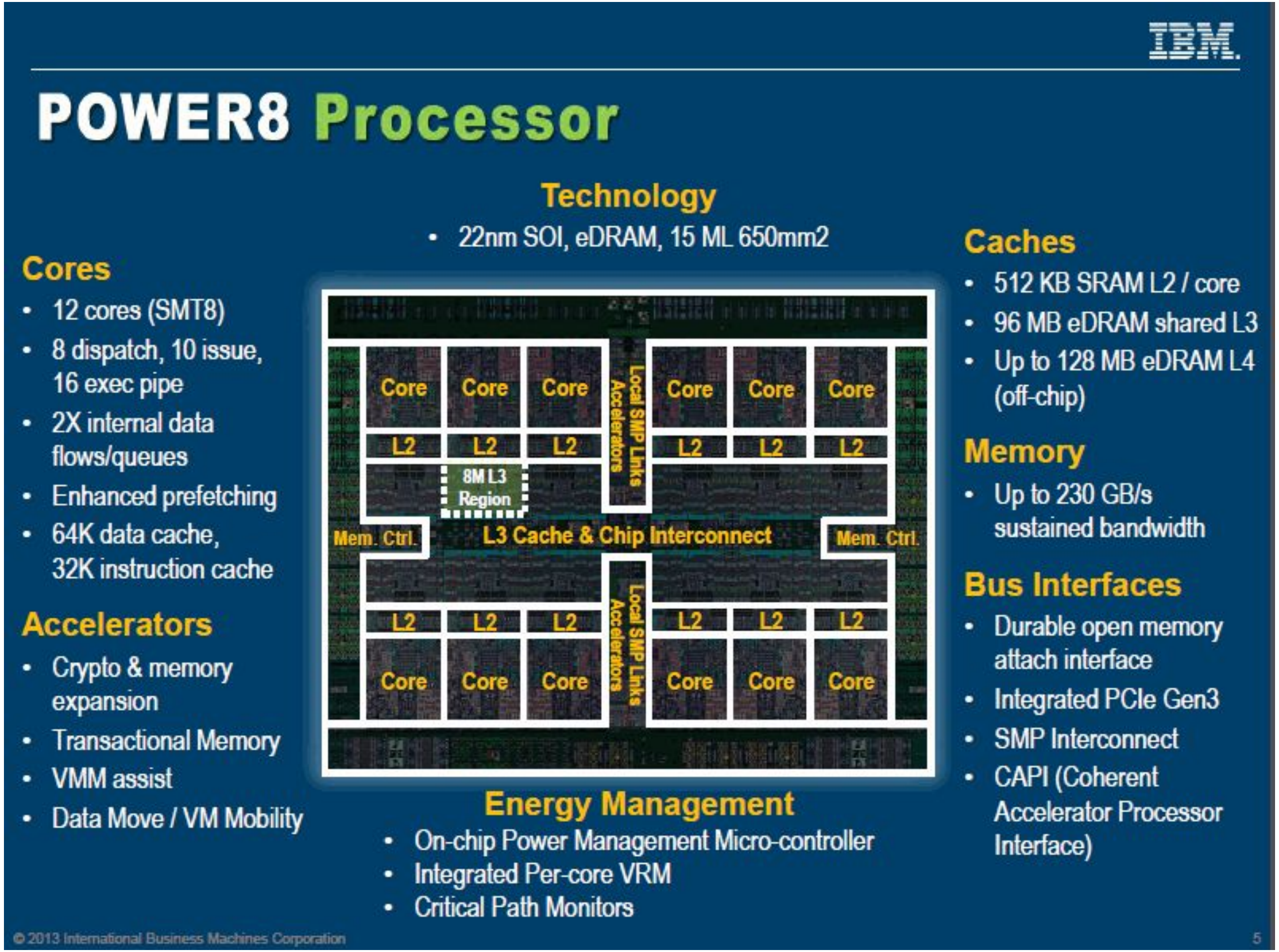
Patryk Duda
pdk@semihalf.com
Wojciech Macek
wma@FreeBSD.org, wma@semihalf.com
Michał Stanek
mst@semihalf.com

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - S821LC
- Power8 system internals
  - ABI and TOC
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- Current state and future work
- Performance measurements
- Q&A

Presentation plan

- Hardware platform
  - **Power8 and PowerNV**
  - S821LC
- Power8 system internals
  - ABI and TOC
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- Current state and future work
- Performance measurements
- Q&A



IBM

POWER8 Processor

Technology

- 22nm SOI, eDRAM, 15 ML 650mm2

Cores

- 12 cores (SMT8)
- 8 dispatch, 10 issue, 16 exec pipe
- 2X internal data flows/queues
- Enhanced prefetching
- 64K data cache, 32K instruction cache

Accelerators

- Crypto & memory expansion
- Transactional Memory
- VMM assist
- Data Move / VM Mobility

Core
L2
8M L3 Region
Local SMP Links Accelerators
Mem. Ctrl.
L3 Cache & Chip Interconnect
Mem. Ctrl.

Energy Management

- On-chip Power Management Micro-controller
- Integrated Per-core VRM
- Critical Path Monitors

Caches

- 512 KB SRAM L2 / core
- 96 MB eDRAM shared L3
- Up to 128 MB eDRAM L4 (off-chip)

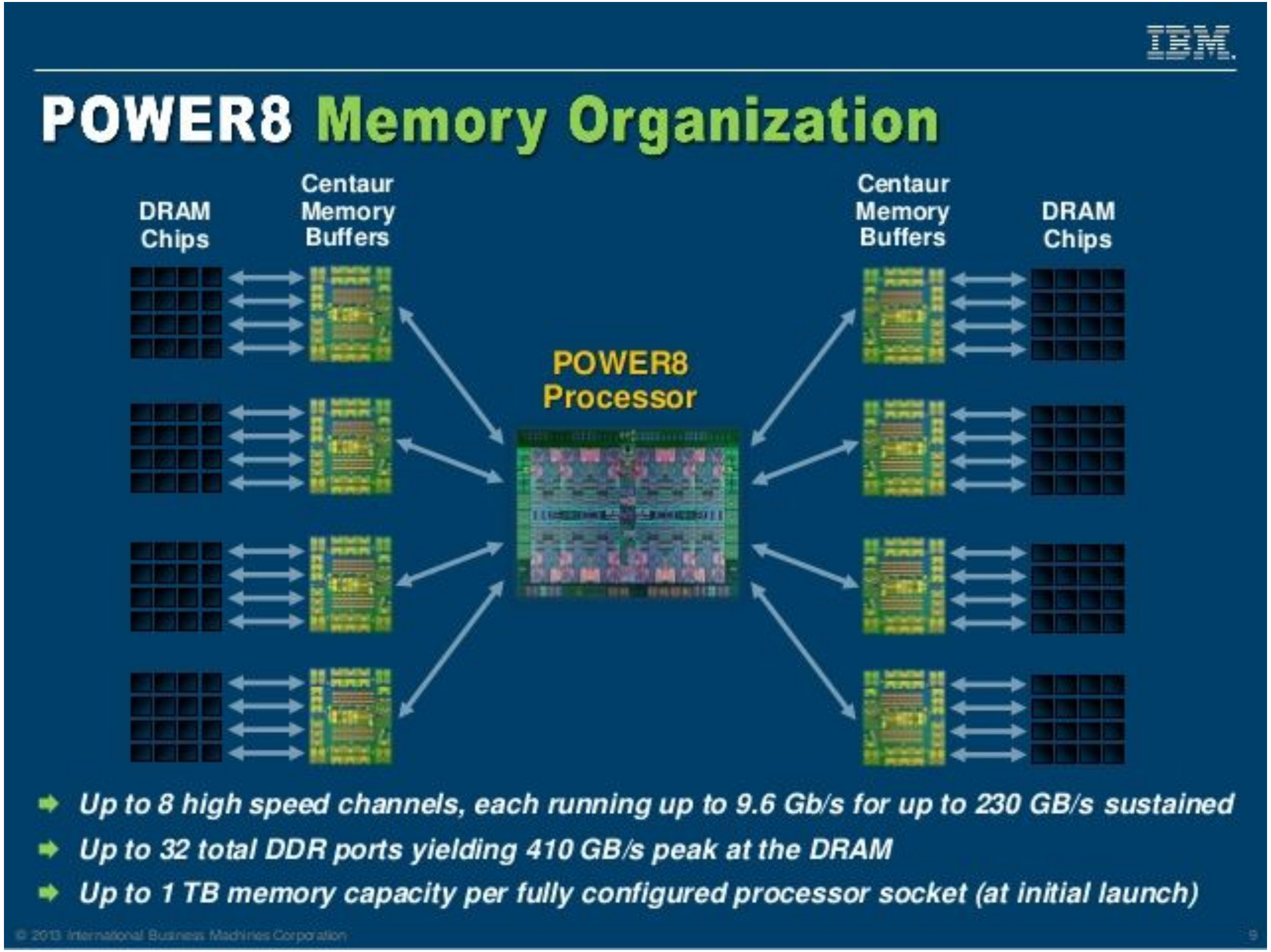
Memory

- Up to 230 GB/s sustained bandwidth

Bus Interfaces

- Durable open memory attach interface
- Integrated PCIe Gen3
- SMP Interconnect
- CAPI (Coherent Accelerator Processor Interface)

© 2013 International Business Machines Corporation



POWER8 Memory Organization

- Up to 8 high speed channels, each running up to 9.6 Gb/s for up to 230 GB/s sustained
- Up to 32 total DDR ports yielding 410 GB/s peak at the DRAM
- Up to 1 TB memory capacity per fully configured processor socket (at initial launch)

© 2013 International Business Machines Corporation

POWER9 core

New POWER9 Cores optimized for Analytics, Cloud and Big Data

- 24 SMT4 Cores per Chip

Two Socket Support

Direct Drive DDR4 Memory

- 8 DDR4 Channels
- 1866-2666 MHz DIMM Support

New Core Microarchitecture

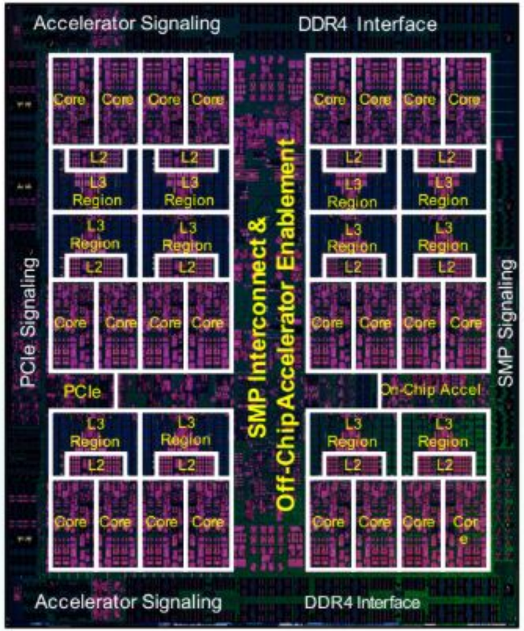
- Stronger thread performance
- Efficient agile pipeline
- POWER ISA v3.0

Enhanced Cache Hierarchy

- 120MB NUCA L3 architecture
- 12 x 20-way associative regions
- Advanced replacement policies
- Fed by 7 TB/s on-chip bandwidth

Cloud + Virtualization Innovation

- Quality of service assists
- New interrupt architecture
- Workload optimized frequency
- Hardware enforced trusted execution



14nm finFET Semiconductor Process

- Improved device performance and reduced energy
- 17 layer metal stack and eDRAM
- 8.0 billion transistors

Leadership Hardware Acceleration Platform

- Enhanced on-chip acceleration
- Nvidia NVLink 2.0: High bandwidth and advanced new features (25G Link)
- CAPI 2.0: Coherent accelerator and storage attach (PCIe G4)
- New CAPI: Improved latency and bandwidth, open interface (25G Link)

State of the Art I/O Subsystem

- PCIe Gen4 – 48 lanes

High Bandwidth Signaling Technology

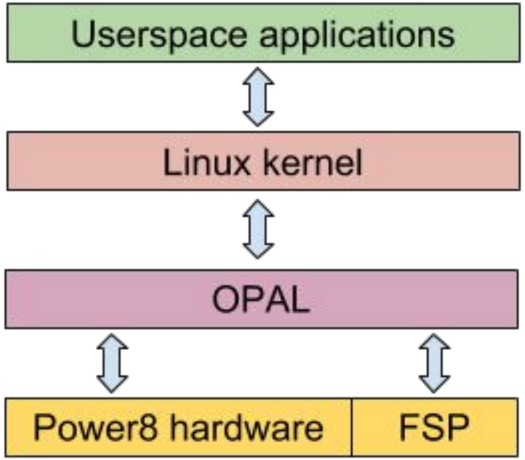
- 16 Gb/s interface
  - – Local SMP
- 25 Gb/s Link interface
  - – Accelerator

Hardware

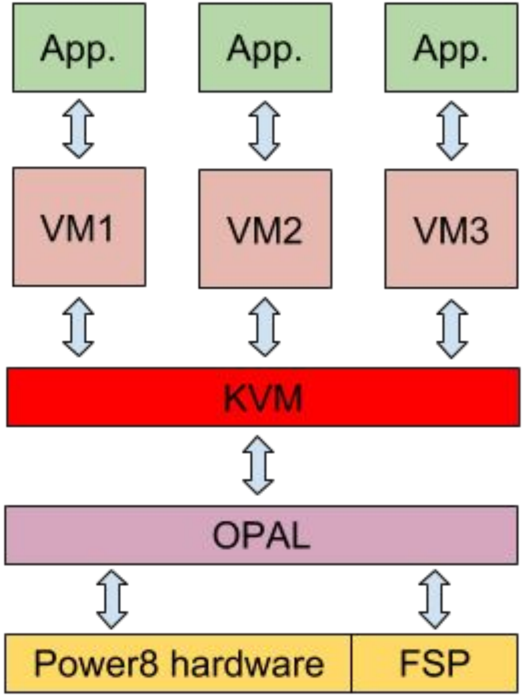
S821LC system:

- dual socket
- 128 cores (2 x 8CPUs x 8SMT)
- 128GB RAM
- 960GB Intel NVMe SSD
- 2x25G Chelsio NIC

PowerKVM and PowerNV software stack



PowerNV



PowerKVM

PowerKVM and PowerNV software stack

Flexible Service Processor (FSP)

- remote console
- server health and management

Open Process Automation Library (OPAL)

- Hypervisor
- Abstraction for:
  - interrupt management
  - PCIe configuration
  - system console
  - reset, power cycle
  - IOMMU set up

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - **S821LC**
- Power8 system internals
  - ABI and TOC
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- Current state and future work
- Performance measurements
- Q&A

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - S821LC
- Power8 system internals
  - **ABI and TOC**
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- Current state and future work
- Performance measurements
- Q&A

ABI and TOC - registers

R0	volatile	Used in function prologs.
R1	dedicated	Stack pointer
R2	dedicated	TOC pointer
R3-R12	volatile	Function parameters / scratch registers
R13	reserved	
R14-R31	non-volatile	Must be preserved across function calls
LR	dedicated	Link register
CTR	dedicated	Loop counter / 64-bit register for branches

ABI and TOC

TOC - table of contents:

- usually, each C-file has its own TOC table,
- a dictionary for all symbols used inside a file,
- contains VA of function and new TOC pointer.

```
.toc_base_XX:
...
printf:
    0x134520 // VA of .printf
    0x561230 // new TOC for .printf
...
```

```
.printf:  /* VA = 0x134520 */
    mfspr   r0, lr
    std     r31, r1, 0xfff8
    std     r0, r1, 0x10
    stdu    r1, r1, 0xff70
    or      r31, r1, r1
    std     r4, r31, 0xc8
    ...
```

ABI and TOC - function call

```
.toc_base_XX:
...
printf:      // at offset TB+0x160
     0x134520 // VA of .printf
     0x561230 // new TOC for .printf
...
```

```
// in C:  printf(...)

// in Assembly:
std r2, 40(r1)     // save current TOC
ld r8, 0x160(r2)   // load VA of .printf
ld r2, 0x168(r2)   // new TOC for .printf
mtctr r8           // move VA to CTR
blctr              // jump to CTR
ld  r2, 40(r1)     // restore TOC
```

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - S821LC
- Power8 system internals
  - ABI and TOC
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- Current state and future work
- Performance measurements
- Q&A

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - S821LC
- Power8 system internals
  - ABI and TOC
- Porting
  - **Initial FreeBSD state**
  - Bugs, bugs, bugs...
- Current state and future work
- Performance measurements
- Q&A

Porting - initial FreeBSD state

In-kernel support:

- generic ppc64 support in the kernel
- PMAP for Power architecture (AIM)

PowerNV project branch:

- console output on hardware
- non-working PCI driver
- boot to multiuser in SMP on Qemu
- boot to multiuser in SMP on hardware with embedded rootfs

Porting - what was missing

Missing features:

- PCIe driver needs to be validated on hardware,
- bootstrap must be aware of endianness change between loader and kernel.

What actually was done:

- IOMMU support for PCIe,
- tons of stability fixes,
- eliminated race conditions in SMP code,
- endianness robustness (loader, NVMe, bootstrap),
- performance optimization.

Presentation plan

- Hardware platform
    - Power8 and PowerNV
    - S821LC
- Power8 system internals
    - ABI and TOC
- Porting
    - Initial FreeBSD state
    - **Bugs, bugs, bugs...**
- Current state and future work
- Performance measurements
- Q&A

Bugs, bugs, bugs...

Few examples of issues we were dealing with:

- TOC in assembly routines (context switch),
- endianness in drivers (cxgbe, NVMe),
- edge-triggered IRQ and why they are dangerous,
- poor performance in SMT group.

Bug: TOC troubles in context switch

Observation:

- FreeBSD scheduler panicked in sched_switch with assert

```
MPASS(td->td_lock == TDQ_LOCKPTR(tdq));
```

- Depending on build, reproduction rate was either 100% or 0%
- Adding printfs (or comments?) “fixed” the issue

Bug: TOC change in context switch

sched_switch (fragment):

```
        ...
        cpu_switch(td, newtd, mtx);
        cpuid = PCPU_GET(cpuid);
        tdq = TDQ_CPU(cpuid);
        ...
        MPASS(td->td_lock == TDQ_LOCKPTR(tdq));
        ...
```

```
#define TDQ_CPU(x)
(&tdq_cpu[(x)])
```

```
.toc_base:
<other toc entries>
.tdq_cpu:  // tdq_cpu = toc_base + 1134
       0x11223300 // VA of tdq_cpu
<other toc entries>
```

```
TDQ_CPU:
// ABI: r2 == toc_base
ld  r3, 1134(r2)
// now r3 contains a pointer to tdq_cpu[0]
```

Bug: TOC change in context switch

```
sched_switch (fragment):

               …
               // r2 = TOC for SCHED_SWITCH
               // update r2 with TOC for CPU_SWITCH prior the call
               cpu_switch(td, newtd, mtx);  // NOTE: cpu_switch modifies stack pointer
               // load previous TOC from the stack
               // ERROR: here, r2 == TOC for cpu_switch
               cpuid = PCPU_GET(cpuid);
               tdq = TDQ_CPU(cpuid);
               ...
               MPASS(td->td_lock == TDQ_LOCKPTR(tdq));
               ...
```

Bug: endianness in NVMe and cxgbe(4)

Problem:

- Not many drivers are designed to work in BE environment
- NVMe: intensive usage of bitfields

```
union cc_register {
        uint32_t       raw;
        struct {
                uint32_t en           : 1;
                uint32_t reserved1    : 3;
                uint32_t css          : 3;
                uint32_t mps          : 4;
                (...)
        } bits __packed;
} __packed;
```

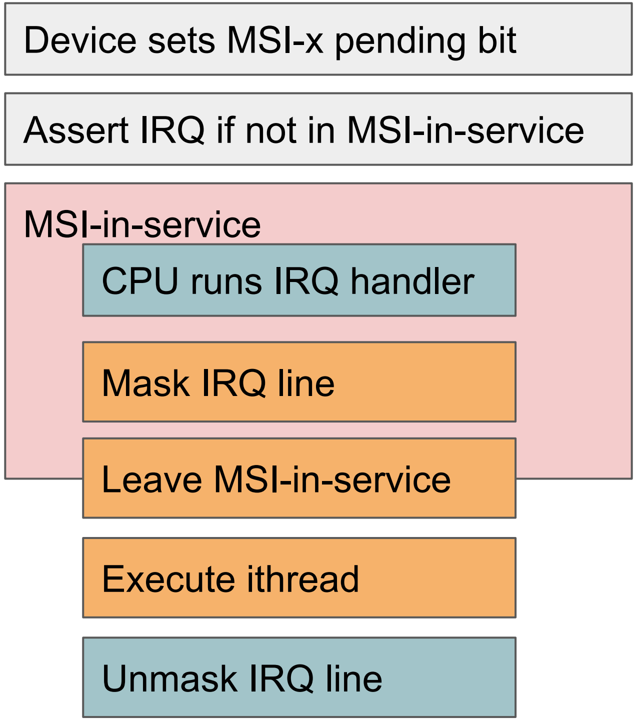
- CXGBE: few nits with endianness parsing
- NVMe: +1000LOC to add BE support

Bug: OPAL and edge-triggered IRQs

Problem:

- After few hundreds seconds running iperf3 over cxgbe interface, the traffic stops and TX queue of the NIC becomes unresponsive.

Bug: OPAL and edge-triggered IRQs



Device sets MSI-x pending bit

Assert IRQ if not in MSI-in-service

MSI-in-service

CPU runs IRQ handler

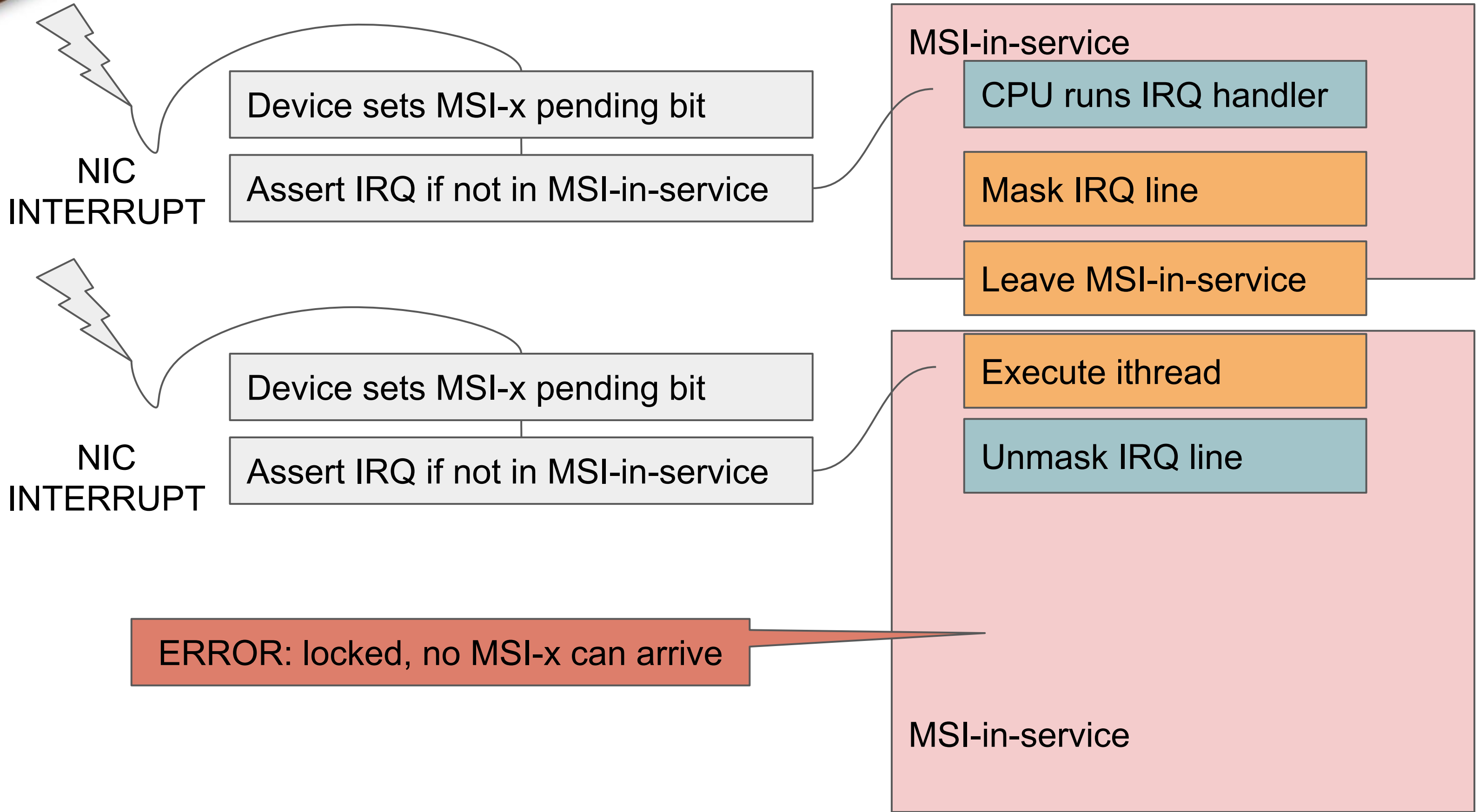
Mask IRQ line

Leave MSI-in-service

Execute ithread

Unmask IRQ line

Bug: OPAL and edge-triggered IRQs



NIC INTERRUPT

Device sets MSI-x pending bit

Assert IRQ if not in MSI-in-service

MSI-in-service

CPU runs IRQ handler

Mask IRQ line

Leave MSI-in-service

NIC INTERRUPT

Device sets MSI-x pending bit

Assert IRQ if not in MSI-in-service

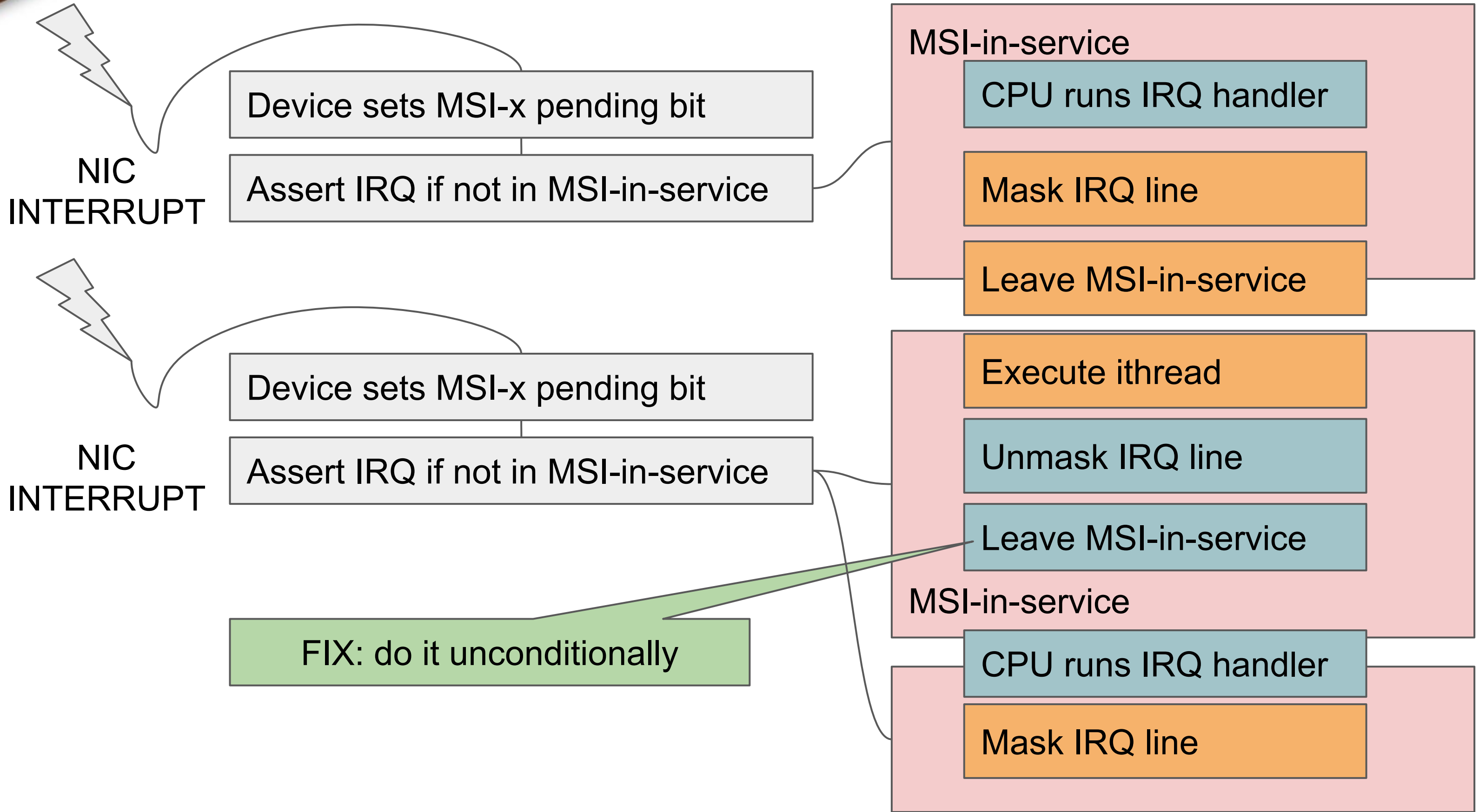
Execute ithread

Unmask IRQ line

ERROR: locked, no MSI-x can arrive

MSI-in-service

Bug: OPAL and edge-triggered IRQs



NIC INTERRUPT

Device sets MSI-x pending bit

Assert IRQ if not in MSI-in-service

MSI-in-service

CPU runs IRQ handler

Mask IRQ line

Leave MSI-in-service

NIC INTERRUPT

Device sets MSI-x pending bit

Assert IRQ if not in MSI-in-service

Execute ithread

Unmask IRQ line

Leave MSI-in-service

MSI-in-service

CPU runs IRQ handler

Mask IRQ line

FIX: do it unconditionally

Bug: poor performance

Problem:

- In a following test

```
~# iperf3 -s > /dev/null &
~# iperf3 -c 127.0.0.1 -P2
```

the system got only 600Mb/s of a total throughput, while Linux shows 70Gb/s.

Bug: poor performance

Debugging:

- Problem was narrowed down to be a generic issue with instruction execution speed. Simple test was created (time of 4G iterations was measured):

```
mtspr   ctr, r3
loop:
bdnz+   loop
blr
```

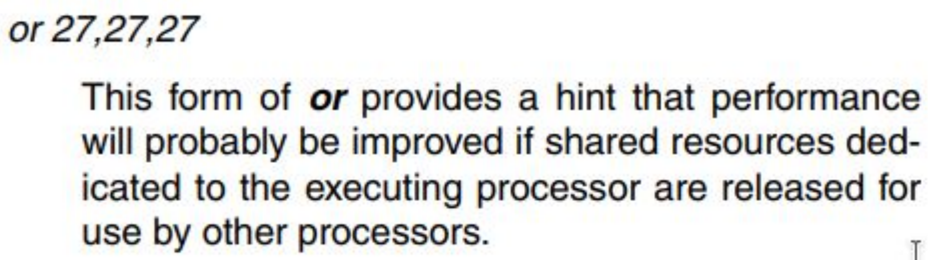
- Results:
  - Linux UP: 12.5s
  - Linux SMP: 5.5s
  - FreeBSD UP: 12.5s
  - FreeBSD SMP: 45s

Bug: poor performance

Idle thread on FreeBSD does:

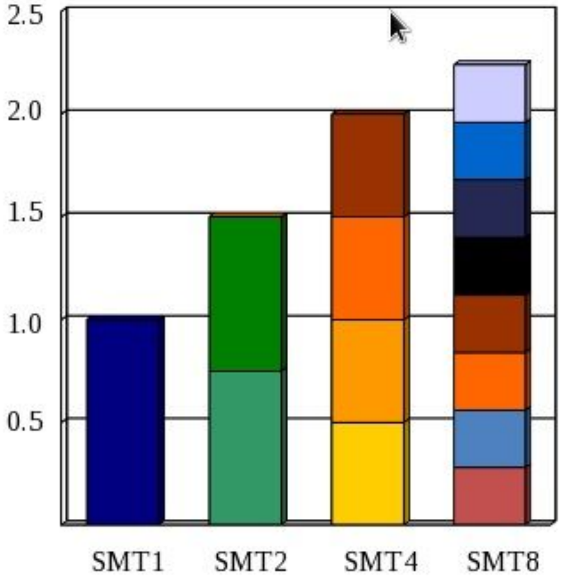
```
#define cpu_spinwait()        __asm __volatile("or 27,27,27") /* yield */
```

Documentation says:



or 27,27,27

This form of ***or*** provides a hint that performance will probably be improved if shared resources dedicated to the executing processor are released for use by other processors.



IBM: *“btw, this opcode is not implemented”*
not mentioned in any erratas...

Bug: poor performance

```
static void
powernv_cpu_idle(sbintime_t sbt)
{

        if (sched_runnable())
                return;

        spinlock_enter();

        // Typical architectures use wait-for-interrupt
        // wfi();
        enter_power_save();
        spinlock_exit();
}
```

```
CNAME(rstcode):
        /*
         * Check if this is software reset or
         * processor is waking up from power saving mode
         * It is software reset when 46:47 = 0b00
         */
        mfsrr1  %r9                             /* Load SRR1 into r1 */
        andis.  %r9,%r9,0x3                     /* Logic AND with 0x30000 */
        beq     2f                              /* Branch if software reset */
        bnel    1f
        .llong  cpu_wakeup_handler

        /* It is software reset */
…
```

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - S821LC
- Power8 system internals
  - ABI and TOC
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- **Current state and future work**
- Performance measurements
- Q&A

Current state and future work

Supported features :

- PowerNV on Power8 in Big Endian mode,
- OPAL integration
  - console,
  - interrupts,
  - IOMMU configuration.
- PCIe bus with following devices:
  - XHCI
  - NVMe
  - Chelsio cxgbe(4) compatible NIC
- Power management
  - reset, on, off
  - core deep sleep

Current state and future work

Missing pieces:

- Support for other drivers in Big Endian mode
  - AHCI
  - Intel NIC
- Fix dtrace
- Optimize libc to utilize SIMD instructions

Roadmap:

- Provide support for Power9 with:
  - Little Endian,
  - XIVE interrupt controller
  - Radix page tables MMU

Presentation plan

- Hardware platform
  - Power8 and PowerNV
  - S821LC
- Power8 system internals
  - ABI and TOC
- Porting
  - Initial FreeBSD state
  - Bugs, bugs, bugs...
- Current state and future work
- **Performance measurements**
- Q&A

Performance - NGINX

Test setup:

- Power8 or 8-core Intel CPU running FreeBSD (DUT),
- Intel PC connected over 10Gb link with DUT,
- stock NGINX serving 200b file over HTTP,
- WRK tool being run on Intel PC.

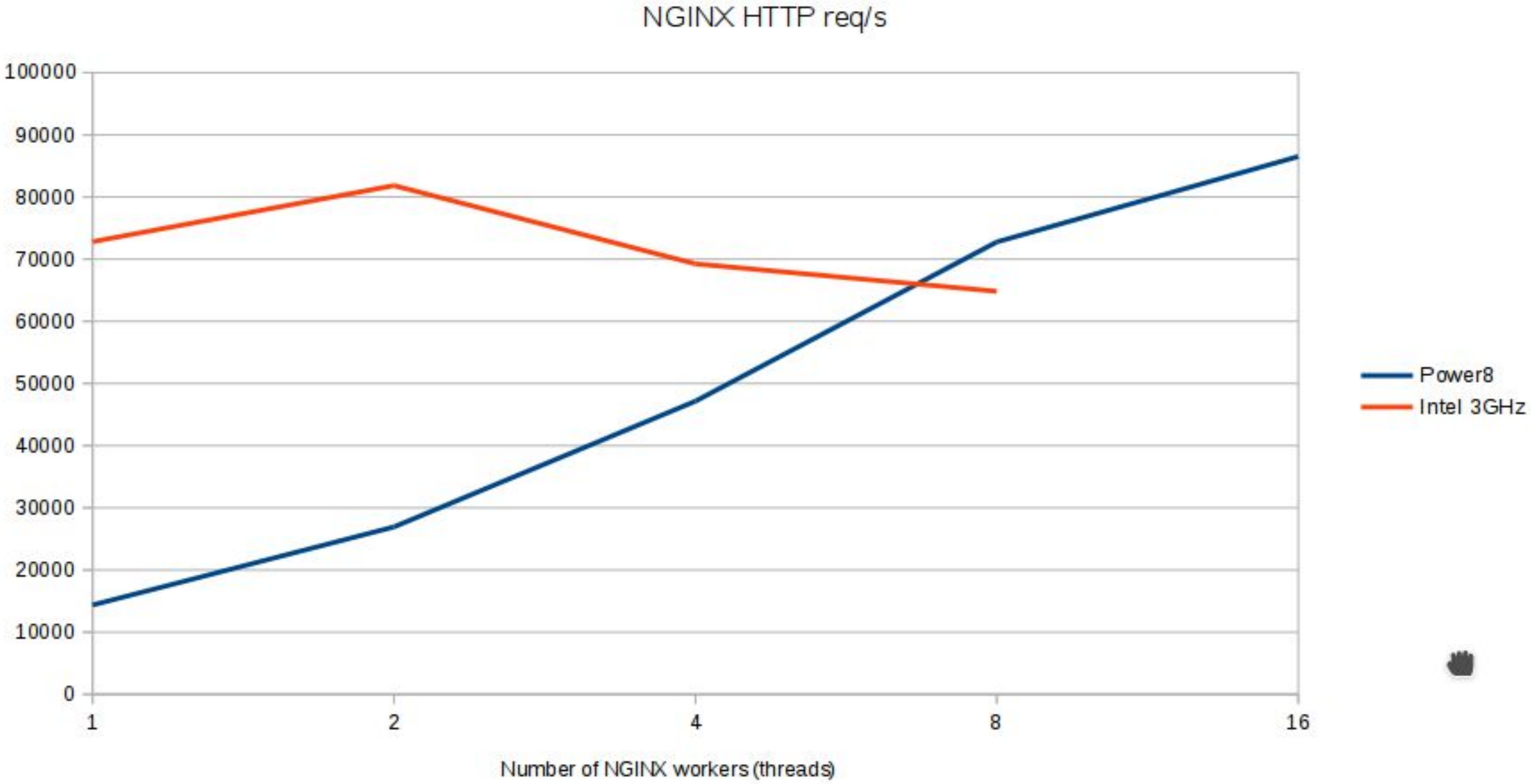
Test:

- Run following command for 1/2/4/8/16 NGINX worker threads:

```
wrk -t1 -c100 -d30s http://192.168.1.10/index.html
```

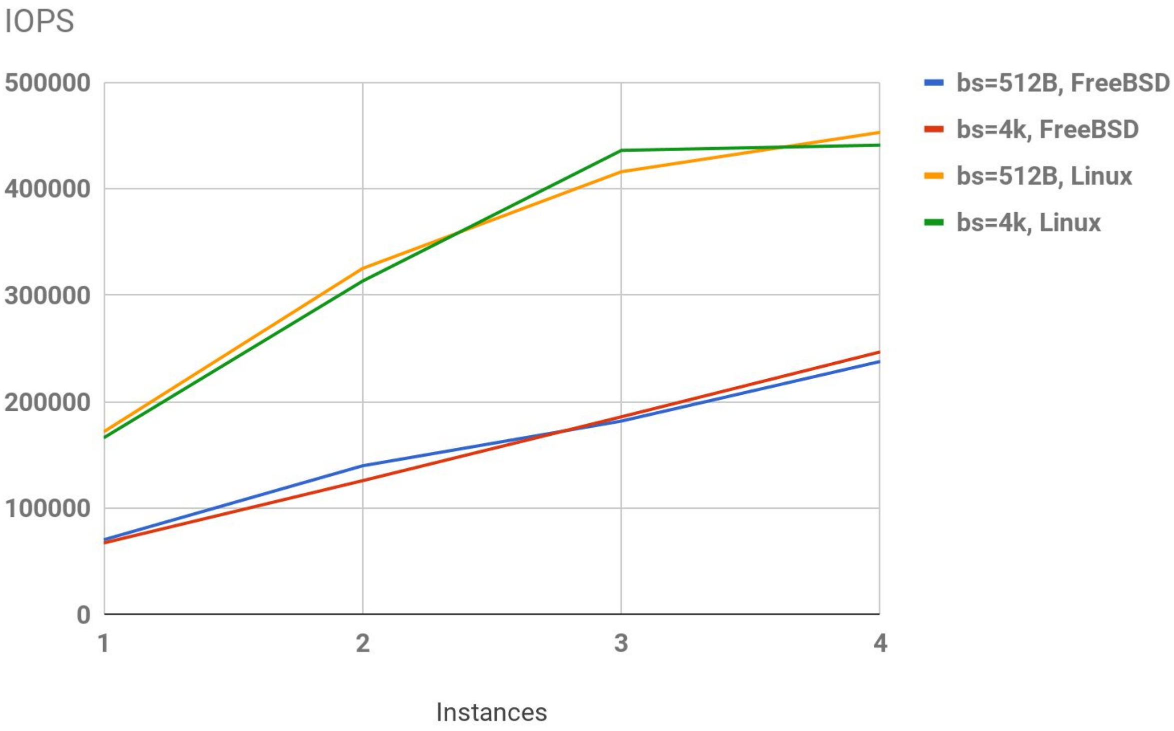
Software Meets Hardware at Semihalf

Performance - NGINX



Software Meets Hardware at Semihalf

NVMe IOPS



Acknowledgements

Special thanks go to:

- Nathan Whitehorn for initial work done for PowerNV and all help,
- Kevin Bowling (Limelight Networks) for organizing this project,
- Sam Montoya (QCM Technologies) for providing Power8 hardware.

Questions?