x86 : Multi Core

Gabriel Laskar <gabriel@lse.epita.fr>

Multi Core

- bsp/ap initialization
- mptables, madt
- idt, ipi, lapic, ioapic
- impact on kernel code
- Kernel Lock
- cache coherency

x86_64 Initialization

- Disable paging
- Set the PAE enable bit in `%cr4`
- Load `%cr3` with the physical address of the PML4
- Enable long mode by setting the `EFER.LME` flag in MSR `0xC0000080`
- Enable paging

x86_64 : Are we done yet ?

- We are still in compatibility mode, with 32-bit code
  - reload segment selector for %cs with
    - DB = 0
    - L = 1
- Now we can relocate all other tables (idt, gdt, tss...)

Interrupt Routing

- If I have multiple core, to which core the interrupt are delivered ?
- We need a new mechanism that enable customisation for interrupt routing

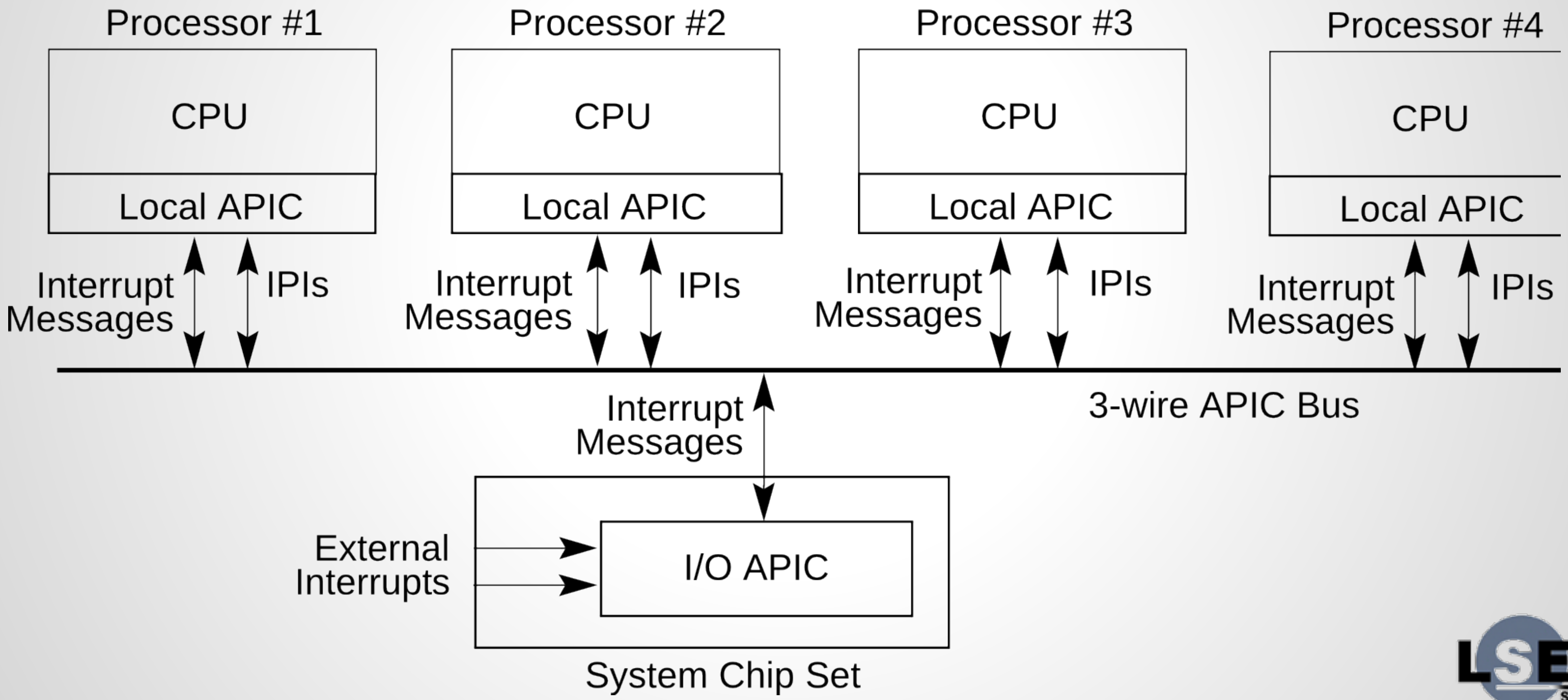
LAPIC

- memory mapped (starting at `0xfee00000`)
- Receive interrupts from multiple sources
  - Locally connected I/O devices (Local & External)
  - Inter-processor interrupts (IPIs)
  - APIC timer, PMC, Thermal, internal errors

IOAPIC

- 83093AA
- at least 24 programmable interrupts
- memory mapped
- more flexible on priorities
- usually connected to the LAPICs

IRQ Routing


Processor #1
CPU
Local APIC
Interrupt Messages
IPIs
Processor #2
CPU
Local APIC
Interrupt Messages
IPIs
Processor #3
CPU
Local APIC
Interrupt Messages
IPIs
Processor #4
CPU
Local APIC
Interrupt Messages
IPIs
3-wire APIC Bus
Interrupt Messages
External Interrupts
I/O APIC
System Chip Set

Talking to another core : IPI

- In the LAPIC
- can send unicast or broadcast requests
- Used for :
    - flushing TLBs
    - flushing Caches
    - power up or down another core
    - arbitrary messages

Caching

- Caches are either shared (L2)
- or specific for a core (L1)
- Synchronisation must be done at the hardware level

Discover Multiple cores

- How many cores do I have ?
- Where is located my APICs ?
- How the interrupt are configured ?

Multiprocessor Specification

- Old deprecated interface
- Easy to use
- But first we must find it !

Where are my MP tables

- Find the MP Floating Pointer Structure
  - In the first kilobyte of the EBDA
  - In the first kilobyte of system base memory (639k → 640k, or 511k → 512k)
  - In the BIOS ROM address space `0xf0000` and `0xfffff`
- Search for the Magic Value "_MP_"

What’s in it ?

- Processor
- Bus (PCI, ISA, VESA, etc...)
- I/O APIC configurations
- I/O Interrupts assignment
- Local Interrupts assignment

ACPI

- provides an open standard for device configuration and power management
- Replace
    - Advanced Power Management
    - MultiProcessor Specification
    - Plug and Play BIOS Specification

ACPI Tables

- Root System Description Pointer (RSDP)
- System Description Table Header
- Root System Description Table (RSDT)
- Fixed ACPI Description Table (FADT)
- Differentiated System Description Table (DSDT)
- Multiple APIC Description Table (MADT)
- Extended System Description Table (XSDT)
- ...

Root System Description Pointer

- Contains address of RSDT and XSDT
- Still in placed at random point in memory
- Magic "RSD PTR "

Root System Description Table

- Header with information about vendor
- Contain addresses to other tables
- XSDT is the same table but with 64-bit addresses

Fixed ACPI Description Table

- Define ACPI information vital to an ACPI-compatible OS
- Registers
- Pointer to DSDT
- Contains also various information (how to enable or disable ACPI)

Differentiated System Description Table

- Contains AML Code blocks
- AML is a generic bytecode
- Describe Hardware configuration
- Contains calls for Power Management states

Multiple APIC Description Table

- APIC structures
- Processor descriptions

Multi Core initialization

- Parse the MP tables to find the other APICs.
- initializes the bootstrap processor's local APIC.
- send Startup IPIs to each other cores with the address of trampoline code.
- trampoline code initializes the AP's to protected mode
- The BSP can initialize the IO APIC into Symmetric IO mode, to allow the AP's to begin to handle interrupts.
- The OS continues further initialization, using locking primitives as necessary.

Changes in the OS

- kind of like multi-threaded application
- We need to care about locking
- And never stop the other cores

Per-cpu context

- Per-cpu context
  - Most of the control structures are per-cpu
  - Some can be shared, for example GDT
- Per-cpu variables
  - we can use %gs or %fs to implement per-cpu pages.

Changes in the OS

- Locking strategies
  - Giant Lock (Big Kernel Lock)
  - Fine grained lock
- Algorithms
  - Scheduling
  - Memory allocation
  - Handling of kernel resources

Exercises

- write a kernel that dumps the mptables
- write a kernel that dumps the relevant ACPI tables