ハイパーバイザの作り方〜ちゃんと理解する仮想化技術〜 付属資料 最近のPCアーキテクチャにおける割り込みルーティングの仕組み

Takuya ASADA syuu@dokukino.com

平成26年4月23日

1 はじめに

Linuxにおける/proc/irq/¡IRQ¿/smp_affinityはハードウェアにどのような設定を行うことにより実現されているのか、或いは最近のPCアーキテクチャにおける割り込みの仕組みはどうなっているのか、という辺りが知りたかったので調べてみた。

結構こんがらがっているので、予想外に時間を食ってしまった…まだ調べ尽くせていないが、一旦現時点での理解を書いておこうと思う。

2 前提条件

1. 2つ以上のCPUコアを持つ、Core2世代或いはCore iシリーズ世代のIntel CPU／チップセット
2. 割り込みを行う主体はPCIeデバイスである（主にNICを想定しているが、これに限定されない）
3. Legacyな8259割り込みコントローラを使うことは考慮しない
4. x86_64向けLinuxカーネル（解析に使っているバージョンは3.2.0+）が動作している
5. 仮想化は使用しない

3　PCIeに於ける割り込みの種類

3.1　レガシー割り込み（INTx）

PCI規格に最初から用意されていた割り込み方法で、大半のPCIデバイスはこの割り込みを用いている。PCIバスのメインラインとは別に用意された割り込み用の物理的なピンを用いて割り込みを通知する。

PCIeには割り込み用ピンは用意されておらず、帯域内メッセージを用いるレガシー割り込みエミュレーションによってソフト的な互換性を維持しているものの、基本的にはMSI／MSI-X割り込みへ移行することが推奨されているものと思われる。

3.2　MSI割り込み

PCI 2.3から追加された割り込みモードでピンを使用しない帯域内メッセージで割り込みを行う。デバイスあたり最大32個のMSIメッセージをサポートしている。

3.3　MSI-X割り込み

PCI 3.0ではオプションとされPCIe 1.0から必須とされた割り込みモードで、MSI割り込みの拡張版。デバイスあたり最大2048個のメッセージをサポートしている。

4　割り込みルーティング

4.1　レガシー割り込み

デバイスからピン経由で割り込みを通知→IOAPICでRedirection Table Entryを参照、通知先LAPICを決定→CPU内のLAPICへ割り込みを通知

4.2　MSI割り込み

デバイスはPCI Configuration SpaceのCapability Structure内のMSIフィールドを参照、MSI AddressレジスタとMSI Dataレジスタの値から通知先LAPICとLAPIC上のベクタ番号を決定→CPU内のLAPICへ割り込みを通知

4.3 MSI-X 割り込み

レジスタの構成が異なる（ベクタ毎に Address と Data が用意されている）が基本的な仕組みは MSI 割り込みと同様

4.4 Capability Structure

[1] を見るとイメージが分かると思うが、Configuration Space から Linked List 状に複数の capability が繋がる構造になっていて、CAPID が 0xd0 なのが MSI のフィールドで、ここには MSICTL, MSIAR, MSIDR の 3 つのレジスタがある。

4.5 MSI Control Register(MSICTL)

どの CPU に割り込むかを考える上では重要ではないので省略

4.6 MSI Address Register(MSIAR)

- 31:20 = 0xfee
- 19:12 = Destination ID
- 11:4 = IA32 では未使用
- 3 = Address Redirection Hint(RH)
  - 0: Directed
  - 1: Redirectable
- 2 = Address Destination Mode(DM)
  - 0: Physical Mode
  - 1: Logical Mode
- 1:0 = 予約

Destination Mode が Logical かつ Redirection Hint が Redirectable な場合は Destination ID でビットが立っている CPU の中で Task Priority Register(TPR) が最も低い CPU の LAPIC へ割り込みが送られる。それ以外の RH, DM の組み合わせでは Destination ID で指定されているビットの中で特定の CPU の LAPIC へ割り込みが送られる。

Physical Mode で Destination ID が 0xff の場合はブロードキャスト割り込みを行う。

4.7 MSI Data Register(MSIDR)

- 31:16 = 0x0000
- 15 = Trigger mode
  - 0: Edge
  - 1: Level
- 14 = Delivery status
  - 0: Deassert
  - 1: Assert
- 13:12 = 0x00
- 11:8 = Delivery mode
  - 0000: Fixed
  - 0001: Lowest priority
  - 0010: SMI/PMI/MCA
  - 0011: Reserved
  - 0100: NMI
  - 0101: INIT
  - 0110: Reserved
  - 0111: ExtINT
  - 1000-1111: Reserved
- 7:0 = Interrupt Vector

Delivery mode が Fixed の場合は Destination に指定された全ての CPU へ割り込みを行う。Lowest Priority の場合は Task Priority Register の値が最も低い CPU へ割り込みを行う。Interrupt Vector に割り込み先 LAPIC の Vector 番号を指定。

4.8 Linux カーネルで実際にレジスタの値を設定している所を見てみる

msi_compose_msg[1] でレジスタに書き込みたい値を用意しているので、これを見てみる。$msg->address\_lo$ が MSIAR レジスタで、$apic->irq\_dest\_mode$ が 0

[1] http://lxr.linux.no/linux+v3.2/arch/x86/kernel/apic/io_apic.c#L3167

なら physical mode、1 なら logical mode を設定、$apic->irq\_delivery\_mode$ が dest_LowestPrio なら Redirectable（MSI_ADDR_REDIRECTION_LOWPRI）を、そうでなければ Directed（MSI_ADDR_REDIRECTION_CPU）を設定、変数 dest を Destination ID として設定している。

$msg->data$ が MSIDR レジスタで、$apic->irq\_delivery\_mode$ が dest_LowestPrio なら Lowest priority を、そうでなければ Fixed を設定、$cfg->vector$ の値を Interrupt Vector として設定している。

$apic->irq\_dest\_mode$ と $apic->irq\_delivery\_mode$ の値は IO APIC のドライバ毎に違うのだが、x86_64 の標準ドライバの apic_flat_64.c[2] では irq_dest_mode は 1, irq_delivery_mode は dest_LowestPrio に設定されている。

これらの値は割り込み初期化時に設定され、/proc/irq/¡IRQ¿/smp_affinity の書き換え時にも維持される。smp_affinity の書き換え時には、Destination ID と Interrupt Vector だけが変更される [3]。

全ての環境で Logical mode かつ Lowest priority が使えるとは限らないので、場合によっては Physical Mode で初期化されていて smp_affinity の値を 0xff にしても CPU0 にしか割り込まないという挙動を行う事も有り得る。実際、論理 CPU が 12 個ある Core i7 上で Linux 3.2.0+を走らせている環境では Extended Physical Mode で初期化されていて、割り込み分散が行われていなかった。

参考文献

[1] BIOS が PCI Express を初期化する手順が見えてきた: なひたふ JTAG 日記 http://nahitafu.cocolog-nifty.com/nahitafu/2007/02/pci_express_2b63.html

[2] ntel 64 and IA-32 Architectures Software Developer Manuals http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

[3] ntel 5520/5500 Chipset: Datasheet http://www.intel.com/content/www/us/en/chipsets/5520-5500-chipset-ioh-datasheet.html

[4] CI Local Bus Specification Revision 3.0

[5] CI Express 2.0 Base Specification Revision 0.9

[2] http://lxr.linux.no/linux+v3.2/arch/x86/kernel/apic/apic_flat_64.c#L180
[3] http://lxr.linux.no/linux+v3.2/arch/x86/kernel/apic/io_apic.c#L3201