Hewlett Packard Enterprise

連載
“まるごと仮想化”の
ここが「ツボ」

ここ 2、3 年で急速に IT 業界に広く浸透してきた仮想マシンベースの仮想化技術。本連載では、Integrity サーバー上で仮想化を実現するソフトウェア「Integrity VM」の最新バージョンが備える機能や能力を紹介し、そのパフォーマンスをフルに引き出し、“まるごと仮想化”するための「ツボ」を考えていきたい。

《連載期間：2009 年 5 月～2009 年 11 月》

―目次―

第 1 回

パフォーマンスを引き出す CPU リソースの配分設定

2009 年 5 月

Integrity サーバー上で仮想化を実現するソフトウェア「Integrity VM」がリリースされてから 4 年が経過した。ご存じのとおり、仮想マシンベースの仮想化技術はここ 2、3 年で急速に IT 業界に広く浸透しており、中小規模の実運用環境を仮想マシン上で構築する事例ももはや珍しくはない。“ミッションクリティカル環境向けの仮想マシン”である Integrity VM も、当初は開発環境やテスト環境を中心に導入されていたが、最新バージョンの登場により実運用環境での本格導入が可能になり、実際にも検討され始めている。そこで本稿では、最新バージョンの Integrity VM が備える機能や能力を改めて紹介し、実運用環境でも十分なパフォーマンスを引き出すための使いこなしのコツを紹介したい。

“まるごと仮想化”で行こう！

2009 年 4 月にリリースされた Integrity VM 4.1（以下、Integrity VM）では、以前からの HP-UX 11i v3 サポートや AVIO（高速化仮想 I/O）に加え、オンラインマイグレーション機能もサポートし、いよいよ本番機を含めた“まるごと仮想化”の機運が熟したといえる。加えて、クラスターウェアの Serviceguard 11.19 では、仮想マシン上で稼動するアプリケーション・プロセスの障害検知に対応した。そこで本稿では、最新バージョンの Integrity VM が備える機能や能力を改めて紹介し、実運用環境でも十分なパフォーマンスを引き出すための使いこなしのコツを紹介したい。

Integrity VM のパフォーマンスをフルに引き出すには、以下の 4 つのサーバーリソースについて、最適な設定を施す必要がある。

- CPU
- メモリ
- ディスク
- ネットワーク

これらのうち、本稿では CPU リソースの割り当て方法について紹介する。

CPU リソースの仮想化を理解する

まずは、Integrity VM における CPU リソースの仮想化について改めて説明しよう。

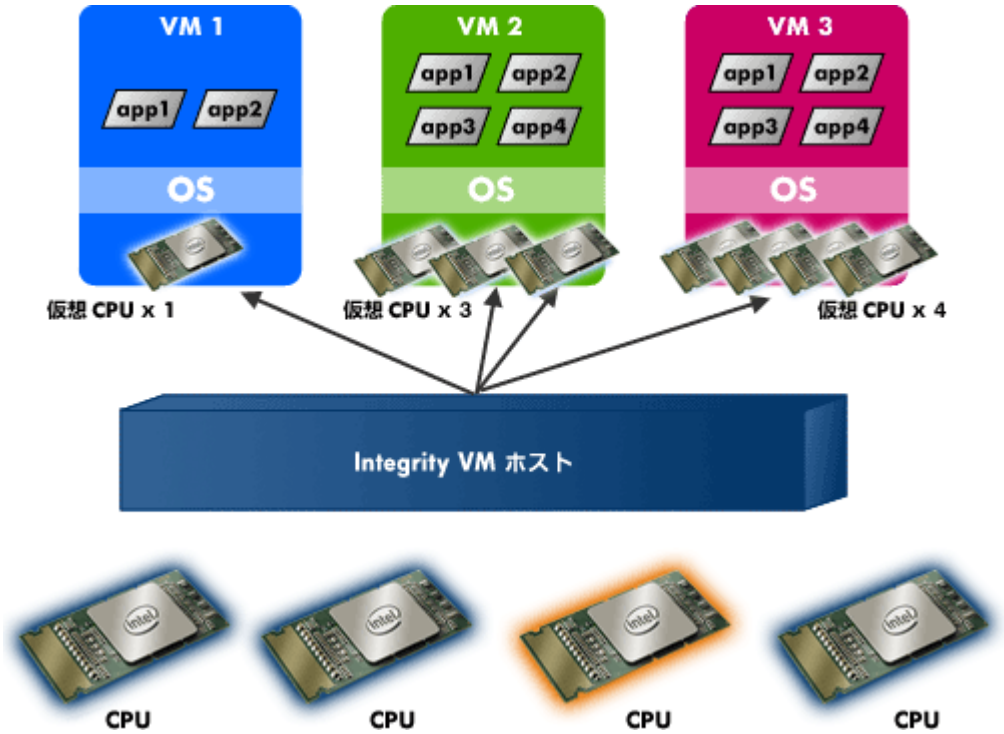

図 1：Integrity VM による CPU リソースの仮想化

図 1 は、Integrity VM における CPU リソースの仮想化を表したものである。この例では、4 個の CPU を搭載した Integrity サーバー上で、3 つの仮想マシンを運用している。ここで、それぞれの仮想マシンに配分されている「仮想 CPU」に注目していただきたい。仮想 CPU とは、「それぞれの仮想マシンが利用する CPU 数」を表している。よって図 1 の例では、左端の VM1 は 1 個、VM2 は 3 個、そして VM3 は 4 個の物理 CPU を用いて動作する。

ちなみに 1 つの仮想マシンには、サーバーの物理 CPU 数を超える仮想 CPU 数を割り当てることはできない。そのため、もし 2-way サーバーで Integrity VM を利用する場合は、各仮想マシンに配分できる仮想 CPU 数は最大 2 個までとなる。また多数の CPU を搭載するサーバーの場合は、仮想 CPU 数を最大 8 個まで割り当てることが可能だ。

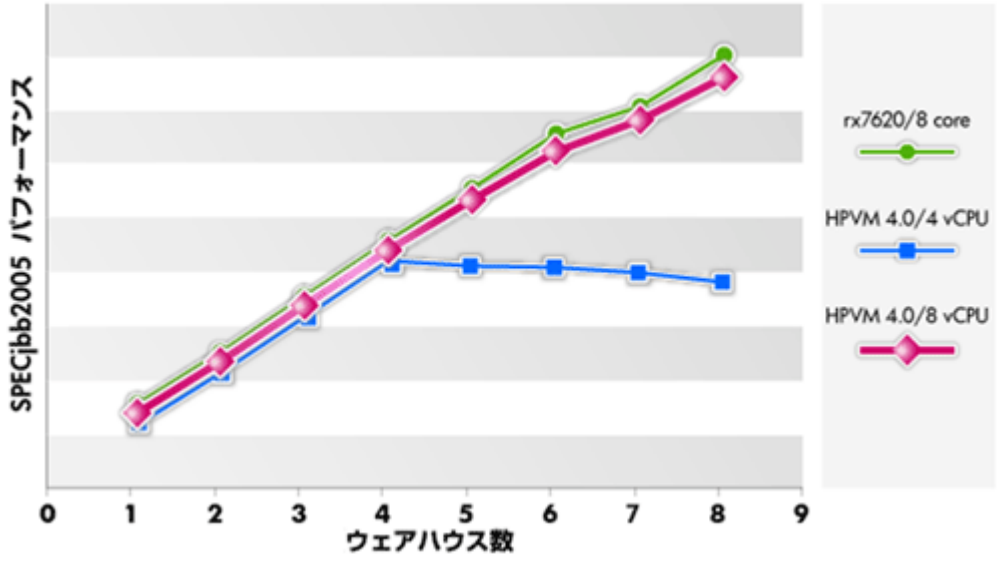

図 2：仮想 CPU 数の増加によるスケーラビリティの向上

スケーラビリティが要求される用途に仮想マシンを用いる場合は、より多くの仮想 CPU を仮想マシンに割り当てればよい。例えば上記のグラフは、アプリケーションサーバーのベンチマークである SPECjbb2005 を利用し、Integrity VM と物理マシンのスケーラビリティを比較したものである。ここで見るとおり、8 個の仮想 CPU を割り当てた仮想マシンで、物理マシンとほぼ同じスケーラビリティが得られることがわかる。

ここがツボ！「エンタイトルメント（使用権）」

先ほどの図 1 の例では、4 個の物理 CPU に対し、合計 8 個の仮想 CPU を各仮想マシンに割り当てている。当然のことながら、すべての仮想マシンの仮想 CPU に 100%の処理能力を配分することは不可能だ。そこで Integrity VM では、「エンタイトルメント（使用権）」と呼ばれる設定を行うことで CPU リソース配分の振る舞いをあらかじめ決めておくことができる。この「エンタイトルメント（使用権）」の設定を抑えれば、実運用環境に適した CPU リソース配分が可能になる。

「エンタイトルメント（使用権）」の設定では、CPU 使用権として「最低保証値」と「最大値」の 2 つの値を定義できる。例えば、すべての仮想マシンが高負荷状態にある場合でも、「最低保証値」で設定した CPU リソースはかならず確保される。また、リソースに余裕がある時には「最大値」で設定した分まで CPU リソースを利用でき、サーバーの CPU リソースを無駄なく使用できるわけだ。

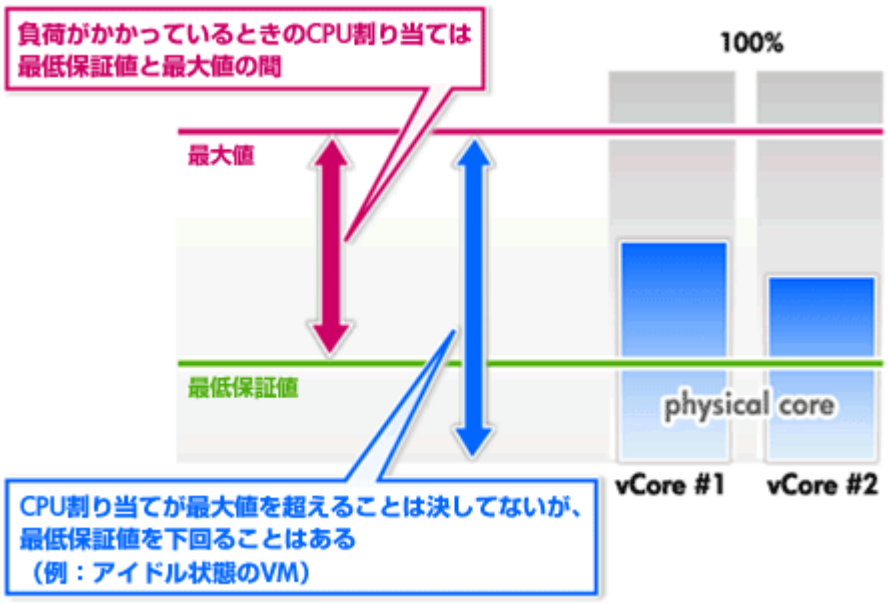

図 3：「最低保証値」と「最大値」

ただし、「無駄なく」というのだから「最大値」を 100%に設定しておけばいいかというと、実運用環境ではそうとばかりも言えない。「最大値」の設定はいわゆるリソースキャッピングを行うことを意味しており、仮想マシンで提供するシステムパフォーマンスの期待値を設定する場合や、制御する場合に役に立つ。

たとえば、仮想マシンを複数の異なる部署にホスティングするような場合を考えてみよう。そのような環境では、サービス開始時には提供する仮想マシンの数が少ないため、ユーザーは非常に快適なレスポンスを感じるだろう。しかし、やがて仮想マシンの数が増えていくにつれて、サービス開始時から利用しているユーザーはパフォーマンスの低下を感じ、クレームをつけることになる。この場合、サービス開始の時点でリソース割り当ての上限を設け、常に一定の性能を保証していれば、このようなクレームは避けることができるわけだ。

さらに、「最大値」の設定は、CPU リソースを非常に多く使う仮想マシンや異常な状態の仮想マシン(たとえば、内部でアプリケーションが暴走している)を封じ込めるためにも役に立つ。実運用環境で仮想マシンを利用する上では必須の機能といえるだろう。ちなみに、CPU 使用権の「最大値」の設定は Integrity VM 4.0 から可能になった設定だ。それ以前のバージョンを使っているのであれば是非バージョンアップを検討していただきたい。

仮想 CPU とエンタイトルメントの注意点

「最低保証値」と「最大値」といったエンタイトルメントの使い方でもうひとつ注意すべき点は、仮想 CPU との関係である。例えば、以下のような設定を行ったケースを考えよう。

	エンタイトルメント（最低保証値）	仮想 CPU 数
VM1	25%	4
VM2	100%	1
VM3	100%	1
VM4	100%	1

表 1：エンタイトルメントの最低保証値と仮想 CPU 数の設定例

このうち VM1 には、最低保証値＝25%、仮想 CPU 数＝4 と設定されているため、合計で 25%×4＝100%、すなわち CPU 1 個分のリソースを最低限確保することになる。一方、VM2～VM4 のほうも 100%×1＝100%と設定されており、それぞれ CPU 1 個分が保証される設定だ。よって、合計 4 個の物理 CPU を備えるサーバー上でこれらの仮想マシンを稼働させれば、すべての仮想マシンのエンタイトルメントを保証できると思われるかもしれない。

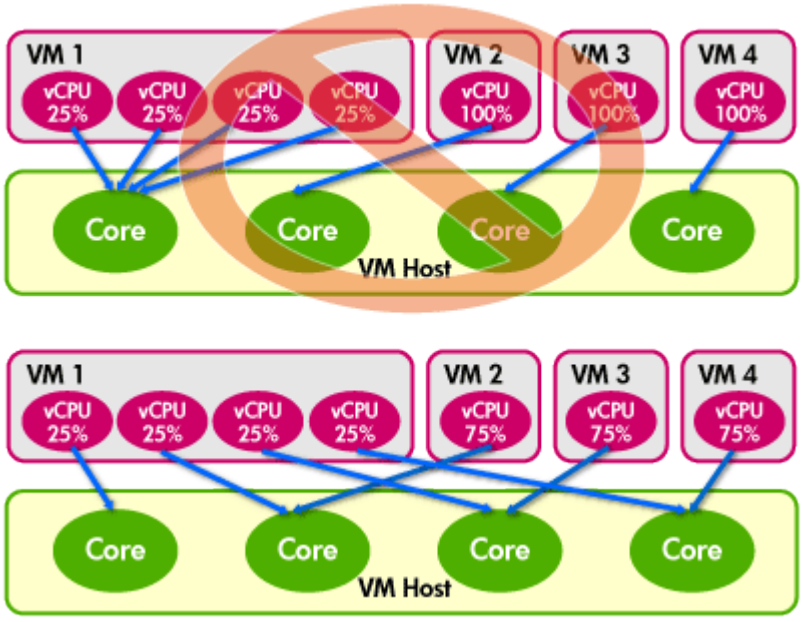

図 4：仮想 CPU とエンタイトルメントの注意点

しかし実際には、この設定ではすべての仮想マシンを起動することができない。なぜなら冒頭でも述べたとおり、Integrity VM の個々の仮想 CPU は、それぞれ別々の物理 CPU で稼働するからだ。VM1 の仮想 CPU が 4 つの物理 CPU すべてから 25%ずつリソースを確保するため、そのほかの仮想マシンに保証可能な CPU リソースは最大で 75%までなのである。CPU リソース配分の厳密な見積もりが不可欠な実運用環境では、こうしたすこし複雑な仮想 CPU やエンタイトルメントの概念を正しく理解して使いこなす必要が生じる。

以上、本稿では Integrity VM を実運用環境へ導入する際に重要となるノウハウのひとつとして、仮想 CPU リソースの配分設定について解説した。開発環境やテスト環境向けの仮想マシン利用ではあまり意識されない、厳密なサーバーリソース管理のコツをつかんでいただけたなら幸いである。

第 2 回

3 種類の仮想ディスクを使い分ける

2009 年 7 月

今回は、Integrity VM のパフォーマンスをフルに引き出すための「仮想ディスク」の構成のポイントを紹介する。仮想ディスクは、ゲスト OS からは一般的な SCSI ディスクとして認識され、また動的な追加も可能だ。この仮想ディスクを構成する手段としては、「Virtual FileDisk（ファイル）」、「Virtual LvDisk（論理ボリューム）」、そして「Virtual Disk（物理ディスクや SAN）」の 3 種類がある。これらそれぞれのメリットとデメリットを説明し、用途に応じた最適な仮想ディスク構成を考える。

「仮想ディスク」の構成方法を知る

前回は、Integrity VM のパフォーマンスをフルに引き出すためのポイントとして、「CPU リソースの割り当て」について説明した。今回は、「ディスク」の最適な設定方法を紹介する。

Integrity VM のゲスト OS は、「仮想ディスク」と呼ばれるストレージを利用できる。ゲスト OS からは、仮想ディスクは一般的な SCSI ディスクとして認識され、また動的な追加も可能だ。

この仮想ディスクを構成する手段としては、以下の 3 種類から選択できる。

- Virtual FileDisk（ファイル）
- Virtual LvDisk（論理ボリューム）
- Virtual Disk（物理ディスクや SAN）

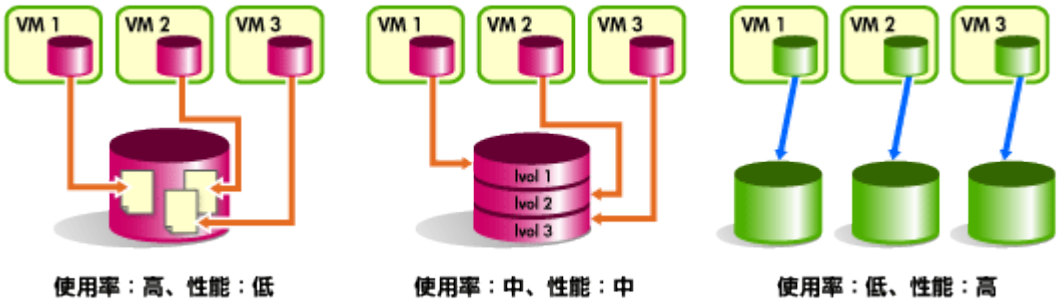

図 1：仮想ディスクを構成する 3 つの方法

ファイルを使う「Virtual FileDisk」

「Virtual FileDisk」は、ホスト OS 上に仮想ディスク用のファイルを作成し、仮想ディスクの内容を同ファイルに保存する方法である。これはもっとも手軽で柔軟性やポータビリティが高い方法であり、たとえば同ファイルをコピーして仮想マシン全体をコピーしたり移動したりできる。また、複数のゲスト OS の仮想ディスクを 1 つの物理ディスク上に集約できるため、ディスク領域の使用効率が高いというメリットもある。

その一方で、いわば「ホスト OS のファイルシステム上にゲスト OS のファイルシステムを構成する」ような二重構造になるので、パフォーマンス面では最良とは言い難い。例えばディスク・アクセスにともなうバッファリングも、ゲスト OS とホスト OS で重複して処理されることになる。また、Integrity VM の新機能である動的なマイグレーション（hpvmmigrate）をサポートできないという制限もある。

これらのメリットとデメリットを勘案すると、Virtual FileDisk は開発環境やテスト環境など、性能よりも管理性や使い勝手が重視される用途に適した構成と言える。

論理ボリュームを使う「Virtual LvDisk」

2 つめの「Virtual LvDisk」は、ファイルの代わりに、論理ボリューム 1 つを仮想ディスクとして割り当てる方法である。この場合、ホスト OS のファイルシステムを介さずに、論理ボリューム上に直接ゲスト OS のファイルシステムを構成するため、上述の Virtual FileDisk に比べてパフォーマンスが向上する。ただし、論理ボリュームをまるごと割り当てるため、ファイルと比較するとコピーや移動、バックアップなどの利便性は低くなる。また Virtual FileDisk と同様、hpvmmigrate によるマイグレーションをサポートできない。

物理ディスクを割り当てる「Virtual Disk」

最後の「Virtual Disk」は、物理ディスク全体を仮想ディスクとして設定する方法だ。この場合、たとえばバックアップやコピーは物理ディスクに対して実施する必要があり、柔軟性やポータビリティはぐんと低くなる。また、1 つのゲスト OS が 1 つの物理ディスクを占有するため、物理ディスク上には未使用の領域が生じてしまい、使用効率は低くなる。

これらのデメリットの一方で、Virtual Disk は仮想ディスクのパフォーマンスをもっとも引き出すことができる構成方法であり、ディスク I/O 性能重視の用途や実運用での利用には Virtual Disk の利用が推奨される。

仮想ディスクのパフォーマンスを高める 2 つのツボ

以下に、これら 3 種類の仮想ディスク構成方法について、メリットとデメリットをまとめた表を示す。

	Virtual FileDisk	Virtual LvDisk	Virtual Disk
概要 (仮想ディスクの実体)	ホスト OS 上の 1 ファイル	ホスト OS 上の 1 論理ボリューム (/dev/vgXX/rlvolXX)	ホスト OS 上の 1 物理ディスク (/dev/rdisk/diskX) (/dev/rdsk/cXtXdX : Legacy)
扱いやすさ (容量拡張, 容量効率)	◎	○ LVM で論理ボリュームの容量拡張可能	△ ホスト OS 上では困難だが、ゲスト上の LVM で対応も可能
わかりやすさ (実デバイスとの対応、構成・I/O 性能管理)	△ ファイル～LVM(LV、VG、PV)～実デバイスの対応をユーザーが把握する必要がある	○ ホスト OS 上の LVM 設定をユーザーが管理する必要がある	◎ 実デバイスと 1:1 なので管理しやすく、利用する実 HBA の把握も容易
性能	○ もっとも処理オーバーヘッドが多い	○	◎ もっとも処理オーバーヘッドが少ない
ゲスト OS マイグレーション(移動)機能	△ offline のみ	× サポートされない	◎ offline、online 両方サポート(online は hpvm v4.1 以降のみ)
ディスクパス、ディスク冗長化対応	○ ホスト OS の Native Multipath ストレージの RAID、MirrorDisk	○ ホスト OS の Native Multipath ストレージの RAID、MirrorDisk	○ ホスト OS の Native Multipath ストレージの RAID
1 ゲストに設定可能なデバイス数	合計：30 VIO + 128 AVIO		
	30 VIO + 30 AVIO	30 VIO + 128 AVIO	30 VIO + 128 AVIO

では、仮想ディスクの構成に際して、パフォーマンスを高めるためのポイントを見ていこう。

物理ディスクへの経路や論理ボリュームの構成を意識

仮想ディスクの性能を引き出すには、物理ディスクへのアクセスに用いられる経路や、論理ボリュームの構成を意識した割り当てが重要となる。例えば、1 つの経路やボリュームグループに複数の仮想ディスクを集約してしまうと、そこがボトルネックとなり性能は頭打ちとなる。よって、物理ディスクへの経路やボリュームグループ等の共有度をできるだけ下げるのがコツとなる。必要に応じて HBA 等を追加し、負荷を分散させる工夫も検討すべきだろう。

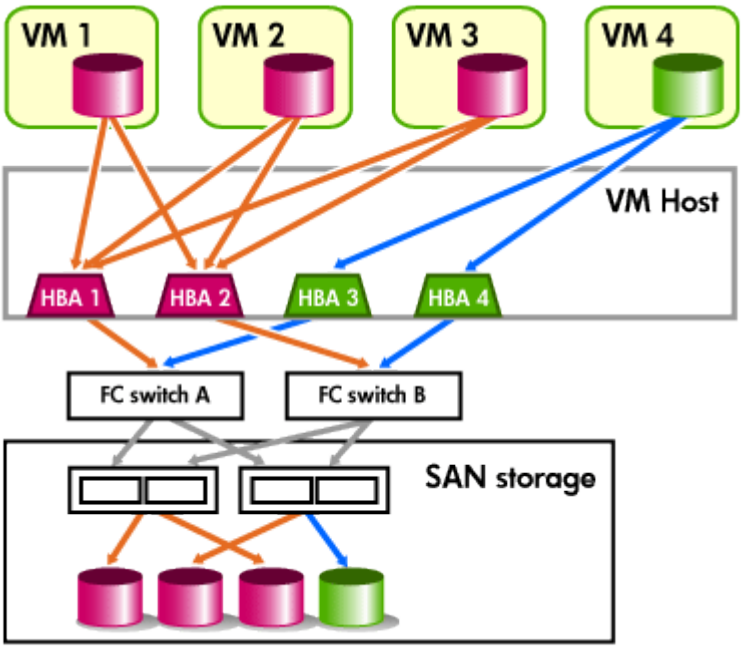

図 2：物理ディスクへの経路を意識する

仮想 SCSI バスのターゲット ID に気をつける

冒頭でも説明したとおり、仮想ディスクはゲスト OS から一般的な SCSI ディスクとして認識され、また動的な追加も可能である。ただし、この動的な追加を行うには、仮想 SCSI バスのターゲット ID に空きがなくてはならない。デフォルトで使用される仮想 SCSI バスは単一ポートのパラレル SCSI MPT をエミュレートするため、1 つの仮想 SCSI バスに定義できるターゲット ID は 15 までに限られており、これ以上の仮想ディスクを追加するには、ゲスト OS を停止して仮想 SCSI バスを追加する必要がある。動的なデバイス追加を必要とする場合には、あらかじめターゲット ID に余裕のある仮想 SCSI バスを複数定義しておくか、ターゲット ID に制限の無い"AVIO ストレージ"で構成しておくのが望ましい。

ゲスト OS に物理ディスクを定義する場合には、以下の hpvmmodify コマンドを使用する。

```
# hpvmmodify -P vm01 -a disk:scsi:0,1,0:file:/vm/vm01disk
```

VM 上での見え方の指定：
device:driver:pcibus,pcislot,scsitgt

物理リソースの指定：
storage:location

device	：	disk、dvd、tape、changer、burner
driver	：	scsi、avio_stor
pcibus	：	仮想マシン上での PCI バス番号（0～6）
pcislot	：	仮想マシン上での PCI スロット番号（0～7）
scsitgt	：	仮想マシン上での SCSI ターゲット番号（0～14）[driver が avio の場合は 0-127]
storage	：	disk、lv、file、null、attach
location	：	ホスト上のリソース名

このとき、PCI バス番号や PC スロット番号を適宜割り振ることで、複数の SCSI バスを定義可能だ。

以上、今回は Integrity VM の仮想ディスク構成方法のポイントを説明した。次回は、仮想ネットワークの構成方法を解説する予定である。

第 3 回

仮想スイッチ構成とメモリ構成のポイント

2009 年 8 月

今回は、「仮想スイッチ」によるネットワーク構成と、仮想マシンのメモリ構成のポイントについて解説する。Integrity VM の特徴のひとつは、仮想的なネットワークスイッチである仮想スイッチを備えている点だ。この仮想スイッチを作成することで、各仮想マシンや物理 NIC を結ぶ仮想的なネットワークセグメントを構成できる。仮想スイッチを構成する上では、個々の物理 NIC にどの程度の負荷が集中するか把握しておくことが重要だ。必要に応じて仮想スイッチを分割し、それぞれに個別に物理 NIC を割り当てるといった工夫が必要となる。

「仮想スイッチ」のコツをつかむ

前回は、Integrity VM の「仮想ディスク」の最適な活用方法を紹介した。今回は、「仮想スイッチ」によるネットワーク構成と、仮想マシンのメモリ構成のポイントについて解説する。

Integrity VM の特徴のひとつは、仮想マシンのための仮想的なネットワークスイッチである仮想スイッチを備えている点だ。仮想スイッチを作成することで、各仮想マシンや物理 NIC を結ぶ仮想的なネットワークセグメントを構成できる。仮想スイッチの設定はすべてコマンド操作で完結するため、物理的なサーバーを並べ、それらをネットワークハブに接続し……といった手間のかかる作業は不要となる。

また仮想スイッチには、物理 NIC の代わりに APA（Auto Port Aggregation）を割り当てることもできる点もおさえておきたい。APA とは、複数のネットワークリンク（最大 4 本）を束ね、1 本の論理的なリンクとして利用できる HP-UX の機能だ。ネットワークパケットをそれぞれの物理 NIC に分散させて「1 本の広帯域ネットワーク」として利用できるほか、いずれかのリン

クがダウンした場合でもパケットを他のリンクに振り分ける冗長性を提供する。仮想スイッチに APA を割り当てることで、ネットワーク帯域を動的に変更することができる。

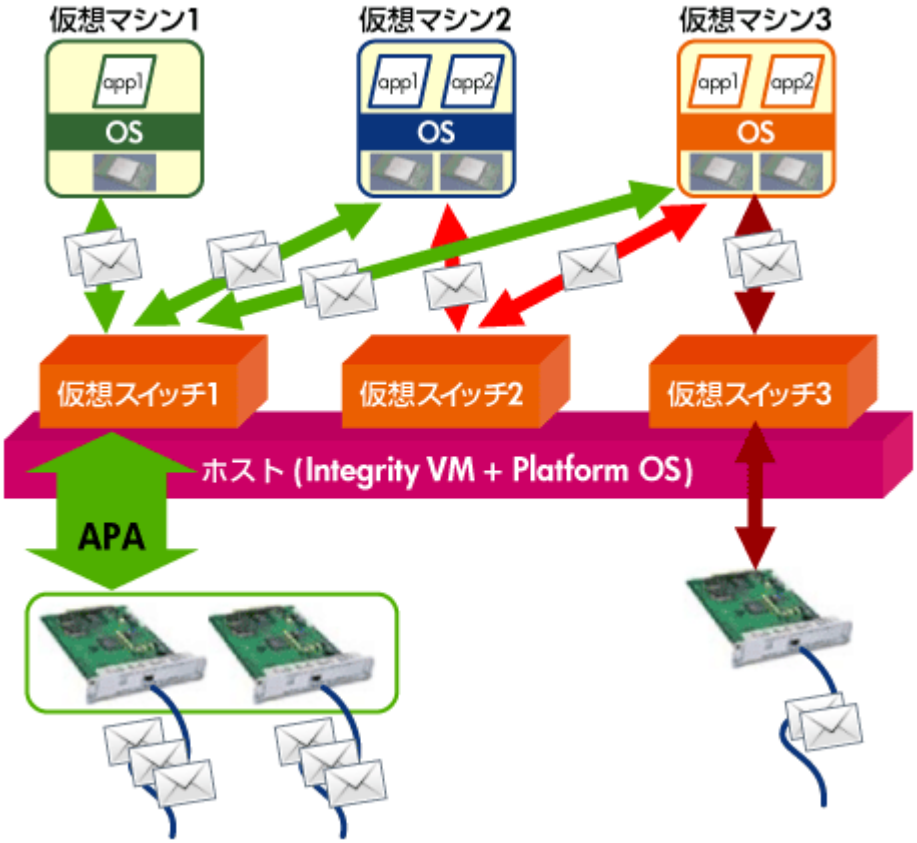

図 1：仮想スイッチと APA の利用

同じ仮想スイッチに接続された複数の仮想マシンは、その仮想スイッチに割り当てられた物理 NIC のネットワーク帯域を分け合うことになる。ネットワーク I/O 負荷の大きいアプリケーションでの利用には注意が必要だ。
複数の仮想スイッチを構成する上では、個々の物理 NIC にどの程度の負荷が集中するか把握しておくことが重要になる。必要に応じて仮想スイッチを分割し、それぞれに個別に物理 NIC を割り当てたり、特定の仮想マシンのみに別の仮想スイッチを割り当てるといった工夫が必要だ。

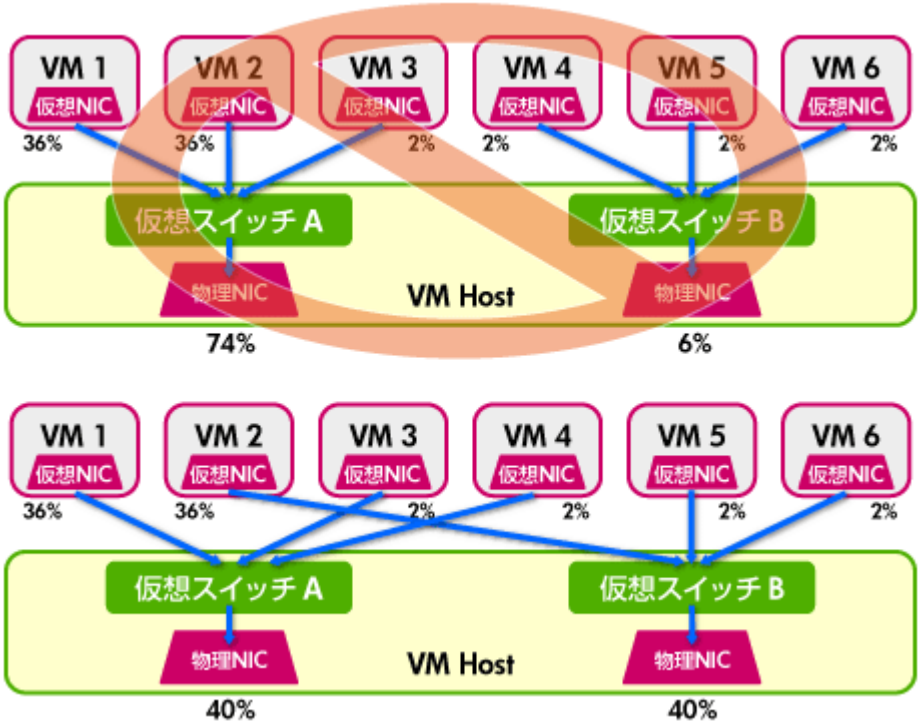

図 2：仮想スイッチへの負荷集中を避ける

仮想スイッチは、物理 NIC に接続しない構成も可能である。この場合、仮想マシン間のネットワーク通信は、物理 NIC を介さずに仮想スイッチ内部に閉じたかたちで行われる。例えばアプリケーション開発においてテスト環境を構築するようなケースでは、仮想のクライアントやサーバーとなる仮想マシンを作成し、それらを仮想スイッチで結ぶだけで環境設定は完了する。また、物理ネットワークとは切り離した別セグメントを構成することで、セキュリティ面の隔離性も得られる。

メモリ構成のポイント

仮想マシンのメモリ構成も、アプリケーションのパフォーマンスを最適化する上で重要なポイントだ。Integrity VM では、個々の仮想マシンに割り当てたメモリ容量の合計に加え、「ホスト OS となる HP-UX が必要とするメモリ」、そして「仮想マシンのオーバーヘッド分のメモリ」を合計した容量を用意しておく必要がある。具体的には、それぞれ以下の方法で概算できる。

- ホスト OS のメモリ消費：750MB ＋ (物理メモリサイズ － 1 GB) × 0.085
- 仮想マシンのオーバーヘッド：仮想マシンのメモリサイズの 8～8.3%

仮想マシンのゲスト OS が HP-UX の場合は、仮想マシンの動作を止めることなくメモリ構成を変更できる「dynamic memory 機能」（オプション）が利用可能だ。ただし、仮想マシン起動時のメモリサイズが最大値となるため、「あらかじめ予想される最大メモリサイズで起動しておき、メモリ消費が少ない状況ではメモリサイズを小さくする」といった使い方になる。

また、メモリ性能の点では「メモリページの断片化」についても考慮すべきだろう。仮想マシンのメモリ性能は、ホスト OS が管理するメモリページの断片化の影響を受ける。よって Integrity VM では、ホスト OS にて「ラージページ」機能を使用し、仮想マシンに割り当てたメモリのロックダウンを行う仕組みが働いている。しかし、例えば Integrity VM 以外のアプリケーションが同じホスト OS 上で動作している場合や、上述の dynamic memory 機能を高頻度に使用した場合などは、断片化が進行する可能性があることを考慮しておきたい。

以上、今回はネットワーク構成とメモリ構成の最適化について解説した。次回は、仮想マシンのバックアップ構成について紹介する予定だ。

第 4 回

Integrity VM のバックアップ手段と AVIO の利用

2009 年 9 月

仮想マシンのシステムバックアップは、通常のサーバーの場合と同じツール、手法を使用して実行することが可能だ。ただし、Integrity VM に固有の要件を考慮すべきポイントもいくつかある。そこで今回は、ホスト OS のバックアップの観点から解説する。また、Integrity VM のネットワーク性能およびディスク性能を改善する上で、いわば“必須アイテム”である「Accelerated Virtual I/O（AVIO）」についても紹介する。

Integrity VM のバックアップ方法

前回は、Integrity VM の「仮想スイッチ」によるネットワーク構成と、仮想マシンのメモリ構成のポイントについて説明した。今回は、Integrity VM における「バックアップ」のポイントについて解説する。

仮想マシンのシステムバックアップは、通常のサーバーの場合と同じツール、手法を使用して実行することが可能だ。ただし、Integrity VM に固有の要件を考慮すべきポイントもいくつかある。まずは、ホスト OS のバックアップの観点から見ていこう。

ホスト OS のバックアップ

ホスト OS をバックアップする際には、ホスト OS の/var/opt/hpvm ディレクトリに格納されているゲスト OS の構成情報をもれなくバックアップする必要がある。この構成情報が万が一失われてしまうと、たとえゲスト OS の仮想ディスクが残っていたとしてもゲスト OS を起動できなくなるので注意が必要だ。

またゲスト OS の仮想ディスクが「Virtual FileDisk」、すなわちファイルベースで構成されている場合は、ホスト OS のバックアップ時にゲスト OS も含めるかたちでまるごとバックアップすることも可能だ。ただしこの場合、バックアップ対象となるゲスト OS はすべてシャットダウンしておく必要がある。仮想ディスクが「Virtual LvDisk」（論理ボリューム）または「Virtual Disk」（物理ディスクや SAN）で構成されている場合は、ゲスト OS のバックアップを個別に実施しなくてはならない。

ゲスト OS のバックアップ

一方、ゲスト OS を個別にバックアップする場合も、考慮すべき点がいくつかある。基本的には通常の HP-UX システムにおけるバックアップ作業と同じであるが、まず先に述べたとおりゲスト OS の構成情報はホスト OS 上にあるため、ゲスト OS 内部のバックアップ作業では構成情報をバックアップできないという点だ。よってホスト OS を対象とした構成情報のバックアップも個別に実施しておく必要がある。

Ignite サーバーによるバックアップ

HP が提供するクライアント／サーバー型のツール「Ignite-UX」を活用すれば、Integrity VM の高い柔軟性を最大限に引き出すことが可能だ。Ignite-UX は HP-UX システムのインストールやリカバリ作業を支援するソフトウェアで、物理マシンと同様に仮想マシンへのシステムインストールや、システムイメージのバックアップを行うことができる。そこで、Ignite-UX によるバックアップ構成の例とそのメリットをいくつか見ていこう。

Ignite-UX サーバーによるバックアップ構成【1】

この構成は、もっとも標準的な“おすすめ”構成である。ホスト／ゲストそれぞれの OS で make_net_recovery コマンドを実行し、システムのリカバリイメージを Ignite-UX サーバーに保存する。仮想マシンの構成情報はホスト OS のリカバリイメージに含まれる。

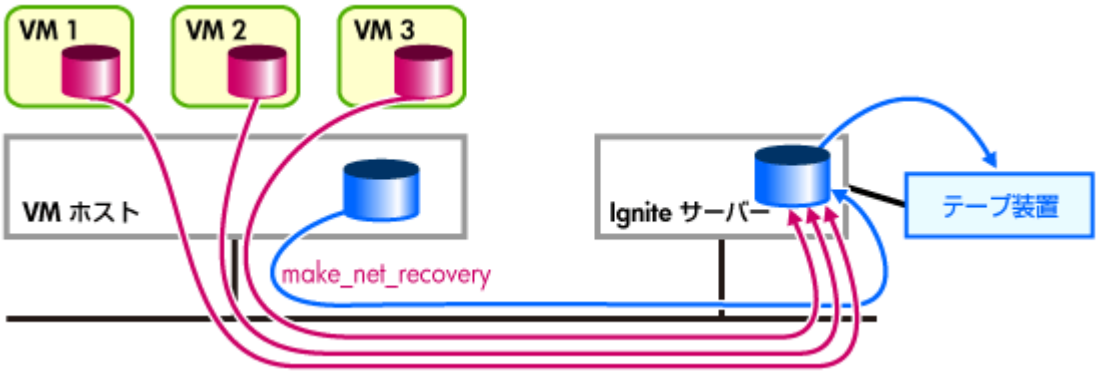

図 1：Ignite-UX サーバーによるバックアップ構成【1】

Ignite-UX サーバーによるバックアップ構成【2】

この構成は、ホスト OS 上に Ignite-UX サーバーを構成する方法である。個々のゲスト OS にて make_net_recovery コマンドを実行し、「Integrity VM 専用」に用意されたホスト OS 上の Ignite-UX サーバーにバックアップを指示する。この方式のメリットはホスト OS の管理者、つまり仮想マシンを作成する人が専用の Ignite-UX サーバーを持ち、仮想マシンの管理に合わせて自由にカスタマイズできる点だ。ホスト OS のバックアップに関しては構成 1 と同様、ネットワーク上の Ignite-UX サーバーに保存する。このとき、ゲスト OS のリカバリイメージをホスト OS のリカバリイメージの一部として保存する事も可能だ。

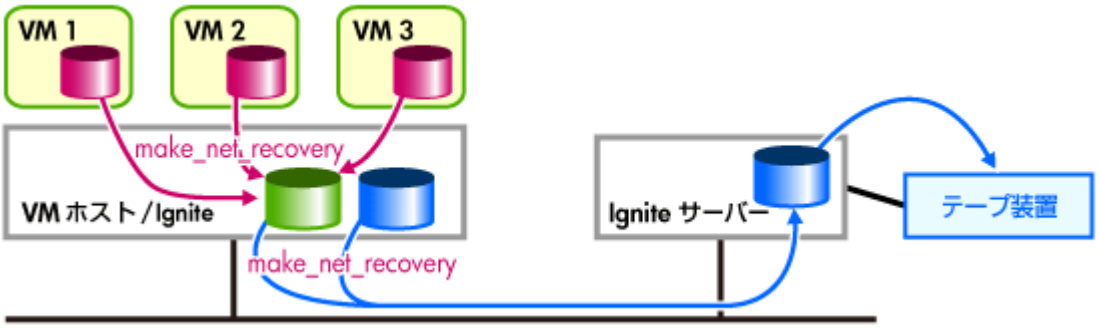

図 2：Ignite-UX サーバーによるバックアップ構成【2】

Ignite-UX サーバーによるバックアップ構成【3】

一方、Ignite-UX サーバー専用のマシンを他に設置できない場合は、ホスト OS が載るマシンにテープデバイスを直接接続する構成も可能である。この場合、個々のゲスト OS は先ほどと同じように make_net_recovery コマンドを用いてホスト OS 上にインストールされた Ignite-UX サーバーにバックアップを実行する。また、ホスト OS では make_tape_recovery コマンドを利用して、ホスト OS に接続したテープデバイスに対するバックアップを実行するという流れになる。

Ignite-UX サーバーを仮想化環境で使いこなす

このようにゲスト OS を Ignite-UX サーバーによってバックアップできる構成を整えておくことで、Integrity VM の使いやすさを一段と高めることが可能だ。例えば、新しいゲスト OS を追加する場合でも、DVD ドライブに OS メディアを挿入してゼロから HP-UX をインストールするような手間がかからない。

またゲスト OS の OS イメージをまるごとバックアップすることも可能なので、例えば既存の開発環境をバックアップしておき、それを他のゲスト OS としてリカバリすることで、特定の開発環境を簡単に用意することが可能になる。

AVIO で I/O 性能を改善する

さて、バックアップ環境と並び、Integrity VM の真価を一段と引き出すために重要なポイントとなるのが I/O 性能である。とりわけ、ネットワーク性能およびディスク性能を改善する上で、いわば“必須アイテム”と言えるのが、「Accelerated Virtual I/O（AVIO）」である。AVIO とは、仮想マシン専用のディスクドライバーおよびネットワークドライバーだ。

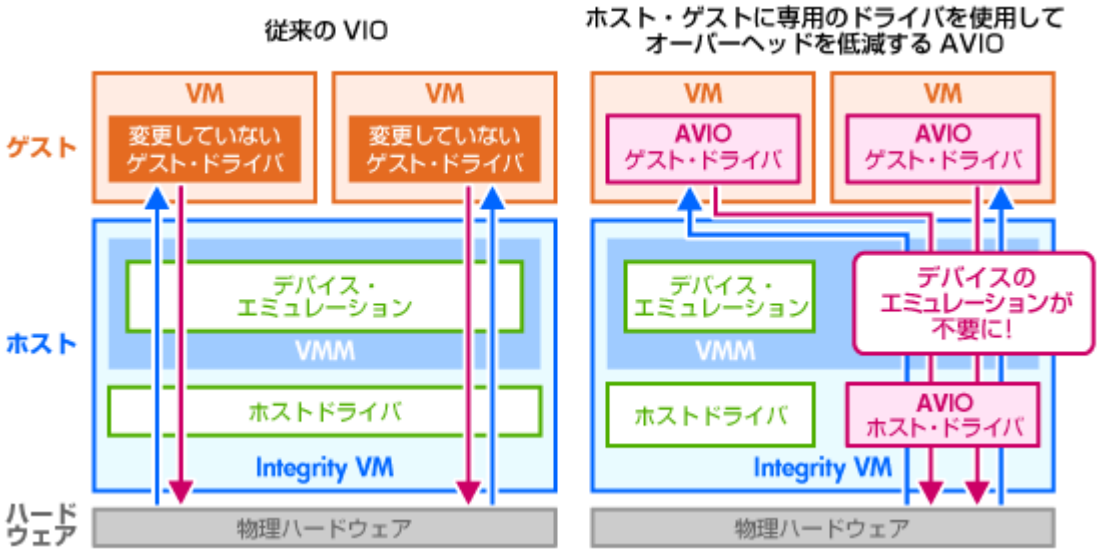

図 3：AVIO のメカニズム

この図が示すように、標準の Virtual I/O（VIO）では、ゲスト OS とホスト OS の間で動作する VM モニタがデバイスエミュレーションを行うため、この部分で I/O 処理のオーバーヘッドが発生する。AVIO は、このオーバーヘッドをスキップすることで、VIO と比較して最大 2 倍の帯域幅が得られるほか、CPU 負荷を最大 50%軽減できる。AVIO がサポートするデバイスの種類は VIO よりも少ないが、両者は併用することが可能だ。よって、AVIO が対応するデバイスについては原則として AVIO を使用することがおすすめである。

以下は、AVIO と VIO のメリットとデメリットを比較した表である。

	メリット	デメリット
VIO	・OS 標準のドライバーである	・エミュレーションによるオーバーヘッドが大きい ・LUN の最大数は 30 まで
AVIO	・高性能、低オーバーヘッド ・LUN の最大数は 128	・専用デバイスドライバーのインストールが必要 ・一部の古いカードがサポート対象外

表：AVIO と VIO の比較

ちなみに、7 月に公開された Integrity VM のパッチリリースでは、AVIO のサポート範囲が拡張され、これまで実施できなかったリムーバブルデバイス（DVD ライターやテープデバイスなど）をゲスト OS 間で共有可能となった。これにより、例えばバックアップ作業時に、テープデバイスを個々のゲスト OS にいちいち付け替える作業は不要となっている。

以上、今回は Integrity VM 利用時のバックアップの構成方法と、AVIO によるパフォーマンス改善の方法について説明した。

第 5 回

オンラインマイグレーション機能の実際

2009 年 10 月

2009 年 4 月にリリースされた最新版「Integrity VM 4.1」では、新たに「Integrity VM オンラインマイグレーション」機能が利用可能になった。これは、Integrity VM 上で動作するゲスト OS（仮想マシン）を、稼働状態のまま別のホスト OS へと移行するという機能である。そこで今回は、このオンラインマイグレーション機能の特徴と使い方を紹介したい。

ゲスト OS を数秒で移行できる「オンラインマイグレーション」

Integrity VM のメリットのひとつは、ゲスト OS の「ポータビリティ」の高さである。物理マシンとは異なり、ゲスト OS の「実体」はファイルや論理ボリューム、もしくはディスクドライブに収まっているため、それらを別のマシンに移動するだけでゲスト OS を簡単に移動できる。物理マシン上でアプリケーションを直接運用する場合とは異なり、移動先の物理マシンの OS 環境を整えたり、アプリケーションをインストールし直したりする手間がかからない。こうした Integrity VM のポータビリティの高さを生かすことで、例えば以下のようなメリットが得られる。

- あるホスト OS に負荷が集中している場合、他のホスト OS を用意してゲスト OS を移動することで、負荷を簡単に分散できる。
- あるホスト OS に対してメモリやディスクの増設、部品交換などのメンテナンスを実施したい場合、他のホスト OS にゲスト OS を一時的に移動してからメンテナンスすることで、サービスの停止を最小限に抑えられる。

しかし従来は、こうしたゲスト OS の移動に先立って、ゲスト OS をいったんシャットダウンしてサービスを停止し、移動先でサービスを再起動するという手順が不可欠であった。よって、例えば「夜間や月末だけゲスト OS を他のマシンに移動したい」といった具合に、臨機応変の負荷分散を実現できるほどの手軽さで移動を行えるとは言い難かったのである。

これに対し、今回リリースされたオンラインマイグレーション機能では、ゲスト OS を稼働させたままで移行が可能になる。このとき、ゲスト OS 上で動作するアプリケーションから見ると、ストレージとネットワークといった I/O 接続はすべてアクティブのままとなる。よって、ゲスト OS やアプリケーションを再起動することなく動作させ続けることが可能だ。ただし移行作業の最終段階では、数秒～10 秒程度の短いあいだゲスト OS がフリーズすることになる。とはいえ、上述のようなオフライン作業での移行に比較して段違いに高速な移行が可能になる。

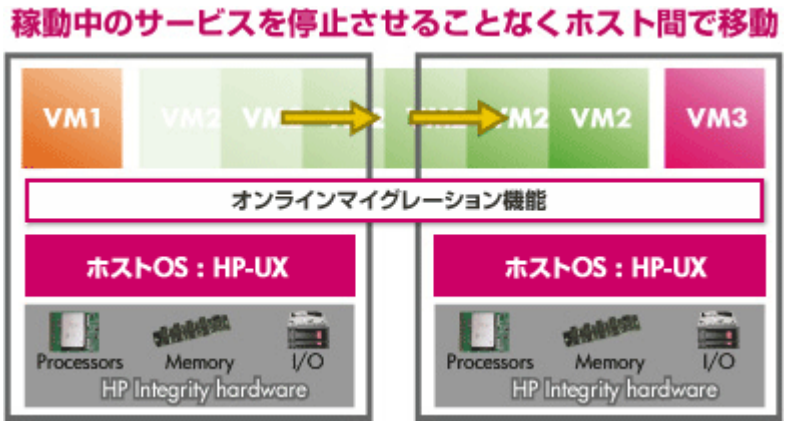

図 1：ゲスト OS を稼動したまま移動可能なオンラインマイグレーション

こうした劇的な高速化によって、以下のようなメリットが得られる。

- プロアクティブな保守が容易になる。例えば、ホスト OS のハードウェアに大規模障害の前兆となりそうなワーニング等が確認された場合、すぐさまオンラインマイグレーションを実施し、サービスを止めずに予防的な保守を実施できる。
- きめ細かな負荷分散が可能になる。あるホスト OS 上のゲスト OS に負荷が集中し、一方で他のホスト OS の負荷には余裕があるような場合、オンラインマイグレーションを用いることで、「夜間だけ」「月末だけ」「特定のイベント期間中だけ」といった短いスパンで臨機応変な負荷分散が容易になる。また、サービス規模が急成長した場合の迅速なインフラ拡張や、逆に負荷があまり高くないゲスト OS を特定のホスト OS に集めて最適化するといった作業が可能になる。

オンラインマイグレーション利用の前提条件

では、Integrity VM におけるオンラインマイグレーションの具体的な流れを説明したい。まずは、オンラインマイグレーションの実行に際して、移行元と移行先の両ホストが満たしておくべき前提条件を確認しておこう。

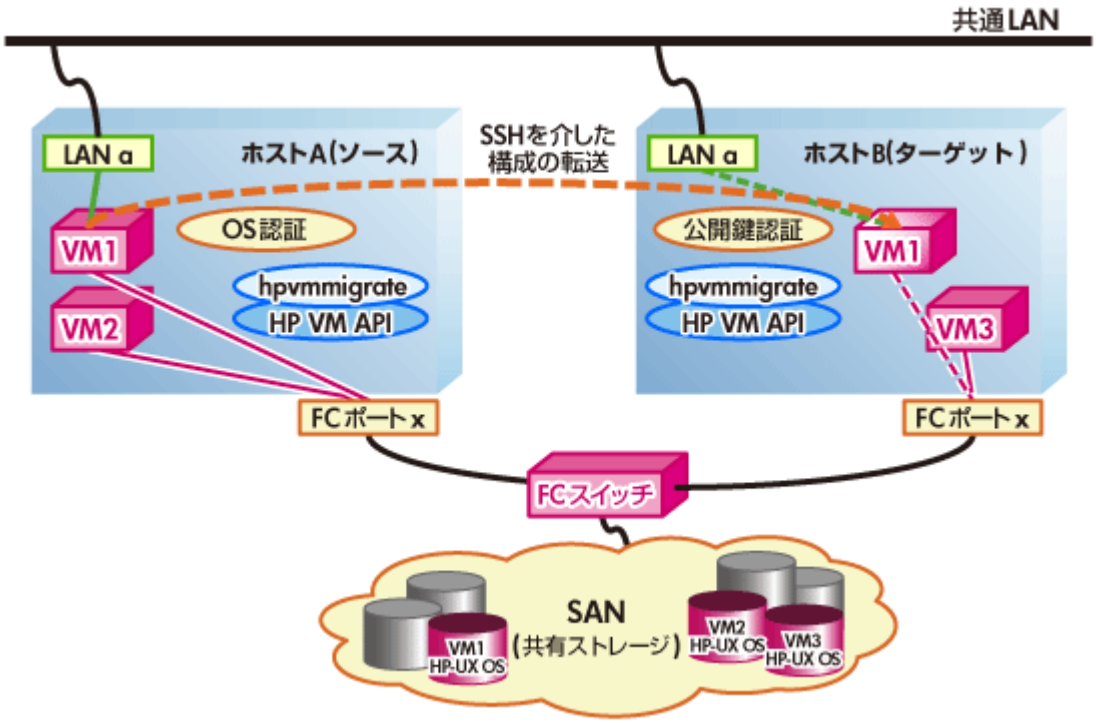

図 2：オンラインマイグレーション用に構成されたホスト

- Integrity VM のバージョンが 4.1 以降であること
- 搭載するインテル® Itanium® プロセッサーが、同じプロセッサーファミリー（例えば Montecito など）であること（なお、プロセッサー数は異なっていてもよい）
- 共通する LAN に接続されていること（マイグレーション専用の高速ネットワークを推奨）
- ゲスト OS の実体を格納したディスクドライブが SAN（Storage Area Network）に接続されていること
- 移行元でゲスト OS が利用するリソースを、移行先でも利用できること（例えば、割り当てるメモリーサイズ、ストレージの LUN、ネットワーク接続、仮想スイッチなど）
- 移行元と移行先のシステムクロックが NTP によって同期されていること

これらの条件が移行元／移行先の両ホストで満たされていれば、稼働中のゲスト OS を止めずにマイグレーションが可能となる。

オンラインマイグレーションの実例

ではここで、オンラインマイグレーションの実際の例を紹介したい。ここでは仮想マシンでデータベースを動作させたままオンラインマイグレーションを行った場合の動きを、検証中のデータを交えながら順を追って紹介しよう。

以下の画面は、移行前のホスト OS およびゲスト OS の稼働状況を管理ツール Integrity Essentials Virtualization Manager（以下、Virtualization Manager）上で表示したものである。

図 3：Visualization Manager 上でホスト OS とゲスト OS を表示した例

この画面では、ホスト OS「hpjwit143」上で 2 つのゲスト OS「hpjwit100」および「hpjwit101」が稼働している状況が表示されている。このうち、hpjwit100 上ではアプリケーションが稼働中だ。このアプリケーションを稼働させたまま、hpjwit100 をもうひとつのホスト OS「hpjwit145」に移行する検証を行う。

以下の画面は、オンラインマイグレーションを実行した際のコマンド表示例である。

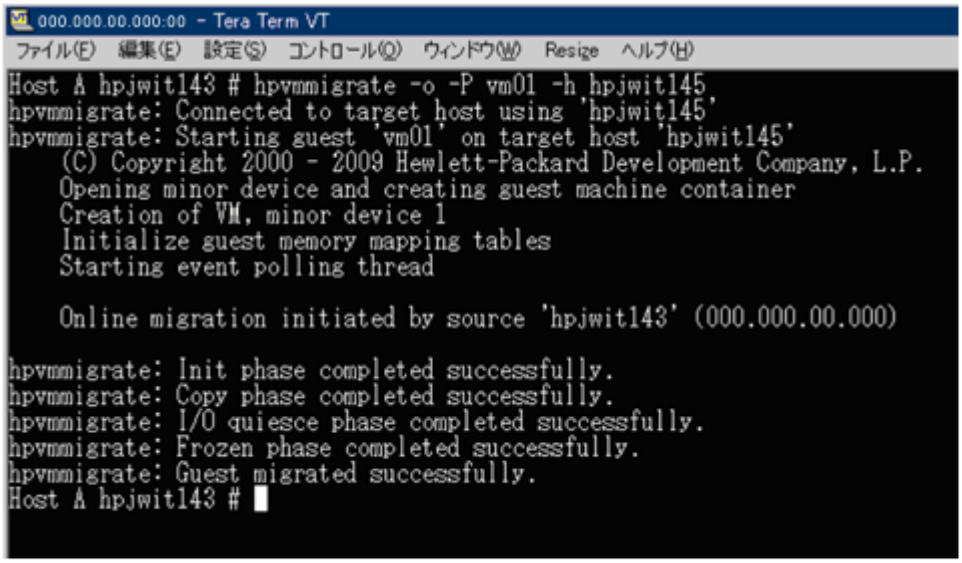

図 4：オンラインマイグレーションの実行例

ここではまず、移行元のホスト OS である hpjwit143 にて以下の hpvmmigrate コマンドを実行し、オンラインマイグレーションを開始している。

```
# hpvmmigrate -o -P vm01 -h hpjwit145
```

ここで、オプション「-P vm01」は移行するゲスト OS（vm01＝hpjwit100）を指定し、またオプション「-o」はオンラインマイグレーションの実行を指定している。またオプション「-h hpjwit145」では、移行先のホスト OS を指定している。

このコマンドを実行すると、以下の各フェーズを経てマイグレーションが実行される。

- 初期化フェーズ：ホスト OS 間の接続の確立、各種の確認、移行先ゲスト OS の起動など
- コピーフェーズ：ゲスト OS のメモリ書き込みを追跡し、メモリー内容をすべてコピーする
- I/O 静止フェーズ：新しい I/O リクエストの発行を停止し、既に発行された I/O が完了するのを待つ
- フリーズフェーズ：ゲスト OS の仮想 CPU を停止し、変更されたメモリー内容とゲスト OS の状態を移行先にコピーする

これらの各フェーズを経て、コンソール上に「Guest migrated successfully.」と表示されると、以下のように Virtualization Manager 上ではゲスト OS が新しいホスト OS 上に移動したことを確認できる。

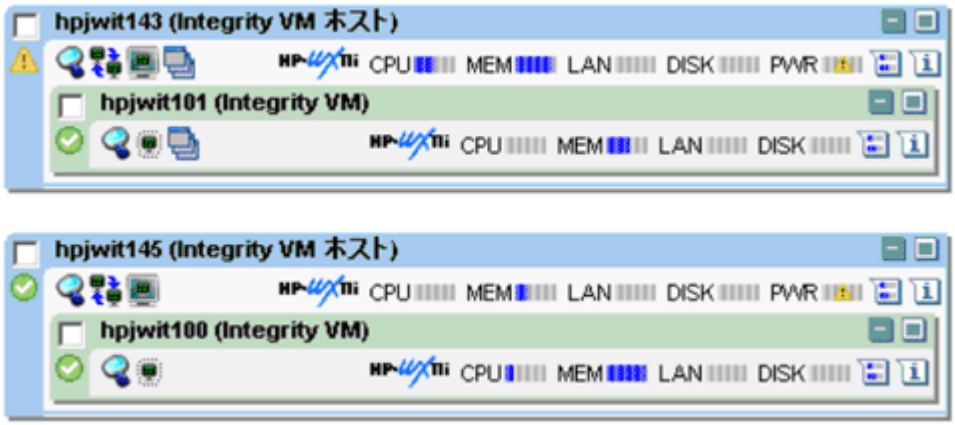

図 5：オンラインマイグレーション実施後のホスト OS とゲスト OS

上述した 4 つのフェーズのうち「初期化フェーズ」から「I/O 静止フェーズ」までは、アプリケーションを稼働させた状態で、移行元から移行先へのゲスト OS のメモリ内容転送といった時間のかかる処理が実施される。今回の検証では、これらの処理におよそ 15 秒を要している。そして最後に、ゲスト OS 上のアプリケーションをフリーズさせて移行を完了する「フリーズフェーズ」が実施される。同フェーズは、今回の検証ではおよそ 7 秒程度で完了している。

実際に、仮想マシン上のアプリケーションの動作状況を管理ツールで確認してみたのが次の画面である。

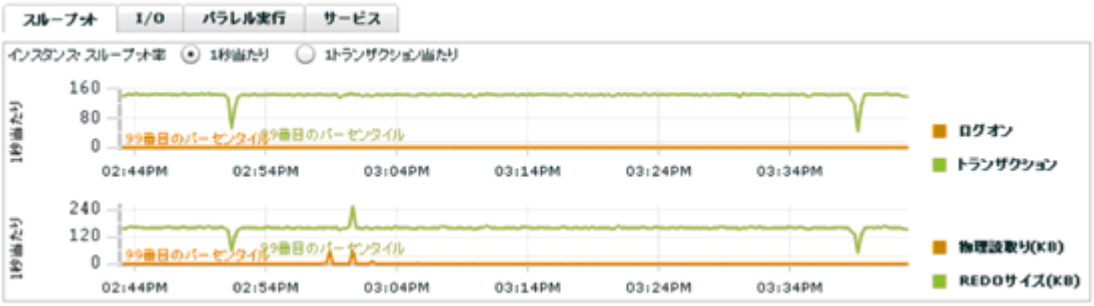

図 6：オンラインマイグレーション実施時のアプリケーションの性能グラフ

グラフ右端の部分で、1 秒あたりのトランザクション数が一瞬落ち込んでいることが確認できる。この部分は、上述したフリーズフェーズによって 7 秒間だけアプリケーションがフリーズした状態を示している。このように Integrity VM のオンラインマイグレーションでは、ゲスト OS 上のアプリケーションへの影響が最小限に抑えられることがわかる。

以上、今回は Integrity VM のオンラインマイグレーション機能の特徴と実際の使用例を紹介した。次回は Integrity VM における高可用性（HA）構成について説明する予定である。

最終回

仮想化環境でのクラスターウェアの使いこなし術

2009 年 11 月

Integrity VM は HP-UX のための仮想化技術であることから、ミッションクリティカル環境での利用がおのずと多くなるはずだ。そこで今回は、HP-UX での可用性確保になくてはならないクラスターウェアである Serviceguard と Integrity VM の組み合わせについて説明する。ここで紹介する構成はいずれも HP 社内で検証済みでありサポート対象のソリューションであるため、ぜひ実運用環境への導入も積極的に検討していただきたい。

「ゲスト OS＋Serviceguard」構成の使い方

Integrity VM と Serviceguard（以下、Serviceguard）を組み合わせるには、大きく分けて以下の 2 種類の方法がある（ちなみにこの両者の共存も可能である）。

- ゲスト OS＋Serviceguard
- ホスト OS＋Serviceguard

まずは、「ゲスト OS で Serviceguard を使う」方法について説明しよう。この場合、ゲスト OS の内部に Serviceguard をインストールし、ゲスト OS 上で動作する個々のアプリケーションについて Serviceguard のパッケージを作成する。これにより、アプリケーション障害をはじめ、ゲスト OS の障害、および物理サーバーの障害のいずれが発生した場合でも、スタンバイ側のゲスト OS 上で動作するアプリケーションにフェイルオーバーされ、サービスの継続が可能になる。

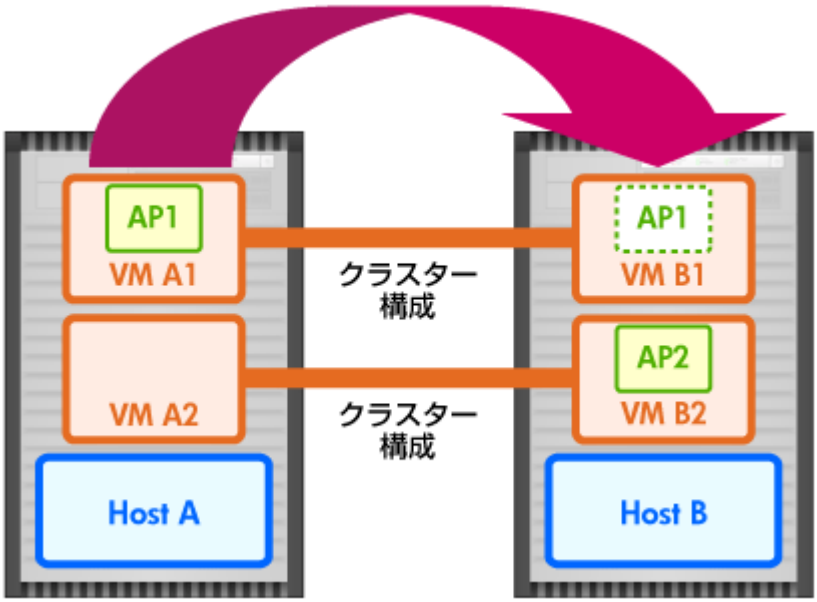

図 1：ゲスト OS 内で Serviceguard を使う

この方法のメリットは、既存の Serviceguard パッケージ設定やスクリプト等をそのまま利用できる点だ。Serviceguard クラスターの運用管理の手順等も特に変える必要がない。

ボックス内クラスターで開発／テスト環境構築

またゲスト OS＋Serviceguard 構成では、「ボックス内クラスター」と呼ばれるユニークな構成も可能だ。これは、1 台の物理マシン上で 2 つのゲスト OS を稼働させ、それらの間で Serviceguard によるフェイルオーバーが可能なパッケージを構成する方法である。

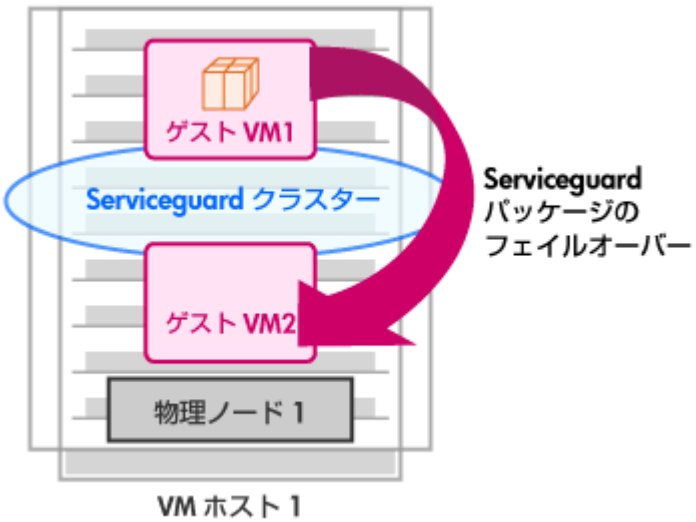

図 2：1 台の物理マシン上で動く 2 つのゲスト OS 間で Serviceguard を使う

もちろん、この場合は物理マシンの障害に対する保護は行われないものの、両ゲスト OS 間でのストレージやネットワークセグメントの共有が簡単に構成できる。“本物”の Serviceguard クラスターを構築するために必要な SAN やネットワーク機器を用意し

なくてもよいので、例えば Serviceguard を利用するシステムの開発環境やテスト環境、もしくは Serviceguard のトレーニング環境などの構築に便利な方法である。

仮想／物理クラスターでスタンバイ機を集約

ゲスト OS＋Serviceguard 構成のもうひとつの使い方は、プライマリ側を従来どおりの「物理マシン＋Serviceguard」という形態にし、スタンバイ側を「ゲスト OS＋Serviceguard」とすることで、それらの間でフェイルオーバーさせる構成である。ちなみに、物理マシンの代わりに nPar もしくは vPar を選択することも可能だ。この場合、スタンバイ側には複数のゲスト OS を集約できるので、物理マシン 1 台を数多くのプライマリ機に対するスタンバイ機として用いることが可能だ。これにより、これまではハードウェアのコスト増などを理由に Serviceguard クラスターの構築が敬遠されていた用途でも、少ないコストで敷居の低い導入が可能になるはずだ。

「ホスト OS＋Serviceguard」構成の使い方

Integrity VM と Serviceguard を組み合わせるもうひとつの方法は、「ホスト OS＋Serviceguard」構成である。

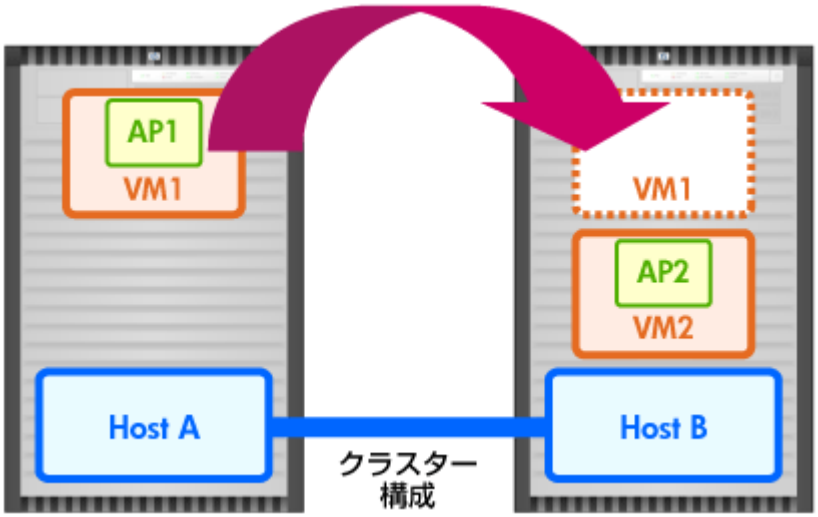

図 3：ホスト OS 内で Serviceguard を使う

この場合は、ゲスト OS の VM プロセスそのものを Serviceguard のパッケージとして構成する。これにより、物理マシンの障害やゲスト OS の VM 障害を検出して、スタンバイ側のホスト OS へ VM のフェイルオーバーが可能だ。この構成のメリットは、個々のアプリケーションについて個別にパッケージを設定する必要がなく、Serviceguard 未導入の既存のアプリケーションに対しても VM まるごと Serviceguard クラスターを構成可能な点だ。

ゲスト OS 内のアプリ監視に対応

ただし、この「ホスト OS＋Serviceguard」構成にはこれまで弱点があった。それは、ゲスト OS の VM が監視対象となるため、その内部で動作する個々のアプリケーションの障害を検知できないという点である。そこで、最新バージョンである Serviceguard 11.19 では、Integrity VM 4.1 との組み合わせ時にゲスト OS 内アプリケーションの監視が可能になった。

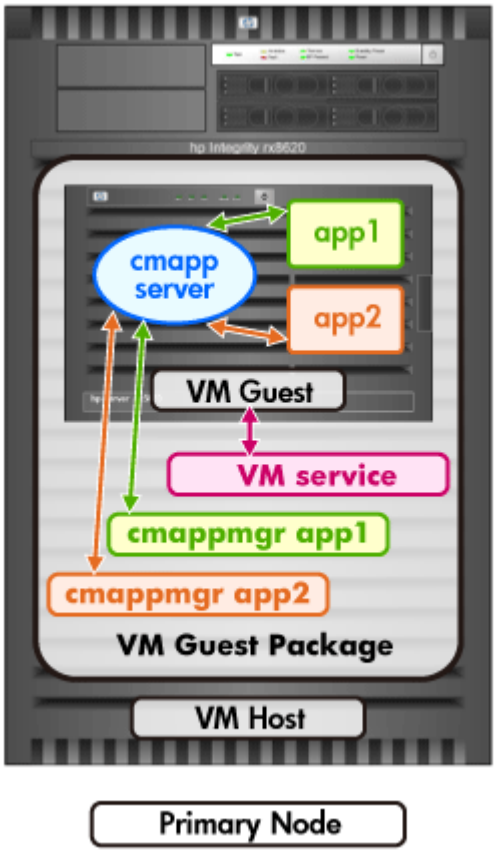

図 4：ゲスト OS 内アプリの監視メカニズム

この機能では、「Application Server」と呼ばれる監視プロセスがゲスト OS 内で 1 つ動作し、個々のアプリケーションを監視する。一方、ホスト OS 上では、個々のアプリケーションに対応する「Application Manager」と呼ばれる複数のプロセスが稼働し、Application Server との SSL 接続を通じて各アプリケーションの状況を監視する。これらの Application Manager がゲスト OS パッケージのサービスとして動作することで、万が一ゲスト OS 内のアプリケーションがダウンした場合は、ホスト OS 側の Serviceguard のフェイルオーバーが起動するという仕組みである。この監視機能は HP-UX、Linux、Windows いずれのゲスト OS にも対応可能だ。

ちなみに、Serviceguard 利用者の多くが用いている ECMT（Enterprise Cluster Master Toolkit）も今回このゲスト OS 監視機能に対応し、ECMT によって構成された Apache Web サーバーおよび Oracle データベースのパッケージ監視が可能になっている。

オンラインマイグレーション＋フェイルオーバーを実現

Serviceguard 11.19 では、もうひとつ新機能として、「Integrity VM によるオンラインマイグレーション」と「Serviceguard によるフェイルオーバー」間の連係機能が追加された。具体的には、「hpvmsg_move」と呼ばれるスクリプトが用意され、これを実行することで Serviceguard のパッケージとして構成されたゲスト OS に対して以下の一連の処理を実行できる。

1. ゲスト OS の VM パッケージのフェイルオーバーを停止
2. オンラインマイグレーションを実行
3. VM パッケージのフェイルオーバーを有効化

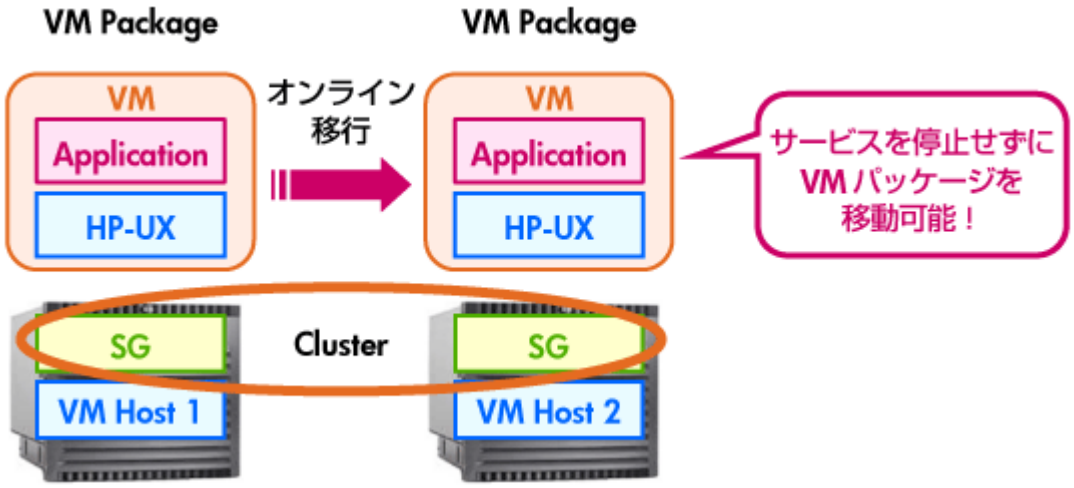

図 5：Serviceguard とオンラインマイグレーションの連携

前回説明したとおり、オンラインマイグレーションを用いることで、ゲスト OS 上で稼働中のサービスを止めずに別の物理マシンへの移行が可能となる。よって、メンテナンスや負荷分散のためのゲスト OS 移動などにオンラインマイグレーションを有効活用できる。

一方、今回の新機能を用いることで、Serviceguard によるフェイルオーバーも同じ構成で利用可能となる。よって、突発的なハードウェアやソフトウェア障害に対する保護をはじめ、オンラインマイグレーションを実行できないケースでの計画的なゲスト OS 移動が可能だ。結果として、「オンラインマイグレーションによる計画停止の排除」と、「Serviceguard による計画外停止の最小化」の両方のメリットが得られるのである。

余裕のある物理マシンを選んでフェイルオーバー

Serviceguard 11.19 の 3 つめの新機能は、フェイルオーバー時に「リソース的に余裕のあるノード」を選んで移行する機能である。具体的には、新規追加されたパラメータ「CAPACITY_NAME」および「CAPACITY_VALUE」をパッケージ内で設定することで、例えば「個々の物理マシンごとに許容可能な VM 数」を定義しておき、VM 数に空きのある物理マシンを選んで移行するといった動作が可能になる。

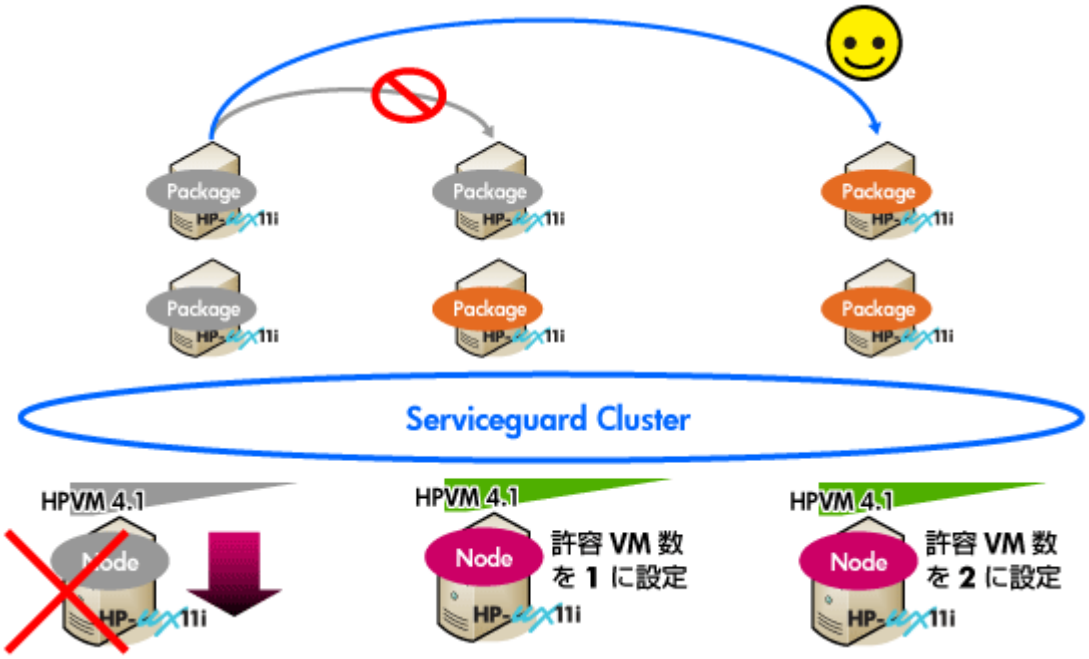

図 6：CAPACITY_NAME / CAPACITY_VALUE を用いたフェイルオーバーの制御

この「CAPACITY_NAME / CAPACITY_VALUE」パラメータで扱えるのは VM 数に限らず、例えばメモリ容量やディスク容量など任意のリソースを表現でき、個々の物理サーバーが有するさまざまなリソース状況に応じてゲスト OS のフェイルオーバー先の決定をきめ細かにコントロール可能となっている。

以上、今回は Integrity VM と Serviceguard の組み合わせ方法について説明した。本連載は今回で最終回となる。今まで 6 回にわたりご愛読いただいたことに感謝の意を表したい。

HP-UX
www.hpe.com/jp/hpux

Hewlett Packard Enterprise