インターシステムズ テクノロジガイド

One Memorial Drive, Cambridge, MA 02142, USA
Tel：+1.617.621.0600 | Fax：+1.617.494.1631
http://www.intersystems.com

高可用性手法

InterSystems Caché、Ensemble、HealthShare Foundation のための高可用性手法

Vik Nagjee、プロダクトマネージャ
Ray Fucillo、ソリューションエンジニア
Mark Bolinsky、テクノロジアーキテクト

2012 年 10 月 12 日（金）

はじめに

このドキュメントでは、InterSystems Caché、Ensemble、および HealthShare Foundation と組み合せて使用できるさまざまな高可用性（HA）手法の概要について説明します。さらに、発生する可能性があるさまざまなシステム機能の停止の概要に加えて、それぞれの配置における適切な手法の選択を可能にするために、特定の停止が各手法でどのように処理されるのかを示します。

このドキュメントで説明する手法は、オペレーティングシステムフェイルオーバークラスタ、仮想化ベース HA、Caché データベースミラーリングの 3 つの異なる HA テクノロジに基づいています。次の表 1 に、この 3 つのテクノロジ間の主な相違点を示します。

	Caché データベース ミラーリング	オペレーティングシステム フェイルオーバー クラスタリング	仮想化高可用性
マシンの電力損失 またはクラッシュ後の フェイルオーバー	デフォルトでは、（クラッシュ、電力損失、ネットワーク損失によって）プライマリマシンにアクセスできなければ自動フェイルオーバーはなし。それ以外の場合は慎重な計画および構成が必要となる。	マシンの障害をシームレスに処理する。	物理マシンおよび仮想マシンの障害をシームレスに処理する。
ストレージの障害および破損からの保護	組み込みレプリケーションがストレージ障害による損失を防ぎ、論理レプリケーションがさまざまなタイプの破損の進行を回避する。	共有ストレージデバイスに依存しているため、障害は壊滅的となる。ストレージレベルの冗長性オプションはあるが、これにより一部の破損が進む可能性がある。	共有ストレージデバイスに依存しているため、障害は壊滅的となる。ストレージレベルの冗長性オプションはあるが、これにより一部の破損が進む可能性がある。
Caché のシャットダウン、ハング、またはクラッシュ後のフェイルオーバー	Caché のシャットダウンまたはハングをすぐに検出し、フェイルオーバーを行う。	Caché の停止後、フェイルオーバーを行うように構成可能。	Caché の停止後、フェイルオーバーを行うように構成可能。
Caché のアップグレード	最小限のダウンタイムでの Cachéアップグレードが可能。*	Caché アップグレードにはダウンタイムが必要。	Caché アップグレードにはダウンタイムが必要。
アプリケーションの 平均リカバリ時間	フェイルオーバー時間は通常数秒間。	フェイルオーバー時間は数分間。	フェイルオーバー時間は数分間。
外部ファイル同期	データベースのみ自動的にレプリケートされる。外部ファイルはサードパーティのソリューションが必要。	すべてのファイルが両方のノードで使用可能。	フェイルオーバー後にすべてのファイルが使用可能。

表 1：一般的な機能の比較（ミラーリング対フェイルオーバークラスタリング）

* アプリケーションコード、ルーチン、およびクラスを、アプリケーションデータを含むデータベースとは別のデータベースに配置する構成にする必要があります。

オペレーティングシステムフェイルオーバークラスタリング

HA を実現するには、オペレーティングシステムレベルで提供されるフェイルオーバーソリューションを使用することが非常に一般的な方法です。そのようなソリューションの例はすべてのプラットフォームに存在し、Microsoft Windows クラスタ、HP Serviceguard、Veritas Cluster Server、および IBM SystemMirror（PowerHA）に加えて、Red Hat や SUSE Linux の各クラスタリングパッケージなどがあります。プラットフォーム間で構成の仕様は若干異なる場合がありますが、そのモデルは一般的に同じで、共有ストレージデバイス（通常は SAN または iSCSI ターゲット）と共有 IP アドレスを持つ同一の 2 つのサーバで構成され、一方がプロダクションワークロードをアクティブに処理し、他方が障害に備えてスタンバイします。アクティブシステムが停止すると、フェイルオーバーテクノロジにより、共有ディスクと共有 IP アドレスのコントロールがスタンバイノードに移され、Caché を含むアプリケーションサービスが起動されます。

Caché は、これらのフェイルオーバーソリューションと容易に統合できるように設計されています。Caché のプロダクションインスタンスは、フェイルオーバークラスタの両方のメンバが認識できるよう共有ストレージデバイスにインストールされ、フェイルオーバーの一部として自動的に起動されるようフェイルオーバークラスタ構成に追加されます。フェイルオーバーにおいて新たにアクティブになったノードで Caché が起動されると、Caché は自動的に WIJ およびジャーナルファイル（これらも共有ストレージデバイス上にあります）から通常のスタートアップリカバリを実行します。Caché が障害のあった元のノードで再起動されただけであるかのように、データの一貫性が維持されます。

長所：

- ↑ マシンの障害をシームレスに処理する。
- ↑ HA の選択肢として最も一般的。
- ↑ OS またはサードパーティベンダにより、サポートされているすべてのプラットフォームで利用可能。
- ↑ すべてのファイル（データベースおよび外部）が両方のノードで使用可能。

短所：

- ↓ ストレージ障害は壊滅的となる。
- ↓ アップグレードにはダウンタイムが必要。
- ↓ アプリケーションの平均リカバリ時間が分単位となる可能性がある。

「Caché 高可用性ガイド」の付録では、より一般的なOSフェイルオーバークラスタと共に正しくCachéを構成する方法を詳細に説明しています。

仮想化ベースの高可用性

VMware vSphere ESX/ESXi などの仮想化テクノロジは高可用性機能を提供します。このテクノロジは通常、物理ハードウェアとそこで実行されるゲストオペレーティングシステム全体の状態および実行可能性を監視します。障害時には、仮想化 HA ソフトウェアが、存続している代替ハードウェアで、障害の起きた仮想マシンを自動的に再起動します。Caché は再起動されると、自動的に WIJ およびジャーナルファイルから通常のスタートアップリカバリを実行します。Caché が障害のあった元のノードで再起動されただけであるかのように、データの一貫性が維持されます。

さらに、ゲストオペレーティングシステムを仮想環境内の別のサーバに再配置することができるため、メンテナンスの目的で、ダウンタイムを必要とせずに仮想マシンを代替の物理インフラストラクチャにすることが可能です。この機能は、VMware vMotion、IBM Live Partition Mobility、HP Live VM Migration などとして利用可能です。

長所：

- ↑ マシンの障害をシームレスに処理する。
- ↑ 仮想環境での HA の選択肢として最も一般的。
- ↑ フェイルオーバー後にすべてのファイルが使用可能。
- ↑ 計画的な物理ハードウェアメンテナンスに、ほとんどまたはまったくアプリケーションダウンタイムを必要としない。

短所：

- ↓ ストレージ障害は壊滅的となる。
- ↓ ソフトウェアアップグレードにはダウンタイムが必要。
- ↓ 計画外のハードウェア障害の場合、アプリケーションの平均リカバリ時間が分単位となる可能性がある。

仮想環境で高可用性を効果的にサポートするには、適切なインフラストラクチャが必要です。これには、ストレージ、ネットワーキング、およびプロセッサ容量が含まれます。ベストプラクティスについては、仮想化の提供元のドキュメントを参照してください。

CACHÉ データベースミラーリング

ミラーは、フェイルオーバーメンバと呼ばれる、物理的に独立した 2 つの Caché システムで構成されます。ミラーは、2 つのフェイルオーバーメンバの一方に自動的にプライマリのロールを割り当て、他方のメンバは自動的にバックアップシステムになります。データはプライマリからバックアップフェイルオーバーメンバにレプリケートされるため、組み込みのデータ冗長性が提供されます。Caché データベースミラーリング（ミラーリング）は、計画的な停止および計画外の停止に対する 2 つの Caché システム間の信頼できる迅速で堅牢な自動フェイルオーバーの経済的なソリューションを提供することを目的としています。

長所：

- ↑ プライマリで Caché がシャットダウンまたはハングした場合の自動フェイルオーバー。
- ↑ 最小限のダウンタイムで Caché のアップグレードが可能（特定の構成ではダウンタイムゼロとすることも可能）。
- ↑ データレプリケーションがプライマリでのストレージ障害による損失を防ぐ。
- ↑ フェイルオーバー時間は通常数秒間で、アプリケーションの平均リカバリ時間が短い。
- ↑ クラスタリングソリューションほどコストがかからない。
- ↑ 論理データレプリケーションは、物理的な破損が別のシステムに波及することを防止できる。

短所：

- ↓ プライマリマシンにアクセスできなければ（OS のクラッシュ、電力損失、ネットワーク損失などにより）、デフォルト構成は自動的にフェイルオーバーを行わない。
- ↓ 常に自動フェイルオーバーを行えるようにするには、慎重な計画と構成が必要。
- ↓ データベースのみ自動的にレプリケートされる。アプリケーションで必要となる外部ファイル（ファイルストリーム、イメージなど）は、サードパーティのレプリケーションソリューションが必要。
- ↓ セキュリティおよび構成管理は、現在のところ分散的。

ミラーリングフェイルオーバーの手法

ミラーリングを使用することで、さまざまな高可用性ニーズを満たすことができます。このような要求を満たす手法には、ミラーリング設定、ハードウェア構成、データセンター構成、手動手順、および時にはカスタムの^ZMIRROR ルーチンが含まれます。

いずれの場合も、プライマリとして処理を引き継ぐには、ソフトウェアによって自動的に、または人が介入して、プライマリフェイルオーバーメンバがダウンしたこと、およびプライマリフェイルオーバーメンバが永続的にコミットしてきたすべてのジャーナルデータがバックアップフェイルオーバーメンバにあることを確認する必要があります。ミラーリングには ISCAgent という実行可能プログラムが用意されており、すべてのミラーメンバで実行して、このような判断に使用することができます。ミラーリングでは、オプションで、プライマリフェイルオーバーメンバがダウンしているかどうかを判断する代替メカニズムを作成することもできます。

このセクションの残りの部分では、さまざまなミラーリングフェイルオーバー手法について説明します。前述した一般的なミラーリングの長所と短所は、各フェイルオーバー手法に当てはまります。各手法に固有の長所と短所を、以下に個別に示します。

ISCAGENT のみでのフェイルオーバー（デフォルト構成）

バックアップミラーメンバは、プライマリの障害を検出すると、プライマリマシンの ISCAgent にアクセスを試みます。ISCAgent にアクセスできる場合、バックアップはプライマリがダウンしていることを確認するか、プライマリが応答しなければ強制終了し、状況の確認に必要なジャーナル情報をすべてダウンロードして、プライマリとして支障なく処理を引き継ぎます。

ISCAgentにアクセスできない場合（プライマリサーバがダウンしているなど）、フェイルオーバーは発生しません。もちろん、管理者が手動で、プライマリがダウンしていること、および必要なジャーナルデータがバックアップにあることを確認し、フェイルオーバーを開始することは可能です。プライマリの計画外の停止後の手動フェイルオーバー方法については、"付録D"を参照してください。

長所：

- ↑ 実装が最も容易なミラーリングフェイルオーバー手法（デフォルトの構成）。
- ↑ 完全に安全なフェイルオーバーで、スプリットブレイン（2つのサーバがどちらもプライマリとして機能すること）のリスクがない。
- ↑ 特別なハードウェアまたはソフトウェアが必要ない。
- ↑ バックアップメンバから ISCAgent にアクセスできる限りは、Caché のシャットダウン、応答しない Caché インスタンス、および Caché の動作を妨げるさまざまなハードウェアまたはソフトウェア障害からの迅速なフェイルオーバーが可能。
- ↑ フェイルオーバーメンバは別々のデータセンターに配置できるため、達成する HA および DR の目標を 2 つのサーバのみに限定することが可能（許容可能遅延はアプリケーションによって異なる）。

短所：

- ↓ ホスト障害など、ISCAgent がアクセス不能な状態となる障害の発生後は、自動フェイルオーバーは行われない。
- ↓ プライマリホストが使用できなくなると、手動フェイルオーバーを支障なく開始できるかどうかを検証するために必要なジャーナル情報がすべてバックアップに存在するかどうかを判断することが困難になる可能性がある。

この手法を実施するには、「フェイルオーバーにエージェントの通信が必要」の構成設定を「はい」（デフォルト）のままにします。

カスタムソリューションでのフェイルオーバー：信頼性のあるネットワークの PING

バックアップミラーメンバは、プライマリの障害を検出すると、プライマリマシンの ISCAgent にアクセスを試みます。ISCAgent にアクセスできる場合、バックアップはプライマリがダウンしていることを確認するか、プライマリが応答しなければ強制終了し、状況の確認に必要なジャーナル情報をすべてダウンロードして、プライマリとして支障なく処理を引き継ぎます。

ISCAgent にアクセスできない場合（プライマリサーバがダウンしているなど）は、パブリックネットワークおよびプライベートネットワークでのネットワーク ping を使用して、プライマリサーバのステータスを判断します（これには、`$$IsOtherNodeDown^ZMIRROR()`に実装されるカスタムプログラミングが必要です）。パブリックネットワークまたはプライベートネットワークのいずれかで、プライマリが ping に応答しない場合、バックアップはプライマリがダウンしていると見なし、フェイルオーバープロセスを続行します。この手法では、スプリットブレイン（2 つのサーバが同時にプライマリとして機能すること）を回避するため、2 つのフェイルオーバーメンバ間のネットワーキングに、冗長性、信頼性、および高可用性が必要です。

長所：

↑ サーバ/ホスト障害の後の迅速なフェイルオーバーが可能。

短所：

↓ カスタムの^ZMIRROR ルーチンの実装が必要（インターシステムズによるサンプルの提供あり）。

↓ スプリットブレイン（2 つのサーバがどちらもプライマリとして機能すること）を回避するため、非常に堅牢なネットワーキングを提供する特殊なネットワーキングハードウェア構成が必要。

↓ ネットワーク分離のリスクを最小限に抑えるため、フェイルオーバーマシンは同じデータセンターに配置することが推奨される。

この手法を実施するには：

1. プライマリフェイルオーバーメンバとバックアップフェイルオーバーメンバ間で非常に信頼性のあるネットワークを提供するハードウェア構成を作成します。2 つのフェイルオーバーメンバ間の信頼性のあるネットワーク構成の例は、"付録A"を参照してください。
2. "付録C"で提供されている^ZMIRRORルーチンのサンプルをインポートし、カスタマイズします。
3. 「フェイルオーバーにエージェントの通信が必要」を「いいえ」に設定します。
4. `$$IsOtherNodeDown^ZMIRROR`メカニズムが機能するために十分な時間を許容するよう、障害タイムアウトの限度を調整します。例えば、フェイルオーバーのテストの際、バックアップフェイルオーバーメンバのcconsole.logファイルに`Mirror recovery time of 7.101 seconds exceeded trouble timeout of 6 seconds. Restarting`のようなメッセージが表示されることがあります。このような場合は、トラブルタイムアウト制限を 8 秒に上げることを検討します。

上記のメカニズムでプライマリがダウンしていると断定できない場合には、プライマリは稼働していると仮定するのが唯一の無難な方法です。もちろん、管理者が手動で、プライマリがダウンしていること、および必要なジャーナルデータがバックアップにあることを確認し、フェイルオーバーを開始することは可能です。プライマリの計画外の停止後の手動フェイルオーバー方法については、"付録D"を参照してください。

カスタムソリューションでのフェイルオーバー：その他の手法

バックアップミラーメンバは、プライマリの障害を検出すると、プライマリマシンの ISCAgent にアクセスを試みます。ISCAgent にアクセスできる場合、バックアップはプライマリがダウンしていることを確認するか、プライマリが応答しなければ強制終了し、状況の確認に必要なジャーナル情報をすべてダウンロードして、プライマリとして支障なく処理を引き継ぎます。

ISCAgent にアクセスできない場合（プライマリサーバがダウンしているなど）、その配置用にカスタマイズされたメカニズムを使用してプライマリがダウンしているかどうかを判断する代替プロシージャを実行します。

長所：

↑ 独自の環境に合わせてカスタム調整されたソリューションが可能。

短所：

↓ ISCAgent にアクセスできない場合にプライマリがダウンしていると判断するカスタムメカニズムを設計し、そのメカニズムを使用するように `$$IsOtherNodeDown^ZMIRROR()` をプログラミングする必要がある。

↓ 特殊なハードウェアが必要となる場合がある。

この手法を実施するには：

1. プライマリがダウンしていると断定する堅牢なメカニズムを提供する構成を作成します。このような構成を作成するために使用されるアーキテクチャは、2 つのフェイルオーバーメンバの位置とその距離によって異なります。
2. プライマリがダウンしているかどうかを判断するカスタムメカニズムを使用するように、`$$IsOtherNodeDown^ZMIRROR()` をプログラムします。選択したメカニズムでプライマリがダウンしていると断定できない場合、プライマリは稼働していると見なす必要があります。
3. 「フェイルオーバーにエージェントの通信が必要」を「いいえ」に設定します。
4. `$$IsOtherNodeDown^ZMIRROR()` メカニズムが機能するために十分な時間を許容するよう、障害タイムアウトの限度を調整します。

管理者が手動で、プライマリがダウンしていること、および必要なジャーナルデータがバックアップにあることを確認し、フェイルオーバーを開始することは可能です。プライマリの計画外の停止後の手動フェイルオーバー方法については、"付録D"を参照してください。

ハイブリッド HA 手法

データベースミラーリングは、フェイルオーバークラスタリングまたは仮想化 HA と組み合せて使用することで、計画的な停止および計画外の停止に対する非常に堅牢な高可用性手法を提供します。

フェイルオーバークラスタリングまたは仮想化 HA は、マシンまたは OS レベルの障害（計画外の停止）に対する自動フェイルオーバーを提供します。データベースミラーリングは、Caché の停止（計画的または計画外）に対する自動フェイルオーバーを提供し、その組み込みデータレプリケーションはストレージ障害による損失を防ぎます。

長所：

↑ ハードウェア、OS、およびデータベースの計画的停止および計画外停止に対する安全な自動フェイルオーバーを提供する。
↑ OS またはサードパーティベンダにより、サポートされているすべてのプラットフォームで利用可能。
↑ 計画的な物理ハードウェアメンテナンスに、ほとんどまたはまったくアプリケーションダウンタイムを必要としない。
↑ ミラーリングレプリケーションがストレージ障害による損失を防ぐ。

短所：

↓ 非常に複雑である。
↓ 実施するにはコストがかかる可能性あり。
↓ 管理および習得するテクノロジが複数ある。

この手法を実施するには、付録Aの"ハイブリッドHAソリューション"で構成オプションを参照してください。「フェイルオーバーにエージェントの通信が必要」ミラーリング構成を「はい」（デフォルト）のままにする必要があることに注意してください。

一般的なシステム停止

システム停止には、大きく分けて 2 つのカテゴリがあります。計画的な停止と計画外の停止です。どの HA 手法を選択するかによって、特定のタイプの停止における自動フェイルオーバーの可能性がほぼ決まります。表 1 は、さまざまな停止のタイプをまとめ、その特定の停止に対して各 HA 手法が自動的にフェイルオーバーを行うかどうかを示しています。

停止タイプおよび HA 手法ごとの自動フェイルオーバー		OS フェイルオーバークラスタ	仮想化 HA	ミラーリング：ISCAgent のみによるフェイルオーバー	ミラーリング：カスタムソリューションによるフェイルオーバー - 信頼性のあるネットワークの Ping	ミラーリング：カスタムソリューションによるフェイルオーバー - その他の手法	ハイブリッド手法
計画的停止	Caché のアップグレード	×	×	○	○	○	○
	OS のアップグレード	○	×	○	○	○	○
	アプリケーションのアップグレード	×	×	△	△	△	△
	サーバのアップグレード/メンテナンス	○	○	○	○	○	○
	ストレージのアップグレード/メンテナンス	△	△	○	○	○	○
計画外停止	Caché のハング、クラッシュ、またはシャットダウン	△	△	○	○	○	○
	プライマリでのすべてのネットワーク障害	○	○	×	N/A†	△	○
	プライマリでのパブリックネットワークインタフェース障害	○	○	△	△	△	○
	サーバ（ホスト）障害、クラッシュ、または電力損失	○	○	×	○	△	○
	ストレージ障害	△	△	△	△	△	△

表 2：停止タイプおよび HA 手法ごとの自動フェイルオーバー

以降のセクションでは、表 2 の結果についての根拠を説明します。

† これは、スプリットブレイン（2 つのサーバがどちらもプライマリとして機能すること）となる可能性がある特殊なケースです。以下の詳細を参照してください。

計画的停止のタイプ

- **Caché のアップグレード**：OS フェイルオーバークラスタおよび仮想化 HA では、構成内に Caché のインスタンスが 1 つしかないため、Caché のアップグレードにはダウンタイムが必要です。一方ミラーリングでは、まずバックアップフェイルオーバーメンバの Caché をアップグレードし、フェイルオーバーを行って、もう一方のノードをアップグレードすることで、互換性のあるバージョン間のローリングアップグレードを実行できます。ハイブリッド手法にはミラーリングが含まれるため、この HA 手法にも同じ長所が当てはまります。
- **OS のアップグレード**：仮想化 HA を除き、概説したすべての HA 手法では、一般的にローリングアップグレードオプションが提供されます。仮想化 HA では、ゲスト OS のインスタンスが 1 つしかないため、ダウンタイムが必要となります。
- **アプリケーションのアップグレード**：OS フェイルオーバークラスタおよび仮想化 HA では、構成内に Caché のプロダクションインスタンスが 1 つしかないため、アプリケーションのアップグレードにはダウンタイムが必要です。ミラーリングでは、（アップグレードの最中および前後における）データ変換が不要であり、アプリケーションコードとデータは別のデータベースに格納される（ベストプラクティス）とすれば、アプリケーションのアップグレードでのダウンタイムを最小限に抑えることもできます。ハイブリッド手法にはミラーリングが含まれるため、この HA 手法にも同じ長所が当てはまります。
- **サーバのアップグレード/メンテナンス**：概説したすべての HA 手法では、一般的にローリングメンテナンスおよびアップグレードオプションが提供されます。
- **ストレージのアップグレード/メンテナンス**：OSフェイルオーバークラスタおよび仮想化HAでは、構成内に共有ストレージデバイスが 1 つしかないため、ストレージのアップグレードまたはメンテナンスにはダウンタイムが [‡] 必要となる場合があります。一方ミラーリングでは、通常独立したストレージデバイスを使用するため、ローリングストレージアップグレードまたはメンテナンスが可能です。ハイブリッド手法にはミラーリングが含まれるため、このHA手法にも同じ長所が当てはまります。

計画外停止のタイプ

- **Caché のハング、クラッシュ、またはシャットダウン**：OS フェイルオーバークラスタおよび仮想化 HA では、Caché の障害時にフェイルオーバーを発生させるように構成できる監視スクリプトが提供されます。ミラーリングでは、このような状況を直ちに検出し、自動フェイルオーバーをトリガします。ハイブリッド手法にはミラーリングが含まれるため、この HA 手法にも同じ長所が当てはまります。
- **プライマリでのすべてのネットワーク障害**：OS フェイルオーバークラスタおよび仮想化 HA では、通常ネットワーク損失を検出し、自動的にフェイルオーバーをトリガします。ミラーリングの場合は、展開されている手法によって動作が異なります。
    - ISCAgent のみによるフェイルオーバー：障害のあったノードの ISCAgent にアクセスできないため、ミラーリングでは自動的にフェイルオーバーを行いません。
    - 信頼性のあるネットワークの Ping によるフェイルオーバー：この手法は、このタイプの障害が発生しないようにするため、100%信頼性/冗長性のあるネットワークがあることが前提となります。このタイプの障害が発生した場合、`$$IsOtherNodeDown^ZMIRROR()`プロシージャにより、ミラーリングで自動的にフェイルオーバーを行うことが可能となりますが、スプリットブレイン（2 つのサーバがどちらもプライマリとして機能すること）が発生します。
    - カスタムソリューションでのフェイルオーバー：自動フェイルオーバーが成功するかどうかは、`$$IsOtherNodeDown^ZMIRROR()`のカスタム実装に依存します。
    - ハイブリッド手法：ハイブリッド手法には OS フェイルオーバークラスタ/仮想化ベース HA が含まれるため、このような障害では自動フェイルオーバーが発生します。
- **プライマリでのパブリックネットワークインタフェース障害**：OS フェイルオーバークラスタおよび仮想化 HA では、通常ネットワーク損失を検出し、自動的にフェイルオーバーをトリガします。ミラーリングの場合、デフォルトではパブリックネットワークインタフェース障害を検出しません。ただし、プライマリサーバでのパブリックネットワークインタフェース障害の際にフェイルオーバーを発生させるよう、カスタムの監視スクリプトを作成することができます。これは任意のミラーリング手法に適用できますが、2 つのフェイルオーバーメンバ間で別個の（プライベート）ネットワークを構成することが必要になります。ハイ

[‡] 一部の SAN ストレージシステムでは、ダウンタイムなしでストレージのアップグレード/メンテナンスができる無停止アップグレード（NDU）が可能です。

ブリッド手法には OS フェイルオーバークラスタ/仮想化ベース HA が含まれるため、このような障害では自動フェイルオーバーが発生します。

- **サーバ（ホスト）障害、クラッシュ、または電力損失**：OS フェイルオーバークラスタおよび仮想化 HA では、これらのタイプの障害を検出し、自動的にフェイルオーバーを行います。ミラーリングの場合は、展開されている手法によって動作が異なります。
    - ISCAgent のみによるフェイルオーバー：障害のあったノードの ISCAgent にアクセスできないため、ミラーリングでは自動的にフェイルオーバーを行いません。
    - 信頼性のあるネットワークの Ping によるフェイルオーバー：`$$IsOtherNodeDown^ZMIRROR()`プロシージャが、プライマリがダウンしていることを検出し、ミラーリングで自動的にフェイルオーバーを行うことができます。
    - カスタムソリューションでのフェイルオーバー：自動フェイルオーバーが成功するかどうかは、`$$IsOtherNodeDown^ZMIRROR()`のカスタム実装に依存します。
    - ハイブリッド手法：ハイブリッド手法には OS フェイルオーバークラスタ/仮想化ベース HA が含まれるため、このような障害では自動フェイルオーバーが発生します。
- **ストレージ障害**：ディスクベースのレプリケーションが使用されている場合、OS フェイルオーバークラスタはストレージ障害に耐えられることがあります。それ以外の場合、両方のノードは共有ストレージデバイス上で動作するため、ストレージ障害は壊滅的なものとなります。同様に、仮想化 HA はレプリケートされたストレージ上に配置される場合があります。ミラーリングでは、フェイルオーバーメンバが独立したストレージデバイスを使用するように構成されていれば、ストレージ障害の際に自動的にフェイルオーバーできる場合があります。自動的にフェイルオーバーできるかどうかは、ストレージ障害の性質により異なります。ミラーリングによって提供されるデータレプリケーションは、プロダクションデータをストレージ障害から保護するため、必要に応じて手動でフェイルオーバーを発生させることができます。ハイブリッド手法にはミラーリングが含まれるため、この HA 手法にも同じ長所が当てはまります。

付録 A：信頼性のあるネットワーク構成の例

この付録では、ミラーリングの展開に使用できる、さまざまな信頼性のあるネットワーク構成について説明します。

カスタムソリューションでのミラーリングフェイルオーバー：信頼性のあるネットワークのPingの手法では、スプリットブレイン（2つのサーバがどちらもプライマリとして機能すること）を回避するため、2つのフェイルオーバーメンバ間のネットワーキングに、冗長性、信頼性、および高可用性が必要です。以下に構成例を示します。

図 1 は、2 つのフェイルオーバーメンバがプライベートネットワーク上でのミラー通信用に互いにクロスオーバーケーブルで直接接続され、外部スイッチまたはハブを完全に回避している非常にシンプルな構成を示しています。これにより、2 つのシステム間でのネットワーク関連の障害の可能性が低減されます。このシステムは、パブリック通信用スイッチの外部バンクにも接続されます。この構成は、非常に経済的で設計がシンプルです。

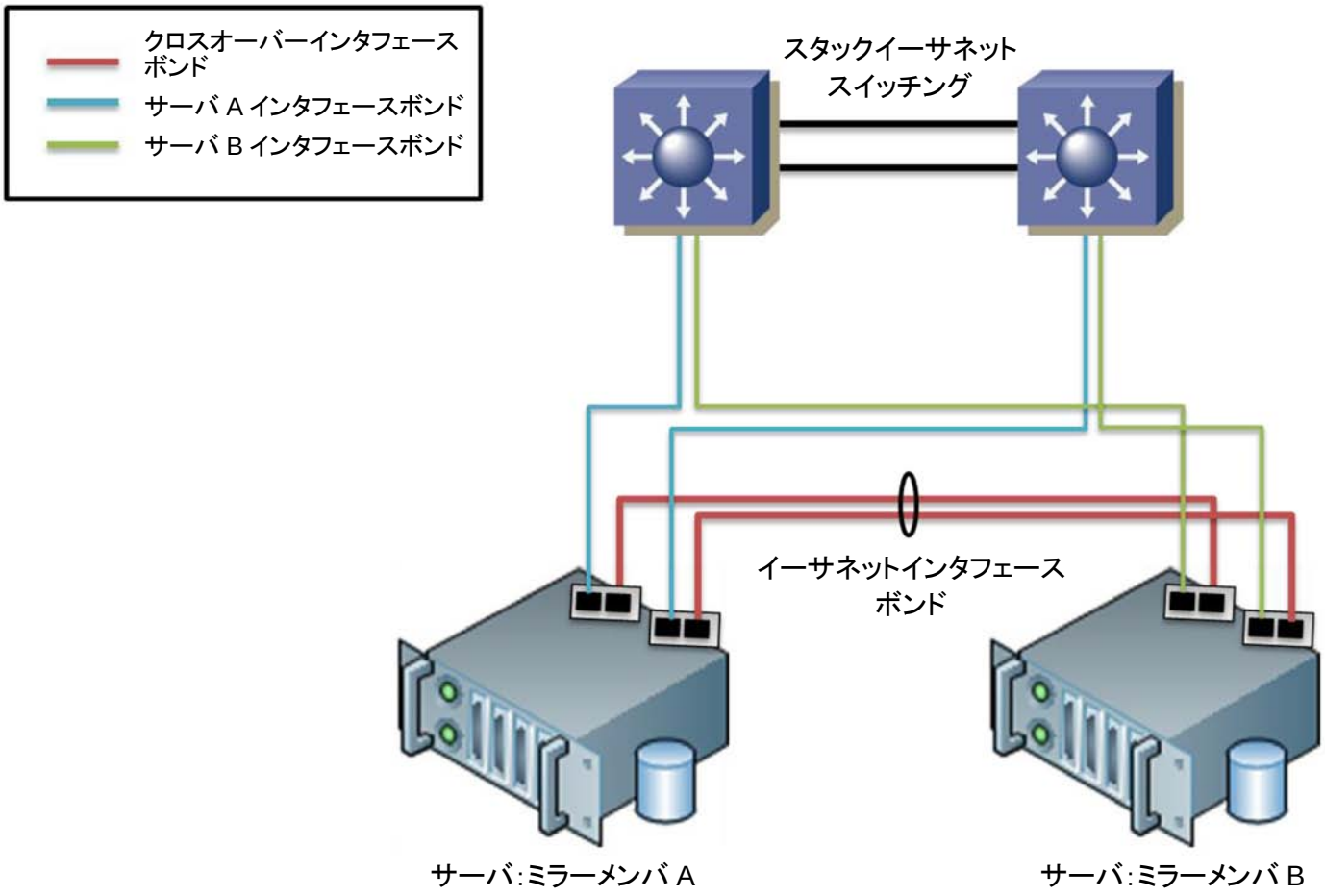

図 1：直接接続の構成例

図 1 の構成例について、以下に注釈を示します。

- 各システムには、2 つのデュアルポート NIC が示されています。これは信頼性および冗長性を目的としており、推奨される構成です。
- クロスオーバーケーブルでボンディングまたはチーミングされたインタフェースポートの使用をサポートするかどうかは、OS および NIC ドライバによって異なります。このタイプのボンディングは、すべてのオペレーティングシステムおよび NIC ドライバでサポートされているわけではありません。互換性については、システムドキュメントを参照してください。

2 つのフェイルオーバーメンバは、内部ストレージ（SAS）で構成されるか、外部ストレージ（SAN、NAS など）を持ちます。ただし、共有 SAN の場合、2 つのシステムが別個の物理スピンドルを使用することを推奨します。

この例は、高可用性ネットワークスイッチングを含めるように拡張することができます。これにより、物理コロケーション要件が緩和されますが、信頼性の高いネットワーキングであることが不可欠です。図 2 は、それぞれ EtherChannel ボンドを使用した可用性の高いスタックスイッチのある 2 つのコンピュータルーム（同じ構内）を使用した構成例を示しています。

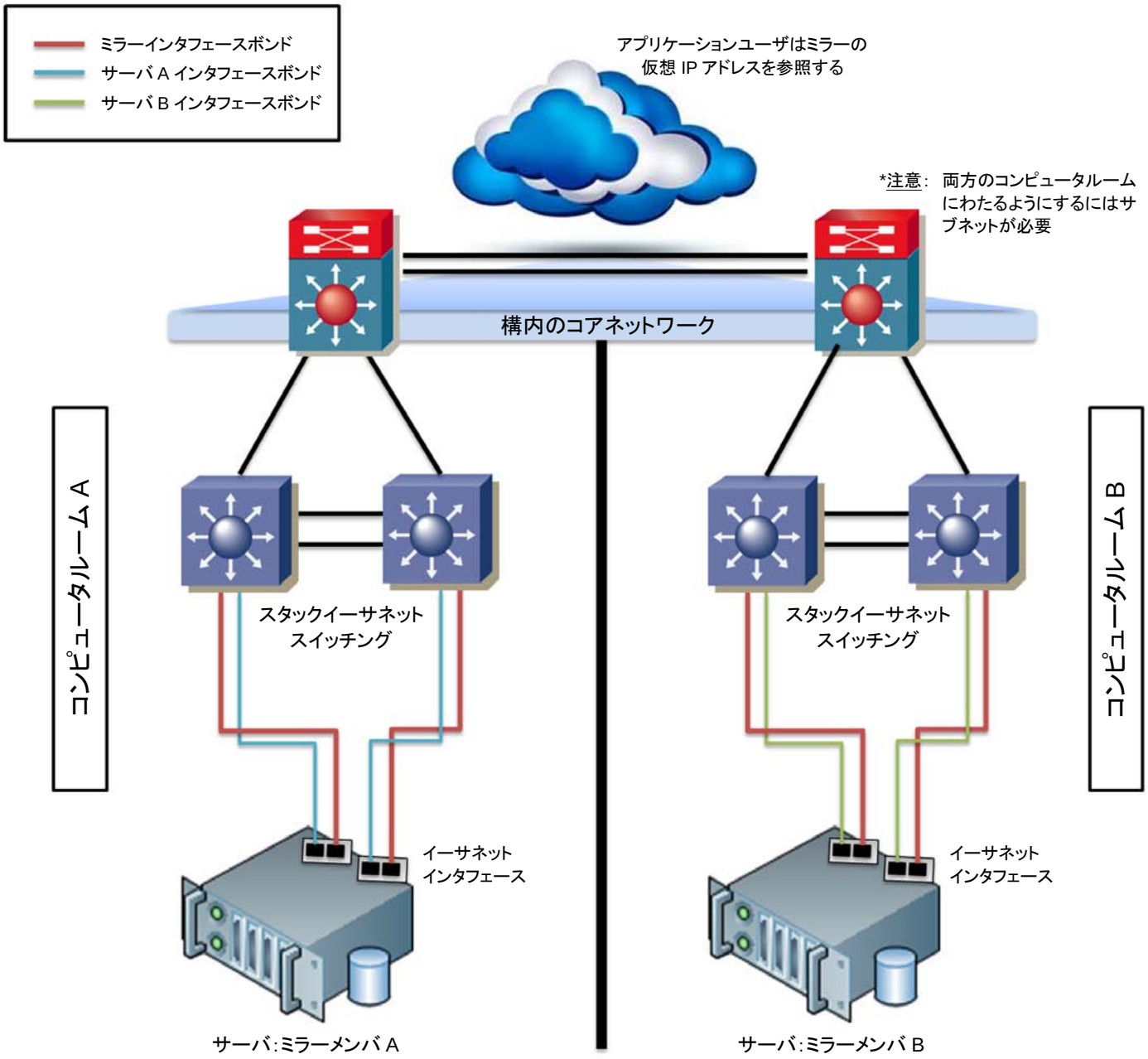

図 2：冗長性のあるスイッチング構成例

付録 B：ハイブリッド HA ソリューション

OS フェイルオーバークラスタおよび仮想化ベース HA のテクノロジでは、システムの停止（電力損失、OS クラッシュ、ネットワーク損失など）に対する自動フェイルオーバーを提供します。ミラーリングでは、レプリケーションに加えて、アップグレード、メンテナンス、Caché の障害など、データベース関連の停止に対する組み込みの自動フェイルオーバーを提供します。ハイブリッド HA ソリューションは、OS フェイルオーバークラスタまたは仮想化ベース HA と Caché データベースミラーリングの両方のタイプの HA テクノロジの長所を活かし、非常にレベルの高い可用性を提供します。ただし、これらの構成は複雑になる場合があり、慎重な計画が必要です。

ミラーリングは、デフォルトモード（「フェイルオーバーにエージェントの通信が必要」=「はい」）で構成し、ハードウェア/サーバレベルの停止（計画外）に対する仮想化HAまたはOSフェイルオーバークラスタに従い、さらにデータベースの停止（計画的または計画外）に対してミラーバックアップメンバに迅速にフェイルオーバーするようにする必要があります。

仮想化とミラーリングのハイブリッド

仮想化の利用には、許容可能な高い可用性を提供するための追加要件が伴います。ゲスト仮想マシンを構成する際、各ミラーメンバにアンチアフィニティルールが定義されている必要があります。これにより、プライマリとバックアップのフェイルオーバーメンバが同じ物理サーバ上で実行されないようにすることができます。

少なくとも 2 つまたは 3 つの物理アダプタにわたる、6 つ以上のイーサネットインタフェースをインストールすることを強く推奨します。ブレードサーバテクノロジでは、VIC（仮想インタフェースカード）または CNA（コンバージドネットワークアダプタ）の使用が一般的で、イーサネット通信とファイバーチャネル通信の両方をサポートしています。各ハードウェア供給業者の仮想インタフェース定義のベストプラクティスを確認してください。

図 3 は、冗長なネットワーキングの 3 ノードVMware ESX[§] クラスタの例を示しています。

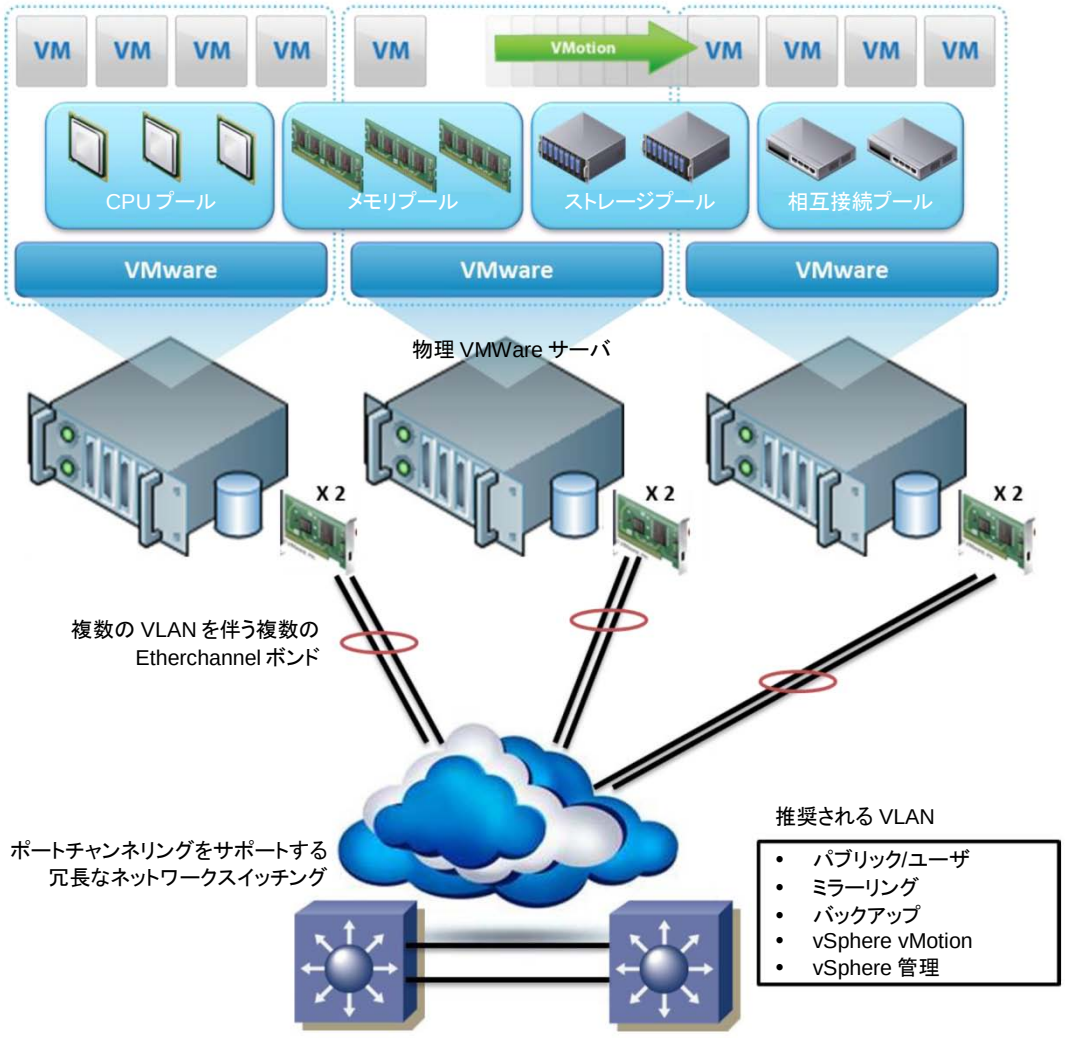

図 3：VMware vSphere を使用した冗長ネットワーキング

[§] VMware は例としてのみ使用されていることに注意してください。自動パーティションまたは仮想マシンモビリティ/再起動を提供する Red Hat RHEV-M など、IBM、HP、Microsoft、および Linux KVM の類似製品のその他の仮想化テクノロジも同様に使用できます。

各ミラーメンバの分離を維持するには、ネットワークにおける冗長性と共に、ストレージに注目することを強く推奨します。分離するには、各ミラーメンバが、少なくとも単一の SAN ストレージアレイ内の別のディスクグループ、理想的には 2 つの別のストレージアレイにある、別個のデータストアを使用する必要があります。図 4 は、仮想化環境のストレージレイヤ分離を示しています。

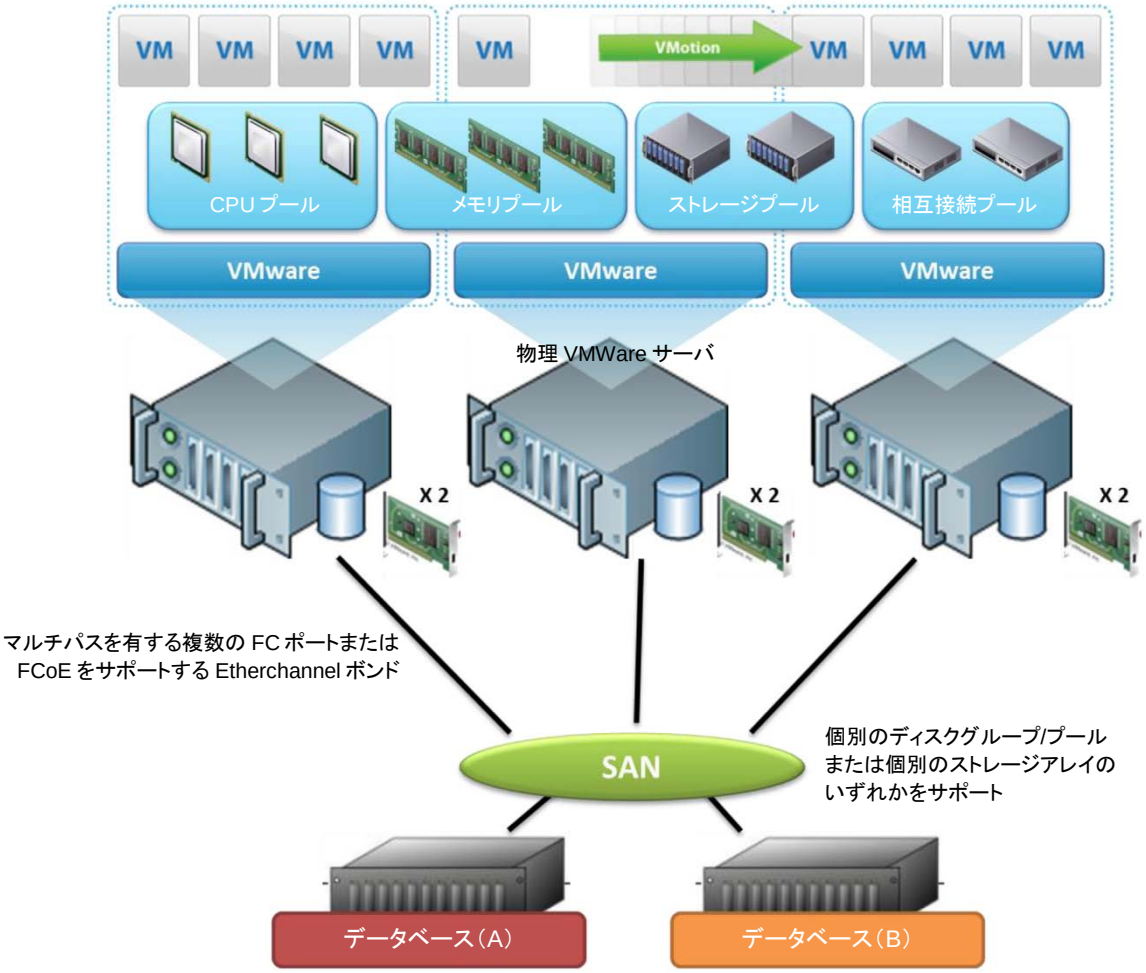

図 4：VMware vSphere を使用した冗長ストレージ

図 5 は、障害のある仮想マシンを他の物理リソースからリカバリするVMware HA** を示しています。ここでは、少なくとも 3 つのサーバが必要です。

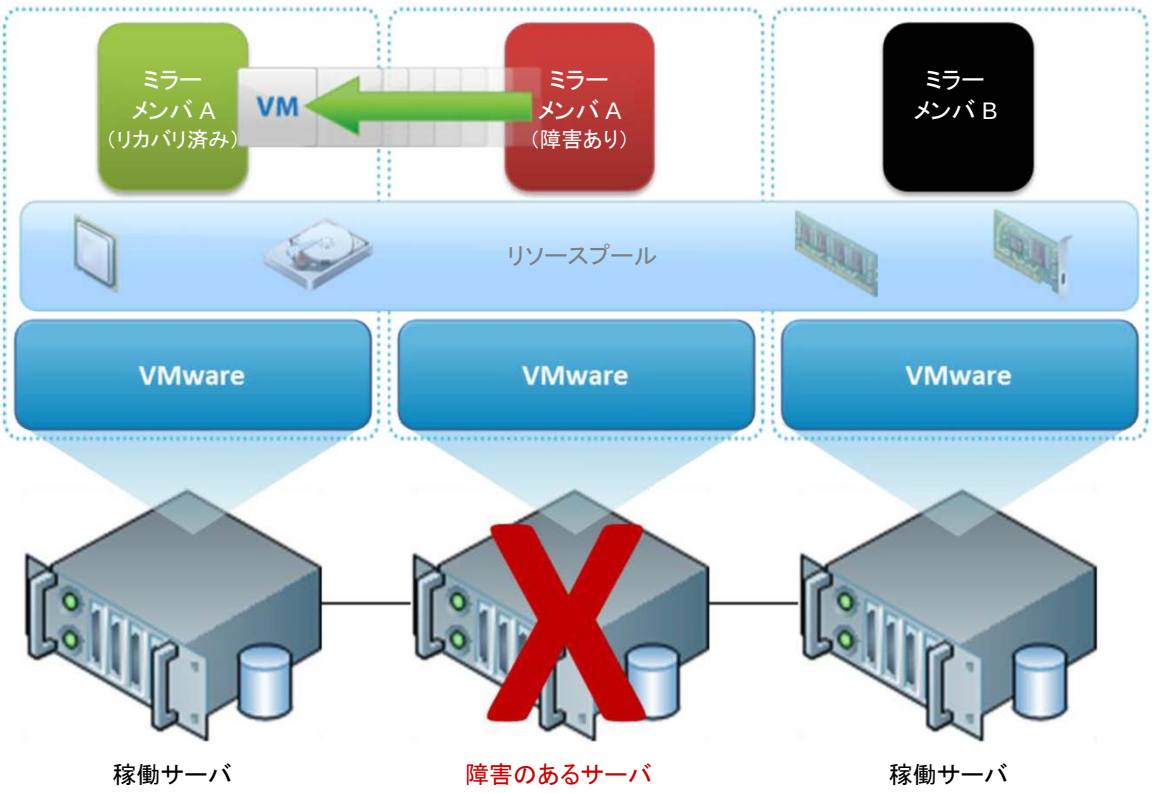

図 5：データベースミラーリングを使用した VMware HA

** VMware は例としてのみ使用されていることに注意してください。自動パーティションまたは仮想マシンモビリティ/再起動を提供する Red Hat RHEV-M など、IBM、HP、Microsoft、および Linux KVM の類似製品のその他の仮想化テクノロジも同様に使用できます。

OS フェイルオーバークラスタリングとミラーリングのハイブリッド

図 6 は、ミラーメンバごとにハードウェア障害から保護するフェイルオーバークラスタリングを示しています。ここで、2 つの 2 ノードフェイルオーバークラスタを配置するには、少なくとも 4 つの物理サーバが必要です。

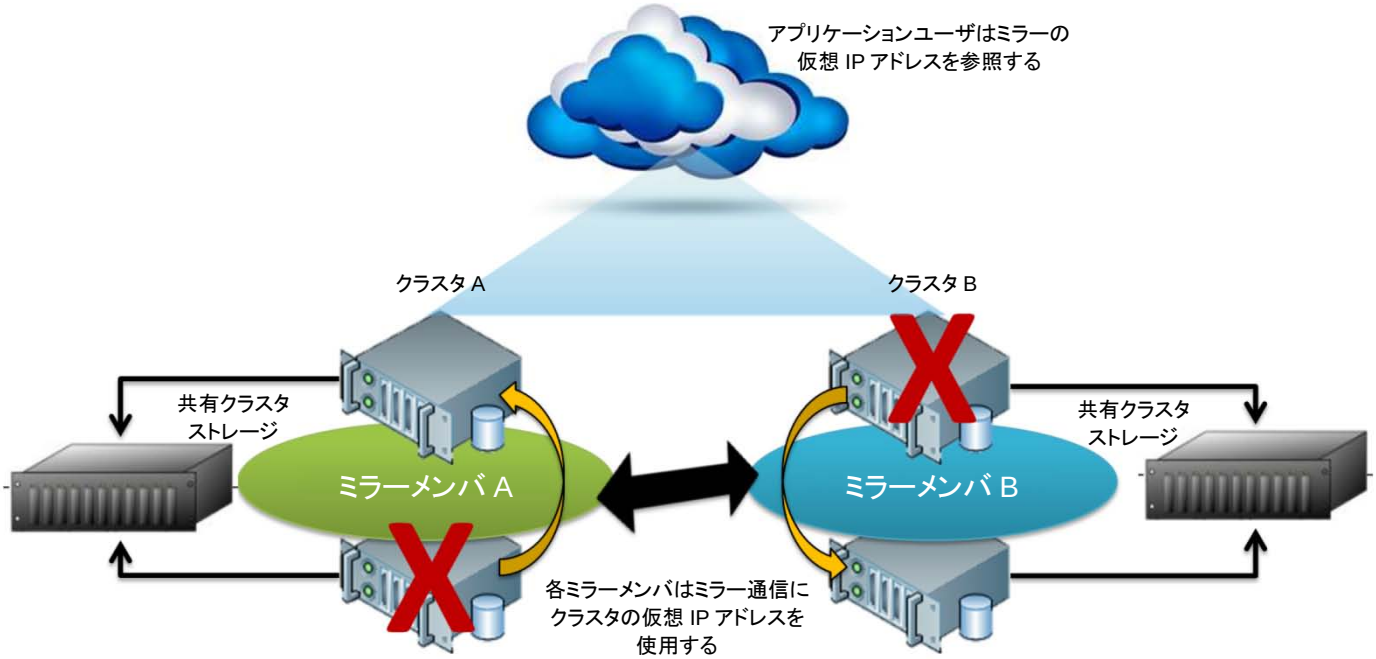

図 6：OS フェイルオーバークラスタリングとミラーリングのハイブリッド

付録 C：信頼性のあるネットワークの PING フェイルオーバーのための^ZMIRROR 例

ミラーリングの信頼性のあるネットワークのPingフェイルオーバー手法を実施できるように、インターシステムズでは^ZMIRRORルーチンのサンプルを提供しています。

最新のサンプルファイルは、以下の URL よりダウンロード頂けます。
http://www.intersystems.co.jp/highavailability/ZMIRROR_jpn.zip

このファイルを自分の環境にダウンロードし、次のようにして%SYS ネームスペースにロードしてください。

```
Do $System.OBJ.Load("<path>/ZMIRROR_RELIABLE_NETWORK_PING.xml","ck")
```

このファイルをロードしたら、環境に合うようにカスタマイズする必要があります。ルーチン自体には、一目瞭然のステップとコメントが含まれています。

必要に応じて、インターシステムズジャパン・サポートセンターにお問い合わせください。

付録 D：プライマリの計画外の停止後の手動フェイルオーバー

この付録では、プライマリの計画外の停止後に、手動でフェイルオーバーを発生させる方法について説明します。これは、ソフトウェア構成だけでは自動で安全にフェイルオーバーできるかどうか判断できない場合に、このドキュメントで説明した手法と組み合せて使用します。

ここでは、プライマリシステムが完全に故障しているものとします。プライマリシステムが稼働している場合は、バックアップシステムが自動的にプライマリの ISCAgent にアクセスし、そのステータスを判断して、場合によっては処理を引き継ぎます。一般に、障害のあるプライマリを単純に再起動できる場合は、以下の手動のフェイルオーバー手順より、その再起動が優先されます。

障害のあるプライマリまたはダウンしたプライマリのジャーナルファイルにアクセスできる場合、プライマリとバックアップ間のデータの不整合を発生させることなく、以下の手順を使用することができます。

1. プライマリマシンがダウンし、この手順の間もダウンしたままであることを確認します。
2. バックアップマシンで Caché をシャットダウンします。
3. 必要なジャーナルファイルをバックアップにコピーします。つまり、バックアップが持っている最終的なミラージャーナルファイルと、それより新しいプライマリが作成したミラージャーナルファイルをコピーし直します。
4. バックアップのミラージャーナルログファイルを削除します。このファイルはインスタンスの mgr（manager）ディレクトリにあり、`mirrorjrn-{MirrorName}.log` という名前が付けられています。これは、後のステップで自動的に再構築されます。
5. バックアップマシンでISCAgentが実行されていることを確認します。
6. バックアップマシンで Caché を起動します。
7. バックアップでプライマリの強制機能を使用してプライマリにします。このオプションは^MIRROR ユーティリティから使用可能です。

障害のあるプライマリまたはダウンしたプライマリのジャーナルファイルにアクセスできない場合、プライマリとバックアップ間のデータの不整合のリスクがある場合があります。以下のすべての状況が確実な場合は、障害以降のプライマリからのすべてのジャーナルデータがバックアップにあるため、手動フェイルオーバーを安全に実行できます。

- プライマリサーバがダウンしている。
- バックアップがプライマリから切断された時間が、プライマリサーバがダウンした時間と一致する。この判断の確実性は、環境と障害のタイプにより異なります。バックアップが切断された時間は、バックアップの cconsole.log ファイルで、`MirrorClient: Primary AckDaemon failed to answer status request.`のようなメッセージを検索することにより確認できます。
- バックアップは、障害を検出した時点でアクティブと見なされていた。アクティブなバックアップとは、状況が確認され、プライマリから同期的にデータを受け取っているバックアップです。これは、バックアップのcconsole.logファイルで、`Failed to contact agent on former primary, can't take over`のようなメッセージを検索することにより確認できます。**注意**：バックアップのcconsole.logファイルに`Non-active Backup is down`のようなメッセージが存在する場合は、障害の検出時にこれがアクティブではなかったことを示します。この場合、バックアップにはプライマリからの必要なデータがすべてあるわけではなく、手動フェイルオーバーの実行は、データの不整合が生じることになるため安全ではないと見なすことができます。

手動フェイルオーバーを続行するには、バックアップでプライマリの強制機能を使用してプライマリにします。このオプションは^MIRROR ユーティリティから使用可能です。

手動フェイルオーバーを続行して、2 つのフェイルオーバーメンバ間にデータの不整合が生じた場合、以下のような結果が予想されます。

- 以前のプライマリでコミットされたがバックアップに転送されていないデータが、アプリケーションで欠落する可能性がある。
- 以前のプライマリが、再起動時にバックアップとして再度ミラーに加わることを拒否する。

- バックアップとして再度ミラーに加わるようにするには、現在の（新しい）プライマリからのデータベースのバックアップを別のシステムにリストアする必要がある。