ハイパーバイザの作り方〜ちゃんと理解する仮想化技術〜 第５回 I/O 仮想化「割り込み編・その２」

はじめに

前回の記事では、割り込み仮想化の話の前提となる x86 アーキテクチャにおける割り込みのしくみを各コンポーネントごとに解説しました (図 1)。仮想環境でどのように割り込みを実現するのか、前回解説した各機能ごとに見ていきます。

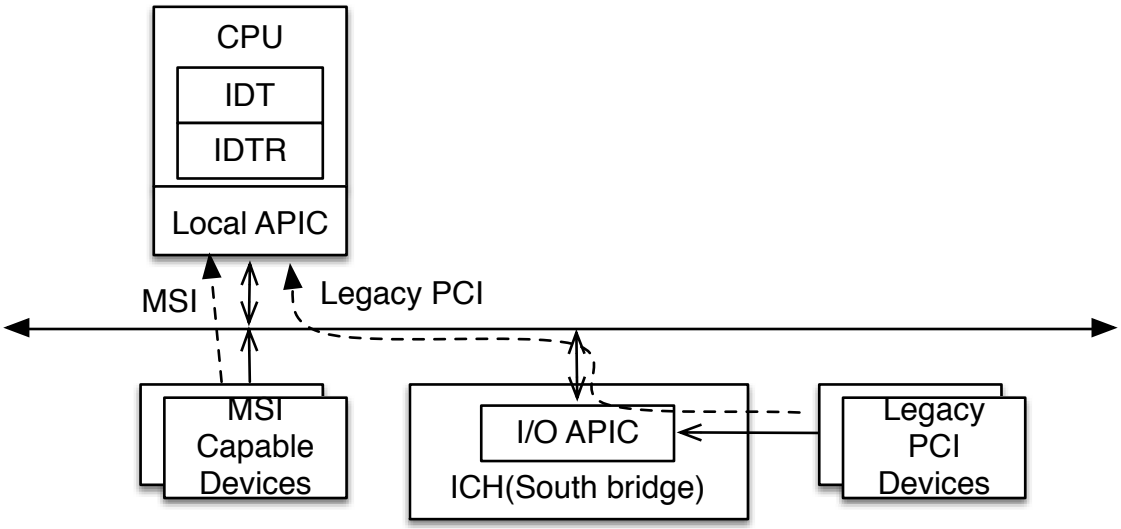

図 1　割り込みにかかわるコンポーネントと仮想化範囲

仮想化における内部割り込みと外部割り込み

VT-x 環境においては、内部割り込みは CPU がすべて処理を行うため、基本的にハイパーバイザが介入する必要がありません。一方、外部割り込みについては、ハイパーバイザの介入が必要となります。それぞれを見ていきましょう。

CPU への割り込みの挿入

仮想 CPU で任意の割り込みを発生させるに は、VMCS の VM-Entry Control Fields にある VM-entry interruption-information field にベクタ番号と割り込みタイプを書き込みます (表 1)。ただし、このフィールドに値をセットして外部割り込みを発生させるだけでは、Local APIC のレジスタ値は適切に更新されず、ハイパーバイザが新しい値を計算しセットする必要があります。

ビットポジション	内容
7:00	ベクタ番号
10:08	割り込みタイプ 通常は 0 (外部割り込み) を使用
11	スタックに例外の error code を push
31	有効化ビット

表 1: VM-entry interruption-information field

内部割り込みの仮想化

実機での内部割り込みは、次の手順で処理されます。

1. ゲストマシン上のソフトウェアが例外を発生させるか、INT 命令の実行により CPU で内部割り込みが発生
2. CPU は IDT 上のゲートデスクリプタを読み込み、割り込みハンドラを実行
3. 割り込みハンドラが内部割り込みを処理
4. IRET 命令で直前のコンテキストへ復帰

これらはすべて、ハイパーバイザの介入が不要です。2〜4 については、IDT/IDTR の仮想化にて説明します。ただし、ここでも VMCS の設定により、内部割り込みを契機として VMExit を発生させることができます[*1]。ただし、この利用方法は一般的ではありません。

[*1] VMCS の VM-Execution Control Fields の Exception Bitmap の各ビットが各例外のベクタ番号に対応していて、ここに 1 を設定するとその例外が発生した時に VMExit が発生するようになります。通常の例外は基本的に VMExit する必要がありませんが、連載第 2 回 (Intel VT-x の概要とメモリ仮想化) で解説したシャドーページングを行うには、ページフォルト例外で の VMExit が必須になります。

外部割り込みの仮想化

内部割り込みはソフトウェアを起因とし CPU 内部で発生するため、VT-x によりハイパーバイザの介入なしに仮想化することが可能でした。一方、外部割り込みは、ハイパーバイザが割り込みを送り込みます。これは、デバイスがソフトウェア的に、ハイパーバイザ内に実装されているためです。

I/O APIC を通して割り込む場合

実機での I/O APIC を通して割り込む場合の外部割り込みは、次のような手順で処理されます。

1. デバイスが割り込みラインから I/O APIC へ割り込みを送信
2. I/O APIC が割り込みを受け取り、RedirectionTable Entry に指定された Destination ID が示す Local APIC へ割り込みを転送
3. Local APIC が CPU へ割り込み
4. CPU は IDT 上のゲートデスクリプタをロードし、割り込みハンドラを実行
5. 割り込みハンドラが外部割り込みを処理
6. 割り込みハンドラが Local APIC へ EOI を書き込み、割り込み処理の終了を伝達
7. IRET 命令で直前のコンテキストへ復帰

このうち、4、5、7 については内部割り込みと同様の処理であり、ハイパーバイザの介入は必要ありません。1〜3 は次のように仮想化されます。まずあらかじめ、ゲスト OS が割り込みを初期化するときに I/O APIC の Redirection Table Entry へ宛先 Local APIC が設定されます。実際にデバイスが使われ始め、ハイパーバイザがデバイスのエミュレーションを行うと、割り込みを仮想 CPU へ送る必要が出てきます。

デバイスエミュレータからの割り込みを受け、ハイパーバイザはデバイスに対応する Redirection Table Entry の値を参照し、宛先の仮想 CPU を選びます。宛先の仮想 CPU が決定されたら、ハイパーバ イザは宛先 CPU の Local APIC の IRR レジスタを更 新し、VMCS に割り込みの挿入を設定します。割り込み挿入が設定された仮想 CPU が VMEnter されると、以降は内部割り込みと同様に、実機とほぼ同じ手順で割り込みの受付が行われて割り込みハンドラが起動されます。

6 の EOI 書き込みに関しては、Local APIC の EOI レジスタへのアクセスを、ハイパーバイザが介入してエミュレーションを行う必要があります。まとめると、外部割り込みを仮想化するには、ハイパーバイザで IO APIC・Local APIC のエミュレーションを行い、仮想 CPU へ割り込みを挿入する必要があります。

MSI/MSI-X 割り込みを用いて割り込む場合

実機では、次の手順で処理されます。

1. デバイスが PCI Configuration Space に指定された Destination ID が示す Local APIC へ割り込みを転送
2. Local APIC が CPU へ割り込み

3. CPU は IDT 上のゲートデスクリプタをロードし、割り込みハンドラを実行
4. 割り込みハンドラが外部割り込みを処理
5. 割り込みハンドラが Local APIC へ EOI を書き込み、割り込み処理の終了を伝達
6. IRET 命令で直前のコンテキストへ復帰

MSI/MSI-X 割り込みを用いる場合の違いは、割り込み先が I/O APIC の Redirection Table Entry に書いてあるのではなく、PCI Configuration Space に書いてある、という点だけです。これを仮想化する場合、ハイパーバイザで宛先の仮想 CPU を選択するときの参照先が変わりますが、あとは I/O APIC を通じた割り込みと同じです。

IDT/IDTR と割り込みハンドラ

IDT/IDTR や割り込みハンドラに関しては、とくにハイパーバイザが介入すべき処理はありません。このため VMExit は発生せず、すべて CPU が仮想化を行います[*2]。 VT-x において一部の汎用レジスタは VMX root mode/VMX non-root mode の切り替えときにコンテキストをハイパーバイザで保存する必要がありました。しかし、IDTR レジスタのコンテキスト保存/復帰については、ハイパーバイザは関与しません。

これは、CPU によって行われるためです。ゲストマシン上の IDT の作成や割り込みハンドラのアドレスの設定は、通常のメモリアクセアスと同様に行われます。また、VT-x non-root mode では、実機での動作と同様に割り込みや例外を受け付け、割り込みハンドラを実行する機能が備わっています。ゲストマシンの IDTR は、IDT の構築後に設定されます。な お、一般的な手法ではありませんが、VMCS の設定[*3]により IDTR への読み書きを契機として VMExit を発生させることもできます[*4]。

このうち、2〜4 でハイパーバイザの介入が不要であることは、すでに説明しました。1 ですが、ゲスト環境から内部を発生させ、割り込みハンドラを起動する処理の中で、とくにハイパーバイザの介入は必要ありません。

ただし、ここでも VMCS の設定により、内部割り込みを契機として VMExit を発生できます。この場合、内部割り込みをゲスト OS に渡さずハイパーバイザで横取りして処理をしたり、デバッグ機能としてゲスト環境上の内部割り込みの回数をカウントしたりといった機能を実装できます。しかしながら、そのような使い方は一般的ではありません。

まとめると、内部割り込みの一連の処理に関しては、特にハイパーバイザが介入すべき処理はありません。このため VMExit は発生せず、すべて CPU が仮想化を行います。

[*2] ただし、割り込みハンドラ内で IO ポートアクセスなどの操作 を行えば VMExit が発生する操作を行えば、そこでは VMExit が発生します。

[*3] VMCS の VM-Execution Control Fields の Secondary Processor-Based VM-Execution Controls にある Descriptor-table exiting にビットを立てることで、LGDT、LIDT、LLDT、 LTR、SGDT、SIDT、SLDT、STR の各命令を実行しようとした 時に VMExit するようになります。

[*4] この場合、ハイパーバイザは ID のシャドーイングを行ってゲ スト OS が意図する割り込みハンドラと異なる割り込みハンドラを設定できます。また、 IDTR へのアクセスをイベントとして受け取り、デバッグ機能を実装することもできます。

Local APIC の仮想化

Local APIC はメモリマップド I/O でアクセスするため、通常のメモリマップド I/O の仮想化手法が使えます。しかし、高速化のために VT-x には Local APIC へのアクセスを特別扱いしてハンドルする機能が実装されています。このことは、連載第 3 回 (“第 3 回 I/O 仮想化「デバイス I/O 編」”) で解説しました。つまり、図 1 で示したように、部分的に VT-x による仮想化支援を受けることができます。

しかしながら、この VT-x による仮想化支援機能はハイパーバイザが何もしなくても完全に CPU 側でレジスタ値の更新などを行ってくれるというものではありません。CPU へ外部割り込みを挿入する場合、ゲスト OS から見てつじつまが合わなくならないよう LocalAPIC のレジスタ値を同時に設定する作業はハイパーバイザから行う必要があります。具体的には、次の操作を行います。形になります。実機での外部割り込みは、次の手順で処理されます。

1. 割り込み発生時点で IRR レジスタに割り込むベクタ番号のビットをセット
2. 仮想 CPU が VMExit するのを待つ/または IPI などを用いて VMExit させる
3. IRR にセットされた最高優先度のビットをクリア、同じビットを ISR にセット
4. TPR と ISR の値に基いて PPR を更新
5. VM-entry interruption-information field に割り込みをセット
6. VMEnter して割り込みを発生させる

また、割り込みハンドラの終了を通知するためにゲスト OS が EOI レジスタへ書き込んできたときにも、ハイパーバイザの介入が必要です。具体的には、 次の作業を行います。

1. EOI への書き込みにより VMExit
2. IRR が 0 なら VMEnter してゲストへ復帰?IRR に値があれば IRR にセットされた最高優先度のビットをクリア、同じビットを ISR にセット
3. TPR と ISR の値に基いて PPR を更新
4. VM-entry interruption-information field に割り込みをセット
5. VMEnter して割り込みを発生させる

こちらも、前回の記事で解説した実機上の Local APIC の挙動と同じです。

I/O APIC の仮想化

I/O APIC もゲスト OS からメモリマップド I/O でアクセスされます。ただし、Local APIC と異なり高速化用の特別な VMExit などの仮想化支援機能はありません。使われ方としては、前述のとおりゲスト OS の初期化時に割り込み先 CPU の設定をメモリマップド I/O 経由で受け取り、仮想デバイスから割り込みを送るときの仮想 CPU 選択に設定された値を 用います。

MSI/MSI-X 割り込みの仮想化

MSI/MSI-X 割り込みの場合は、PCI Configuration Space への書き込みにより割り込み先 CPU の設定を受け取ります。書き込むデバイスやレジスタの フォーマットは違いますが、使い方はほぼ I/O APIC と変わりません (“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”)。

物理ハードウェアからの割り込みへの対処

VMX non-root mode の実行中に物理ハードウェアから割り込みが来た場合、ハイパーバイザでこれを受け取り割り込みハンドラを起動して処理する必要があります。このために、ハイパーバイザは VMCS の初期化時に VM-Execution Control Fields の Pin- Based VM-Execution Controls にある External- interrupt exiting にビットをセットします。これにより、ゲストマシンは外部割り込み発生時に VMExit するようになります。また、VM-Exit Control Fields にある VM-Exit Controls の Acknowledge interrupt on exit ビットを 1 に設定した場合、外部割り込みは VMExit 時に “acknowledged” になり、割り込みベクタ番号は VM-Exit information fields の VM-exit interruption information に保存されます。VMM はこのベクタ番号を参照して、割り込みハンドラを起動し割り込みを処理します。

一方、Acknowledge interrupt on exit ビットを 0 に設定した場合は割り込みは “acknowledge” されず、RFLAGS レジスタの IF ビットでマスクされている状態になります。このまま IF フラグをセットすれば、IDT に設定された割り込みハンドラが起動して割り込みを処理できます。通常の OS の上にハイパーバイザを実装する方式では、IDT による割り込みハンドルが行われているため、後者の方法を取る場合がほとんどです。

まとめ

いかがでしたでしょうか。今回は Intel VT-x における割り込みの仮想化方法を中心に解説してきました。次回はソフトウェア側の実装に移り、「VT-x を用いたハイパーバイザの実装方法の基礎」を中心に解説します。

ライセンス

参考文献

“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.” http://syuu1228.github.io/howto_implement_hyperv

“第 3 回 I/O 仮想化「デバイス I/O 編」.” http://syuu1228.github.io/howto_implement_hypervisor/part3.pdf.