クラスタ技術に関する調査

Ken Moreau

Solutions Architect, OpenVMS Ambassador, MCSE

概要

このホワイトペーパーでは、HP のサーバ・プラットフォーム上で動作する HP-UX、Linux、NonStop Kernel、OpenVMS、Tru64 UNIX、Windows 2000 の各オペレーティング・システムのクラスタ技術について取り上げます。すべてのクラスタ技術に共通する機能を説明し、各プラットフォームでの共通点と異なる点を示し、業務のニーズに照らしてどのオペレーティング・システムのクラスタ技術を導入するのが適切か、公平に評価する方法を紹介します。なお、パフォーマンス、オペレーティング・システムの基本機能、およびシステムのハードウェアについては説明しません。

このドキュメントの大部分は、HP ETS (Enterprise Technical Symposium) 2002 で行った同じタイトルのプレゼンテーションを基に作成しています。

はじめに

個々のクラスタ・テクノロジは、相互に密接に関連しており、ほとんどの要素が他のすべての要素に影響を与えます。しかし、筆者はこのテーマを 5 つの分野に分けました。

- シングル・システム・ビューとマルチ・システム・ビュー ----- これにより、クラスタ・システムを統合された 1 つのシステムとして管理/運用するか、あるいは個々のクラスタ・メンバ・システムをそれぞれ単独のシステムとして管理/運用するかが決まります。
- クラスタ・ファイル・システム ----- クラスタ環境でストレージを扱う方法を定義します。クラスタ・ファイル・システムは、UNIX の世界ではようやく本領を発揮し始めているところであり、その仕組みについて詳しく説明します。
- 構成 ----- クラスタの物理的および論理的な構築方法を定義します。
- アプリケーションのサポート ----- 単一のスタンドアロン・システムで現在動作しているアプリケーションを、クラスタ環境でどのように利用できるかを説明します。アプリケーションの変更が必要かどうか？ 必要な場合にはどのように変更するか？ クラスタ環境で実行する利点は何か？ などの疑問に答えます。
- 耐災害性 ----- 良好に稼動しているクラスタ・システムに何らかの障害が発生した場合の動作について説明します。このトピックでは、ホスト・ベースの RAID、広域「ストレッチ」クラスタ、拡張クラスタについて説明し、耐災害運用の事例を示します。

このドキュメントでは、クラスタ・ソフトウェアとして Linux LifeKeeper、NonStop Kernel G06.13、HP-UX および Linux の両方に対応した Serviceguard 11i、TruCluster V5.1b、OpenVMS Cluster Software V7.3-1、および Windows 2000 DataCenter システム・クラスタを取り上げます。

Linux については、HPTC 技術(つまり Beowulf)ではなく、ハイ・アベイラビリティ (高可用性) の面を主に取り上げます。

シングル・システム・ビュー・クラスタとマルチ・システム・ビュー・クラスタ

クラスタ技術を公平に評価するには、次の 4 つの用語を定義する必要があります。すなわち、可用性(アベイラビリティ)、信頼性、拡張性(スケーラビリティ)、および管理容易性の 4 つです。

- 可用性は、クラスタのコンポーネントがダウンしてもアプリケーションが動作を続けられるかどうかを示します。たとえば、2 つのシステムで構成されているクラスタにおいて、一方のシステムがダウンしても、もう一方が作業負荷を引き継げば、アプリケーションの可用性は保たれることになります。可用性は、フェイルオーバに要する時間からも影響を受けます。たとえば、別のシステムへアプリケーションをフェイルオーバするのに 30 秒かかった場合、最初のシステムを使用していたユーザは、アプリケーションが 30 秒間ダウンしていたと認識します。
- 信頼性は、いくつかのコンポーネントが故障したときにシステムの性能がどれだけ保たれるかを示します。たとえば、クラスタを構成するすべてのシステムが正常に動作している場合に、クエリに対する応答が 1 秒以下で返されバッチ・ジョブが 8 時間で完了するようなクラスタ環境において、クラスタ内のシステムが 1 つまたは複数故障したときに同じレベルの性能が得られるかどうか？ あるいは、 2 つのシステムで構成されるクラスタにおいて、それぞれのシステムに 500 人のユーザが接続しても十分な性能が得られている場合に、一方のシステムが故障したために他方のシステムを 1,000 人のユーザが使用することになったときに、十分な性能が得られるか？ 一般ユーザは、クラスタを構成するシステムの数については意識しないことに注意してください。作業を完了するために信頼できる環境であるかどうかだけを気にします。

  信頼性の高さと可用性の高さは必ずしも一致しない点に注意してください。一方が得られても他方が得られない場合があります。たとえば、システムにログインできても (つまり、可用性が保たれていても)、あまりにも処理が遅いため実用にはならない (つまり、信頼性がない) ということは良くあることです。
- スケーラビリティは、クラスタ内のシステムから得られる有効なパフォーマンスの割合を示します。たとえば、クラスタに 2 台目のシステムを追加すると、パフォーマンスが 2 倍になるのか？ あるいは、得られるのはそれよりも数パーセント低い値なのか？ 3 台目のシステムを追加すると、パフォーマンスが 3 倍になるのか？ そうではないのか？
- 管理容易性は、クラスタにシステムを追加すると、それらを管理する手間がどれだけ増えるかを示します。クラスタに 2 台目のシステムを追加すると、すべてのことを 2 回実施しなければならないため作業負荷も 2 倍になるのか？ それとも、クラスタを単独のシステムとして管理できるので、増えた手間はほんのわずかなのか？

マルチ・システム・ビュー・クラスタは一般に、2 つのシステムで構成されており、それぞれのシステムが異なる一連のタスクを担当します。ストレージは、物理的には両方のシステムに接続されていますが、各ファイル・システムは、どちらか一方のシステムにのみマウントできます。つまり、アプリケーションが両方のシステムから同じデータに同時にアクセスすることはできないということです。また、両システムの間でオペレーティング・システム・ファイルを共有できないといため、それぞれのシステムに、オペレーティング・システム・ファイル一式とクラスタ・ソフトウェア・ファイルを格納した完全に独立したブート・デバイス (「システム・ルート」または「システム・ディスク」と呼ばれます) が必要になります。

アクティブ・パッシブ・モードのマルチ・システム・ビュー・クラスタ

アクティブ・パッシブ・モードのマルチ・システム・ビュー・クラスタは、ベンダーにとって実装が最も簡単なクラスタです。主たるシステムが何らかの理由で故障した場合に備えて、スタンバイ状態の予備システムを用意しておくというものです。予備システムは、アクティブなシステムが故障した場合を除いて、ほとんどの時間、アイドル状態です。これが、「アクティブ・パッ

シブ」の由来です。つまり、通常の運用において、一方のシステムがアクティブである間、もう一方のシステムはそうではない状態だからです。従来は、「N＋1 クラスタリング」と呼ばれていました。2 つのシステムで構成されるクラスタの場合、N＝1 です。システムの数がさらに多いクラスタでは、任意の数のアクティブ・システムを引き継ぐのに十分な性能を持った予備サーバを用意します。

フェイルオーバは、手動または自動で行えます。両方のシステムが同じストレージ・アレイに接続されているため、予備システムはメイン・システムを監視し、メイン・システムの故障を検知すると、予備システム側でサービスを開始することができます。「ハートビート」機能は、ネットワーク経由または専用のインタフェースを経由して使用できます。

アクティブ・パッシブ・モードのマルチ・システム・ビュー・クラスタの可用性、信頼性、スケーラビリティ、管理容易性は、次のとおりです。

- 2 つのシステムで処理に対応できるので、可用性が高くなります。ただし、同時に両方のシステムが故障する可能性は非常に低いですが、ゼロではありません。
- この環境の場合、2 つのシステムのハードウェアはまったく同じで、アプリケーションはどちらのシステムで動作しても同じパフォーマンスが得られるため、信頼性は完全であるといえます。
- アクティブ・パッシブ・クラスタのスケーラビリティは、貧弱です (無いに等しい)。アプリケーションが両方のシステムから共通のデータ・セットにアクセスできないため、マルチ・システム・ビュー・クラスタには、スケーラビリティがありません。つまり、システム 2 つ分のハードウェアで、システム 1 つ分の作業を行うことになります。
- 管理容易性も貧弱です。管理の手間が、単独のシステムのほぼ 2 倍かかるからです。システムのルートが 2 つあるため、パッチや他の更新を 2 回適用する必要があり、バックアップも 2 回取る必要があります。また、フェイルオーバとフェイルバックもテストの必要があるため、システム管理の手間はさらに増えます。

2 つめのシステムはほとんどの時間、アイドル状態であり、その間、そこから得られるビジネス上のメリットは何もない点に注意してください。したがって、考えうる代替案は、両方のシステムに作業を行わせることです。

アクティブ・アクティブ・モードのマルチ・システム・ビュー・クラスタ

アクティブ・アクティブ・モードのマルチ・システム・ビュー・クラスタの環境は、物理的にはアクティブ・パッシブ・モードの場合とまったく同じです。一般に、共通のストレージ・セットに物理的に接続されている 2 つのシステムで構成されますが、ファイル・システムは一方のシステムにのみマウントできます。違いは、複数のシステムがそれぞれ有益な作業をしており、また、お互いの状態を監視しているという点です。ただし、システム間で共有しているものはないため、同じアプリケーションが同じデータを使用するということはありません。

たとえば、データベース環境において、名前が A ～ M で始まる顧客と N ～ Z までの顧客の 2 つのグループに分けて顧客データを処理することができます。共有ストレージの別々のパーティションにそれぞれの顧客グループのデータを配置し、それぞれのシステムがどちらか一方のグループを扱うように設定します。これを「フェデレーテッド」データベースと言います。あるいは、一方のシステムでデータベース全体を扱い、もう一方のシステムでそのデータベースにアクセスするアプリケーションを実行するということも考えられます。

障害が発生した場合には、1 つのシステムで両方のシステムの処理を扱うことになります。

このようなクラスタ環境を、「N＋M クラスタ」と言います。任意のシステムが他の任意のシステムの処理を引き継ぐからです。N と M を定義するとき、自動車に付いているタイヤを思い浮かべるとよいでしょう。ほとんどの人はタイヤの数を反射的に「4 本」と考えますが、N＋1 の環境では、自動車が機能するのに必要となる最低限の数が 4 本なので、実際にはスペアを含めタ

イヤが 5 本必要となります。この考え方の変形として、「ドーナッツ」タイヤ[1] を使用することが考えられます。このタイヤの性能は限定的ですが、当座をしのげる程度には使用できます。この場合は、自動車にタイヤを 4.5 本用意するというように考えることができます。大事なのは、必要となる性能と機能のレベルを規定し、その環境に合った N と M を定義することです。

フェイルオーバは、手動または自動で行えます。「ハートビート」機能は、ネットワーク経由または専用のインタフェースを経由して使用できます。

アクティブ・アクティブ・モードのマルチ・システム・ビュー・クラスタの可用性、信頼性、スケーラビリティ、管理容易性は、次のとおりです。

- 2 つのシステムで処理に対応できるので、可用性が高くなります。アクティブ・パッシブ環境の場合と同様に、同時に両方のシステムが故障する可能性は非常に低いですが、ゼロではありません。
- 信頼性は、この場合には保証されません。クラスタを構成する各システムが 60% の能力で動作している場合、障害が発生したときに処理を引き継ぐシステムは 120% の能力で動作することになります。能力の 80% を超える負荷で動作するシステムがどのような状態になるかは、ご存知のとおりです。
- 作業負荷がそれぞれ 1 つのシステムに割り当てられるので、スケーラビリティは貧弱になります。1 つのアプリケーションを複数のシステムで分散実行させる方法はありません。
- 管理容易性は、アクティブ・パッシブ・モードに比べて若干劣ります。依然として 2 つの独立したシステムが存在し、フェイルオーバ・スクリプトとハートビートによるオーバーヘッドがあるからです。

マルチ・システム・ビュー・クラスタのフェイルオーバ (アクティブ・アクティブまたはアクティブ・パッシブ)

マルチ・システム・ビュー・クラスタの可用性に影響を与える要素の 1 つに、アクティブ・アクティブまたはアクティブ・パッシブのどちらのモードであるかにかかわらず、フェイルオーバを完了するまでの時間があります。処理を引き継ぐ場合、システムは次の処理を行います。

- 利用不可能になったシステムの検知。処理を引き継ぐシステムの「ハートビート」機能に対して、障害が生じたシステムからの応答がなくなったことにより検知します。
- 障害が発生したシステムにマウントされていたディスクのマウント。ファイル・システムは、一度に 1 つのシステムにのみマウントできます。これは、絶対的なルールであり、マルチ・システム・ビュー・クラスタを規定するものです。処理を引き継ぐシステムは、他方のシステムにマウントされていたディスクをマウントする必要があります。ディスクの数が多い場合、または、規模の大きい RAIDset を使用している場合には、この処理に時間を要することがあります。
- 障害が生じたシステムで実行していたアプリケーションの起動。
- 当該ソフトウェアのリカバリ処理の開始。データベースの場合、障害が生じたシステムで処理中だったトランザクションを完了するために、REDO ログを処理する必要があるかもしれません。

大規模な環境では、フェイルオーバに 30 分～ 60 分かかることも珍しくありません。このリカバリ時間の間、障害が生じたシステムで動作していたアプリケーションが利用できない状態となる一方で、処理を引き継ぐシステムではより多くの処理を行うことになるため、そこで動作していたアプリケーションに信頼性低下の問題が発生します。

---

[1] 使用時にコンプレッサーで空気注入して装着するタイプのコンパクトなスペアタイヤ。

シングル・システム・ビュー・クラスタ

シングル・システム・ビュー・クラスでは、クラスタ全体を単体として見ることができます。クラスタを構成するすべてのシステムがどのストレージにも物理的に接続されていて、どのシステムもすべてのストレージを直接マウントできます。つまり、どのシステムも、すべてのアプリケーションを実行でき、同じパーティションの同じデータを見ることができ、非常に下位レベルでの協調動作が可能です。さらに、オペレーティング・システムのファイルを単独の「共有ルート」または「共有システム・ディスク」に配置して共有することで、ストレージの容量と、システムの保守に必要な管理時間を削減できます。予備のシステムはありません。すべてのシステムがすべてのアプリケーションをいつでも実行できます。シングル・システム・ビュー・クラスタは、多数のシステムで構成できます。シングル・システム・ビュー・クラスタの可用性、信頼性、スケーラビリティ、管理容易性は、次のとおりです。

- 複数のシステムで処理に対応できるため、可用性が高くなります。多数のシステムでクラスタが構成される場合、クラスタ内のすべてのシステムが同時に故障する確立ははるかに低くなります。
- 信頼性も高くなります。クラスタが複数のシステムで構成されていれば、いずれかのシステムが故障しても、その処理を複数のシステムに引き継ぐことができるため、負荷の上昇も抑えられます。たとえば、4 台のシステムが 60% の能力で動作していて、その 1 つが故障した場合、その作業の負荷を 3 分の 1 ずつ、それぞれのシステムに振り分ければ、負荷の増加は容量の 80% までに抑えられ、信頼性が大きく損なわれることはありません。
- 作業負荷を複数のシステムに分散できるため、スケーラビリティには優れます。たとえば、(たとえ、64 個または 128 個の CPU を搭載し、数百ギガバイトのメモリ、および、数 10 枚の I/O カードがあっても) 1 台のコンピュータ・システムで対応するには巨大すぎるアプリケーションがあった場合、十分なリソースがありそれぞれが同じデータにアクセスする多数のコンピュータ・システムで同時に実行させることができます。
- クラスタ全体を単体のシステムとして管理できるため、管理容易性は、同数の独立のシステムを管理するよりもはるかに容易です。そのため、小規模なシステムが多数あったとしても、管理の作業負荷は増えません。

フェイルオーバ時に行う作業が少ないため、マルチ・システム・ビュー・クラスタに比べてフェイルオーバに要する時間が短くなります。

- 処理を引き継ぐシステムは、他のシステムの故障を検出する必要があります。これは、どちらの方式にも共通します。
- 処理を引き継ぐシステムは、故障したシステムがマウントしていたドライブをあらためてマウントする必要がありません。すでにマウントされているからです。
- また処理を引き継ぐシステムは、アプリケーションを起動する必要もありません。すでに起動されているからです。
- リカバリのためのスクリプトを実行する点は、どちらの方式でも同じですが、シングル・システム・ビュー・クラスタの場合はただちに実行を開始できます。アプリケーションのリカバリ時間はどちらの場合も似たようなものですが、より多くの小規模システムがあれば、リカバリ時にも並列処理を行って、リカバリまでの時間が短いことが考えられます。

HP のシステムで構成されるクラスタがこれらの方式にどのように対応するか？

用語を理解したところで、HP のシステムで構成されるクラスタがこれらの方式にどのように対応するかを見ていきます。

	マルチ・システム・ビュー	シングル・システム・ビュー	共有ルート
LifeKeeper Linux、Windows	○	×	×
Serviceguard	○	×	×
NonStop Kernel	○	○	ノードごと (16 CPU)
TruCluster	×	○	○
OpenVMS Cluster Software	○	○	○
Windows 2000 DataCenter	○	×	×

Linux におけるクラスタ

Linux におけるクラスタは、大容量システム・コンピュータ・ファーム (Beowulf など) の方式と、マルチ・システム・ビューのフェイルオーバ・クラスタ方式に二分されています。ここでは、大規模な問題を (何百、何千もの) 多数の小さな問題に分割し、(何百、何千もの) 小規模なコンピュータ・エンジンに任せる HTPC 市場の話をしているわけではありません。重要なポイントは、Linux におけるクラスタは可用性の高い環境ではないという点です。なぜならば、コンピュータ・エンジンのどれかに障害が発生すると、そこで処理されていた作業は、一から再実行する必要があるからです。

SuSE などによる高可用性への取り組みは、一方のシステムから他方のシステムにアプリケーションをフェイルオーバできる、2 台のシステムで構成されるマルチ・システム・ビュー・クラスタを中心としたものです。後ほど、Lustre、GFS、PolyServe などのクラスタ・ファイル・システムのプロジェクトを取り上げますが、これらは共有ルートを提供しないため、Linux クラスタではシステム・ディスクが個別に必要です。

シングル・システム・イメージ Linux プロジェクトの一部として HP が行っている作業については、まだ製品化されていないため触れません。しかし、製品化された暁には、他のすべてのクラスタ製品と同等またはそれらを超える高い機能を持ったものとなるでしょう。

Serviceguard

Serviceguard は、マルチ・システム・ビュー方式のフェイルオーバ・クラスタです。Serviceguard クラスタを構成する各システムは、個別にシステム・ディスクが必要です。Service Control Manager、Event Management Service を利用した優れたシステム管理機能が提供されており、System Configuration Repository へのソフトウェアの登録、システムのスナップショット取得、クラスタ内のシステムの比較などを行うことができます。また、HP/OpenView とも密接に統合されています。

Himalaya NonStop Kernel

Himalaya NonStop Kernel は、マルチ・システム・ビュー・クラスタとして構成することも、より一般的には、シングル・システム・ビュー・クラスタとして構成することもできます。ハードウェアとソフトウェアの両方における非常に優れたクラスタ・インターコネクトにより、業界最良のスケーラビリティ、実のところ、まさに直線的なスケーラビリティを提供します。16 CPU

まで搭載できるそれぞれのキャビネットはシステム・ディスクを共有でき、単体のシステムと見なされます。

TruCluster V5.1b

TruCluster V5.1b は、UNIX クラスタ技術における大幅な進歩を体現しています。シングル・システム・ビューとしての構成のみが可能で、単体システムも大規模クラスタも全く違いなく、同じツールを使用し、ほぼ同じ労力で管理できることを目指したものです。完全に共有可能なルートを提供し、ほぼすべてのシステム・ファイルについて、1 つのコピーだけで済みます。

OpenVMS Cluster Software

OpenVMS Cluster Software は、常にクラスタリングの最高の標準であり続けています。マルチ・システム・ビューとシングル・システム・ビューのどちらにも構成できますが、一般的にはシングル・システム・ビューで構成します。単独または複数のシステム・ディスクをサポートします。

Windows 2000 DataCenter

Windows 2000 DataCenter は、マルチ・システム・ビュー方式のフェイルオーバ・クラスタです。アプリケーションは、あるシステムから別のシステムにフェイルオーバするように作成されます。Windows 2000 DataCenter クラスタを構成する各システムは、個別にシステム・ディスクが必要です。これによって生じる負担をいくらか軽減する Cluster Administrator のようなツールがないわけではありませんが、システム・ディスクは別々に維持しなければなりません。

クラスタ・ファイル・システム

クラスタ・ファイル・システムは、クラスタ環境においてシステムがストレージ・サブシステムと通信するための方法を表します。これには 2 つの技術が関係します。1 つは、すべてのシステムに物理的に接続されているボリュームと各メンバ・システムが通信するための技術で、もう 1 つは、1 台のシステムにのみ物理的に接続されているボリュームと各メンバ・システムが通信するための技術です。

ネットワーク入出力は、クラスタのすべてのシステムがデータにアクセスすることを可能にします。ただし、効率が非常に悪いためスケーラビリティに劣ります。たとえば、システム A 専用の IDE または SCSI アダプタにボリューム A というディスクまたはテープ・ドライブが物理的に接続されていたとします。そのドライブには、クラスタの他のシステムから物理的にアクセスすることはできないため、クラスタ内の他のシステムからそのドライブ上のファイルにアクセスしたければ、ネットワーク入出力を行う必要があります。通常は、NFS のような機能を使用します。

たとえば、システム A にマウントされているデバイスにシステム B からアクセスしたければ、システム B のネットワーク・クライアントはシステム A のネットワーク・サーバと、次のように通信します。

- システム B からシステム A に対してクラスタ・インターコネクトを介して入出力が開始されます。
- システム A は要求を受け取り、ボリュームに対して入出力要求を開始します。
- システム A はボリュームからデータを受け取り、システム B に対する入出力を開始します。

ディスク・アクセスのたびに 3 つの入出力が発生する点に注目してください。NFS の場合、多くの NFS クライアントにおいてロックによる大幅なオーバーヘッドも生じます。そのため、アクティブ・アクティブ方式のシステムでは入出力のパフォーマンスが劣ることになります。

では、どのシステムもネットワーク入出力を提供するのは、なぜなのでしょう。テープ、CD-ROM、DVD、ディスケットなどの共有できないシングル・ユーザ・デバイスに対応するととも

に、専用の IDE または SCSI バス上のディスクなど、専用の通信経路に接続されているデバイスにアクセスできるようにするためです。

これに対し、ダイレクト・アクセス入出力 (同時入出力とも言います) では、各システムがクラスタ内の他のノードの助けを借りずに、個別に任意のどのデバイスにもアクセスできます。これは、単にファイル・システム・キャッシュをバイパスするダイレクト入出力とは異なる点に注意してください。ほとんどのデータベース・システムは、データベース自身でデータをキャッシュしており、ファイル・システムのキャッシュを利用する必要がないため、クラスタ環境であれ非クラスタ環境であれ、どちらの環境でもダイレクト入出力を行います。

ダイレクト・アクセス入出力を実装すると、各システムがストレージのインターコネクトを介してボリュームと直接対話するため、クラスタ・ファイル・システムはネットワーク入出力のディスク・アクセスで発生する 3 つの入出力処理のうち 2 つを排除できます。また、クラスタ全体に渡って、ファイル・システムの完全な透過性とキャッシュの一貫性が実現されます。

しかしここで、1 つのディスクに大量の要求があると過負荷になる可能性があるという指摘も考えられます。もちろん、そのとおりです。しかし、それはクラスタ化されているか否かにかかわらず、どのファイル・システムについても同じことです。単独のディスクや単独のデータベース行がボトルネックになることは避けられません。すでに持っている知識とツールを使用し、単体のオペレーティング・システムの場合とまったく同じように、クラスタ環境でもこうしたボトルネックを考慮して設計とチューニングを行います。

HP のクラスタにおける入出力の特性を次の表にまとめます。

	ネットワーク入出力	ダイレクト・アクセス入出力	分散ロック・マネージャ
LifeKeeper Linux、Windows	NFS	Oracle の raw デバイス、GFS	Oracle 社が提供
Serviceguard	○	Oracle の raw デバイス	OPS Edition
NonStop Kernel	Data Access Manager	実質的に ○	該当せず
TruCluster	Device Request Dispatcher	Cluster File System	○
OpenVMS Cluster Software	Mass Storage Control Protocol	ODS-2 または ODS-5 上の Files-11	○
Windows 2000 DataCenter	NTFS	Oracle 社が提供	Oracle 社が提供

世の中のほとんどのシステムは、ここではネットワーク入出力と言っている、クライアント/サーバ型の入出力が可能です。これは理にかなっています。なぜなら、世の中のどのシステムも、共有のバス上にないデバイスを共有するための何らかの方法が必要だからです。

FailSafe と Serviceguard は、これを NFS または Veritas を使用して実現します。NonStop Kernel は、Data Access Manager (DAM) を使用します。TruCluster は、Device Request Dispatcher (DRD) と Cluster File System の両方を使用します。OpenVMS Cluster Software は、Mass Storage Control Protocol (MSCP) を使用します。そして Windows 2000 は、NTFS と Storage Groups を使用します。

ネットワーク入出力よりも興味深いのがダイレクト・アクセス入出力です。ここで 1 つ強調しておきたいのは、Oracle が、サポートしているほとんどすべてのシステムで raw デバイスに対するダイレクト・アクセス入出力機能を提供している点です。raw デバイスは、ファイル・システムほどの機能は持っていませんが、実際に使用されており、業界のすべての主要なコンピューティング・プラットフォームにおいて (少なくとも Oracle の場合)、ダイレクト・アクセス入出力を提供します。

HP と Cluster File Systems Inc. がアメリカ合衆国エネルギー省のために行っている Linux プロジェクトの 1 つに、もともと Carnegie Mellon 大学で開発された Lustre File System の機能を強化するというプロジェクトがあります。このプロジェクトは、ハイ・パフォーマンス・テクニカル・コンピューティング環境に重点を置いています。Oracle 社は、Oracle 9i Real Application Clusters 9.2 の一部として、Linux 向けにデータベース・ファイル用のクラスタ・ファイル・システムを開発しました。これは、raw デバイス上に作成されています。

NonStop Kernel (NSK) は、厳密に言うとすべての入出力がネットワーク入出力でありながら、NSK の効率性と信頼性、および CPU 間でボリュームの所有権を透過的に受け渡しできる機能のおかげで、他のネットワーク入出力方式に見られる貧弱なパフォーマンスと大きなーバーヘッドなしで、ダイレクト・アクセス入出力の優れた部分をすべて備えている点で注目に値します。つまり、NSK は、ネットワーク入出力が実装されてはいるものの、実質的にはダイレクト・アクセス入出力を提供します。NonStop Kernel (実質的には NonStop SQL) は、「shared-nothing」データ・アクセス方式を利用します。各プロセッサはディスク・ドライブのサブセットを所有し、それぞれへのアクセスは Data Access Manager (DAM) と呼ばれるプロセスによって制御されます。ディスクへのすべてのアクセスが DAM によって制御および調整されるため、DLM が不要です。

Serviceguard と Windows 2000 DataCenter は、独自のダイレクト・アクセス入出力方式を提供せず、Oracle の raw デバイスに依存しています。Oracle 社は、Oracle 9i Real Application Clusters 9.2 の一部として、Windows 2000 DataCenter 向けにデータベース・ファイル用のクラスタ・ファイル・システムを開発しました。これは、raw デバイス上に作成されています。

TruCluster はクラスタ・ファイル・システム(CFS)を提供し、クラスタ内のどのシステムからも任意のファイル・システムへの透過的なアクセスが可能です。しかし書き込み処理は、64K バイト未満のファイルの読み取り処理の場合と同じように、CFS クライアント・システムの要求に基いて CFS サーバ・システムによって行われます。つまり、すべての書き込み処理と小さなファイルの読み取り処理は、ネットワーク入出力によって行われます。ただし、O_DIRECTIO でファイルをオープンするアプリケーションの場合は例外です。

OpenVMS Cluster Software は、Files-11 ファイル・システムの動作をクラスタ環境でも利用できるように拡張します。単一システム上の 2 つのプロセスから共通にアクセスするようにオープンされたファイルも、クラスタ内の 2 つのシステム上のそれぞれのプロセスが共通にアクセスするようにオープンされたファイルも、全く同じように処理されます。つまり、すべてのファイル操作は自動的にクラスタ対応となります。

どのオペレーティング・システムにも、非クラスタ環境におけるファイルを対象としたロック・マネージャがあります。分散ロック・マネージャは、単にこの考え方をシステム間の処理に適用したものです。Oracle 社は、多数のオペレーティング環境で同じように動作しなければならない必要性から、独自に分散ロック・マネージャを開発しました。この機能は、Linux と Windows システムで利用できます。

Serviceguard には、分散ロック・マネージャが Serviceguard Extension for RAC の一部として含まれています。

NSK には、分散ロック・マネージャという考え方自体がありません。リソース (ファイル、ディスク・ボリューム、その他) はすべて特定の CPU にローカルなものであり、それらのリソースに対する通信はすべて CPU 間の標準のメッセージングにより行われるため、必要なロック処理はすべて CPU ごとにローカルに行われます。しかしこの場合も、実装の効率性から、非常に優れたスケーラビリティを提供します。

TruCluster には、ローカルのロック処理のための UNIX API の標準セットと、アプリケーションが分散ロックを行えるように実装された別の API のセットがあります。アプリケーションをクラスタ対応にするためには、この分散ロック API を使用するように変更を加える必要があります。

OpenVMS Cluster Software は、すべてのロックについて共通のロック API を使用し、ローカル・ロックとリモート・ロックを区別しません。したがって、アプリケーションはすべて自動的にクラスタ対応となります。

クォーラム

クラスタの構成について議論する前に、クォーラムの概念を理解しておくことが重要です。クォーラム・デバイス (ディスクの場合とシステムの場合があります) は、2 つのシステムがクラスタを形成するのに同等の能力があり双方ともすべてのディスクをマウントしている場合に、どちらかのシステムを選択する手段を提供するもので、クラスタの分断を防ぎます。

クラスタを最初に構成するとき、それぞれのシステムに特定のボート数 (通常は 1) を与えます。それぞれのクラスタ環境では、最適なパフォーマンスを得るために必要となる期待ボート数 (expected vote) を定義します。ほとんどの場合、期待ボート数はクラスタを構成するシステムの数と同じになります。これに基づいて、クラスタを形成するのに必要となるボート数を示す「必須クォーラム」値を算出できます。「実際のクォーラム」値がこの値を下回る場合、ソフトウェアはクラスタの形成を拒否し、通常は実行そのものを全く拒否します。

たとえば、クラスタを構成する 2 つのメンバ、システム A とシステム B がある場合、それぞれに 1 票ずつ持たせるとクラスタの必須クォーラムは 2 になります。

実行中のクラスタのボート数は、接続マネージャが通信可能なメンバの総数となります。ここではクラスタのインターコネクトが動作中であるため、2 つのシステムが利用可能です。クォーラム・ディスクはありません。したがって、ボート数は 2 票となります。つまり、実際のクォーラム値が必須クォーラム値以上であるため、クラスタは有効です。

しかし、クラスタのインターコネクトに障害が生じた場合はどうなるでしょう。クラスタが壊れ、クラスタ遷移が行われます。

システム A の接続マネージャがシステム B と通信できなくなるため、実際のボート数がそれぞれのシステムで 1 票ずつとなります。その結果、実際のクォーラムが 1 となり、クラスタを形成するのに必要な必須クォーラムの値より低いため、どちらのシステムも処理の継続を拒みます。

しかし、一方または両方のシステムがそれぞれ独自に処理を継続するとしたらどうなるでしょう。システム間で通信ができないため、どちらも次のようにシングル・システム・クラスタの形成を試みます。

- システム A は、クラスタを形成するためにすべてのディスクのマウントを試みます。
- しかし、システム B も、クラスタを形成するためにすべてのディスクのマウントを試みます。これがクラスタ分断の状態です。
- これらのディスクが、相互に通信のできない 2 つのシステムにマウントされた状態となります。通常このようになると、ロック処理やキャッシュの一貫性保持を考慮せずに、双方のシステムがファイルの作成、削除、拡張、書き込みを行おうとするため、ディスクが即座に不正な状態となります。

それでは、この状況を避けるにはどうすればよいのでしょう。クォーラム・デバイスを使うのがその答えです。

構成は前と同じですが、ここでは両方のシステムに物理的に接続された 1 つのクォーラム・ディスクを追加しています。それぞれのシステムが 1 票を持っており、クォーラム・ディスクも 1 票持っています。システム A の接続マネージャは、システム B およびクォーラム・ディスクと通信できるため、期待ボート数は 3 票となります。この場合、クォーラムは 2 となります。ここでクラスタのインターコネクトが再び故障したとします。今度は何が起こるでしょうか。

- 両方のシステムがクラスタの形成を試みます。しかし、システム A が競争に勝って、先にクォーラム・ディスクへのアクセス権を取得したとします。システム A は、システム B に接続できないため、システム A のクォーラム・ディスク・ウォッチャによって、クォーラム・ディスクに現在行われているリモート入出力がないことが確認されると、クラスタの創設メンバとなり、クラスタ創設メンバの ID や、クラスタが新たに形成された時間などの情報をクォーラム・ディスクに書き込みます。その後、システム A はクラスタ・メンバの票数を数え (自身とクォーラム・ディスクの分を合わせて 2 票)、クラスタを形成するのに必要な票が集まったと判断します。そこでクラスタを形成し、共有バスのディスクをすべてマウントします。
- システム B は、2 番目にクォーラム・ディスクにアクセスします。システム A には接続できないため、自身がクラスタの創設メンバであると想定し、クォーラム・ディスクを確認します。しかしそこで、実際にはシステム A が創設メンバとなっていることを知ります。しかし、システム B はシステム A と通信できないため、クォーラム・ディスクにアクセスできません。そこでシステム B はクラスタ・メンバの票数 (自身のみの 1 票)が、クラスタを形成するのに必要な票が足りないと判断します。その後、システム B のその他の設定によって、ブート処理が続行されるかどうかが決まりますが、クラスタの形成またはクラスタへの参加は試みません。つまり、クラスタ分断は発生しません。

このようにして、一方のシステムだけがクラスタのすべてのディスクをマウントします。クラスタ内に他のシステムがある場合、必須クォーラムと期待クォーラムの値は高くなりますが、同じアルゴリズムにより、クラスタの創設メンバと通信できるシステムはクラスタに参加でき、クラスタの創設メンバと通信できないシステムはクラスタから除外されます。

クラスタの構成

次の表に、HP のクラスタ技術における重要な構成上の特性をまとめます。

	クラスタを構成するシステムの最大数	クラスタのインターコネクト	クォーラム・デバイス
LifeKeeper Linux、Windows	16	ネットワーク、シリアル	○ (オプション)
Serviceguard	16	ネットワーク、HyperFabric	2 台まで＝○、3 台以上＝オプション
NonStop Kernel	255	SystemNet、TorusNet	×
TruCluster	一般に 8。Alpha SC では 32	100Enet、QSW、Memory Channel	○ (オプション)
OpenVMS Cluster Software	96	CI、ネットワーク、MC、共有メモリ	○ (オプション)
Windows 2000 DataCenter	4	ネットワーク	○

Linux ではマルチ・システム・ビューのフェイルオーバ・クラスタに重点を置いているため、FailSafe では 最大 16 台のシステムを使用してクラスタを構成できます。これらのシステムはネットワーク・ケーブルまたはシリアル・ケーブルを使用して接続されます。任意のシステムが他

の任意のシステムの処理を引き継げます。クォーラム・ディスクはサポートされていますが、必須ではありません。

Serviceguard では、クラスタ・インターコネクトに標準のネットワークまたは HyperFabric を使用して、最大 16 台のシステムでクラスタを構成できます。この場合の特別な要件の 1 つとして、最初にクラスタを形成するときにクラスタ・メンバがすべてそろっている必要があり (クォーラム達成率 100%)、運用を継続するには 50% を超えるクラスタ・メンバがそろっている必要があります。運用可能なシステムが偶数あって半々の状態になった場合、クォーラム・ディスクまたはクォーラム・システムが均衡を断つ手段として使用されます。ノードが 2 つだけの場合、ディスクまたはシステムがクォーラム・デバイスとして必須となります。それ以外の規模のクラスタでは、クォーラム・デバイスはオプションとなります。クラスタ・クォーラム・ディスクは、2 ～ 4 個のノードのクラスタでサポートされており、クラスタ・クォーラム・システムは、2 ～ 16 台のシステムのクラスタでサポートされています。

NonStop Kernel は、最大 255 台のシステムを使用してクラスタを構成できますが、システム間のやり取りを考慮すると、最大で 16 個のプロセッサを搭載したシステムを 255 台使用し(つまり 4,080 個のプロセッサを使用して)クラスタを構成できる、と言うほうが適切です。各システムの通信経路として SystemNet が使用され、TorusNet が旧型の K シリーズに対するクラスタ・インターコネクトを提供し、SystemNet が、リモート・データセンタを含め S シリーズに対するより最近のクラスタ・インターコネクトを提供します。NSK は、クォーラム方式が意味を持たない、何も共有しない環境であるので、クォーラム方式は使用されません。

TruCluster は、任意の規模のシステムを 8 つまで使用してクラスタを構成できます。HPTC 市場向けには、Alpha System SuperComputer システムを 32 台のシステムで構成できます。この構成は、非常に高速なインターコネクトとして Quadrix Switch (QSW) を使用します。

OpenVMS Cluster Software では 96 台までのシステムを使用してクラスタを構成でき、VAX システムと Alpha システムの任意の組み合わせ、および 2004 年以降は Itanium システムと Alpha システムの任意の組み合わせで、アーキテクチャ混在クラスタを構成できます。

TruCluster と OpenVMS Cluster Software では、2 つのノードで構成されるクラスタではクォーラム・ディスクの使用が推奨されますが、それを超えるノード数のクラスタではクォーラム・ディスクの使用はオプションとなります。

Windows 2000 DataCenter では、最大 4 台のシステムを使用してクラスタを構成できます。Windows 2000 DataCenter はサービスとしての販売となるため、このクラスタを構成して提供できるのは、HP のような Microsoft 認定パートナーのみです。この場合のクラスタ・インターコネクトは、標準の LAN 接続のみです。

アプリケーションのサポート

次の表に、HP のクラスタ技術がアプリケーションに対して提供するサポートをまとめます。

	シングル・インスタンス (フェイルオーバ・モード)	マルチ・インスタンス (クラスタ全体)	回復の手段
LifeKeeper Linux、Windows	○	×	スクリプト
Serviceguard	○	×	パッケージとスクリプト
NonStop Kernel	○ (引き継ぎ)	実質的に ○	プロセス二重化

TruCluster	○	○	Cluster Application Availability
OpenVMS Cluster Software	○	○	Batch /RESTART
Windows 2000 DataCenter	○	×	登録、クラスタ API

シングル・インスタンス・アプリケーションとマルチ・インスタンス・アプリケーション

クラスタ環境におけるアプリケーションの形態として、シングル・インスタンス・アプリケーションとマルチ・インスタンス・アプリケーションの 2 種類があります。この 2 つの用語は、最初に紹介したマルチ・システム・ビュー・クラスタとシングル・システム・ビュー・クラスタとは反対の見方である点に注意してください。

マルチ・システム・ビュー・クラスタではシングル・インスタンス・アプリケーションとしての実行が可能で、この場合、別のメンバ・システムへアプリケーションをフェイルオーバすることにより高可用性を提供します。ただし、クラスタ内の複数のメンバ・システムで同じアプリケーションを実行する場合、それらが同じデータを処理することはできません。

シングル・システム・ビュー・クラスタではマルチ・インスタンス・アプリケーションとしての実行が可能で、この場合、別のメンバ・システムで実行中のアプリケーションへのフェイルオーバにより高可用性を提供します。クラスタ内の複数のシステムで実行する同じアプリケーションが同じデータを使用し、相互に協調して処理することも可能です。

アプリケーションがシングル・インスタンス型かマルチ・インスタンス型かを見極める方法として、単一システムの複数のプロセスでアプリケーションを実行してもアプリケーションが相互に干渉することなく、単一システムの 1 つのプロセスで実行した場合と同様に正常に実行されるかどうかを確認するという方法があります。正常に実行される場合、アプリケーションはシングル・インスタンス型と言えます。

シングル・インスタンス型のアプリケーションの例としては telnet があります。クラスタ内の複数のシステムで telnet サービスを提供しても、複数の telnet セッションで同じデータを処理することはなく、相互に干渉することもありません。特定のシステムで障害が起きた場合、ユーザはクラスタ内の次のシステムにログインし、セッションを再開します。これは単純なフェイルオーバです。この方法は、Linux や Windows 2000 DataCenter の他、多くの競合システムが、NFS サービスをフェイルオーバ・モードのシングル・インスタンス・アプリケーションとして設定する方法です。

これに対し、複数のプロセスで実行中のアプリケーションが、キャッシュやロック・データ構造を共有し、正しく協調動作する場合、そのアプリケーションはマルチ・インスタンス型と言えます。

マルチ・インスタンス・アプリケーションの例としては、複数のシステムが、環境内の他のシステム対して同じディスク・セットをサービスするクラスタ・ファイル・システムがあります。複数のシステムで 1 つのディスク・セットを共有するには、シングル・システム・ビュー・クラスタ方式のクラスタ・ファイル・システムが必要です。このファイル・システムは、オペレーティング・システム・ソフトウェア自身(TruCluster および OpenVMS Cluster Software などの)、あるいは他社製のソフトウェア (Oracle 9i Real Application Clusters など) によって提供されます。このため、FailSafe、Serviceguard、Windows 2000 ではマルチ・インスタンス・アプリケーションが利用できないと言っても、Oracle は例外となります。先に述べたように、NonStop Kernel ではこれを別の手法で実現していますが、実質的に同等な機能を提供します。

リカバリ手法

リカバリ手法は、意図的なものであれシステム障害によるものであれ、システムがクラスタから切り離された場合にそのシステムで実行されていたアプリケーションを、クラスタ内でどのように回復するかを実装したものです。Linux のようなマルチ・システム・ビュー・クラスタでは、クラスタ・メンバ間のハートビート・メッセージでシステム障害を検出した際に、スクリプトを起動することでこれを実現しています。Linux 向けのリカバリ手法の例に、mon と monit があります。

NonStop Kernel のようなフォールト・トレラント・システムの場合、リカバリ処理は二重化プロセスによって実現されます。二重化プロセスでは、プライマリ・プロセスを追随するバックアップ・プロセスが待機しており、障害が発生した場合にいつでも処理を引き継げるようになっています。

Serviceguard には、アプリケーションとそれらの実行に必要なリソースをまとめて、1 つの単位として扱えるようにパッケージ化するためのさまざまなツールが用意されています。システム管理者は、サーバで障害が発生した場合に、クラスタ内の別のシステムでアプリケーション・パッケージをリカバリし再起動するためのプロシージャを定義できます。これは、Serviceguard が UNIX のクラスタ技術に関して業界をリードしている領域の 1 つです。

TruCluster および OpenVMS Cluster マルチ・インスタンス・アプリケーションには、障害が発生したシステムでリカバリを行うための機能が組み込まれています。TruCluster および OpenVMS Cluster Software は共に、複数のアプリケーションを監視し、それらを自動的にリカバリできます。TruCluster では、これを Cluster Application Availability 機能を使用して実現し、OpenVMS Cluster Software では、バッチ用 SUBMIT コマンドの /RESTART スイッチを使用して実現します。

Windows 2000 DataCenter の場合、3 つの主要なリカバリ手法があります。

- 一般のアプリケーション/一般のサービス。この手法の場合、開発の必要はなく、Windows 2000 DataCenter による保護を受けるためにアプリケーションを一度登録するだけで済みます。管理者が登録するためのウィザードも用意されています。
- カスタム・リソース型。アプリケーションそのものに変更を加える必要はありませんが、Windows 2000 DataCenter がアプリケーションの監視とフェイルオーバを実行できるように、アプリケーション・ベンダー (またはその他の組織) がアプリケーション固有のリソース DLL を開発します。この手法の場合、導入時に開発の必要はなく、専用のリソース DLL を使用してアプリケーションを登録するだけで済みます。
- クラスタ API。この場合はアプリケーションに変更を加え、クラスタ環境で実行していることを認識させ、フェイルオーバ、ノードの照会などのクラスタ関連操作を実行できるようにします。

クラスタの耐障害性

次の表に、HP のクラスタ技術におけるクラスタの耐障害性に関する特性をまとめます。

	動的パーティション	データの高可用性	耐災害性
LifeKeeper Linux、Windows	×	Distributed Replicated Block Device (DRBD)	拡張ミラーリング

Serviceguard	vPars	マルチパス I/O (a) MirrorDisk/UX	拡張クラスタ
NonStop Kernel	×	マルチパス I/O (p)、RAID-1、プロセスの二重化	リモート・データベース機能
TruCluster	×	マルチパス I/O (a) LSM RAID-1	StorageWorks Continuous Access
OpenVMS Cluster Software	Galaxy	マルチパス I/O (p) HBVS RAID-1	DTCS、StorageWorks Continuous Access
Windows 2000 DataCenter	×	SecurePath (p) NTFS RAID-1	StorageWorks Continuous Access

動的パーティション

クラスタの主なセールスポイントの 1 つは、ストレージ・サブシステムの障害、想定以上の作業負荷の発生、あるいは物理的な災害のような重大な障害が発生した場合の高い可用性です。HP のシステムで構成されるクラスタでは、これらの問題や障害にどのように対応するのでしょうか。

動的パーティションにより、作業負荷の増加や変化に対応します。従来は、大半の時間は予備のハードウェアが使用されないことを承知の上で、最大の負荷を想定した CPU とメモリ容量を使用してシステムを構築していました。そしてシステムの集約によってハード・パーティショニングの採用が多くなるにつれて、それぞれのハード・パーティションで最大の負荷を想定して十分な CPU とメモリを用意する必要が生じます。しかし動的パーティショニングでは、こうした予備のハードウェアを、より規模の大きなシステムのパーティション間で共有できます。そのため、たとえば、日中は CPU の大半をオンライン処理用のパーティションに割り当て、夜間はバッチ処理用のパーティションに割り当てるといったことが可能です。この機能を提供しているのは、HP-UX の vPars、および OpenVMS の Galaxy だけです。どちらも、GUI またはコマンド行から、CPU をパーティション間で動的に移動する手段を提供します。Galaxy は、2 つ以上の Galaxy パーティションで物理メモリを共有する手段を提供します。HP-UX は、目標達成方式の運用要件に応じて CPU の割り当てを変更するための機能を Work Load Management (WLM) ツールで提供します。

上で述べたソフトウェアはすべて、クラスタ製品だけでなく、基本オペレーティング・システムで動作する点に注目してください。vPars と Galaxy のパーティションは、それぞれのオペレーティング・システムのその他のインスタンスと同じように、どちらもクラスタ化することができます。

データの高可用性

データの高可用性は、ホスト・アダプタの冗長化、ホスト・アダプタからストレージ・コントローラまでの経路の冗長化 (FibreChannel を使用している場合は冗長スイッチを使用して)、自動フェイルオーバのために構成されたストレージ・コントローラの冗長化、ディスク本体の適切な RAID レベルなど、ストレージ・サブシステムのレベルで必要なことがすべて行われていることが前提となります。さらに、これらの冗長性のいくつかは、特にマルチパス I/O の領域で、ホスト・オペレーティング・システムからの協調が必要となります。

マルチパス I/O により、システムはホストから特定のボリュームへの物理的なパスを複数持てるようになります。たとえば、複数のホスト・アダプタを冗長化されたストレージ・コントローラに接続できます。この方法は FibreChannel ではきわめて一般的ですが、SCSI でも実現が可能です。

マルチパス I/O には、アクティブとパッシブの 2 通りの方法があります。アクティブ・マルチパス I/O では両方のパスが同時にアクティブな状態になり、それぞれの I/O において負荷が最も低いホスト・アダプタを選択することにより、オペレーティング・システムが複数の物理的なパス間で I/O 要求の負荷のバランスをとることができます。ホスト・アダプタ、スイッチ、またはストレージ・コントローラの障害が原因で、パスに障害が発生した場合、オペレーティング・システムは別のパスに対して I/O 要求を再発行します。この動作は、アプリケーションからは見えません。

パッシブ・マルチパス I/O ではアクティブなパスは常に 1 つで、仮に最初のアクティブなパスで障害が発生した場合は次のパスがいつでもパスを引き継ぐことができます。この処理は、システムが I/O 要求を再発行するというアクティブ・マルチパス I/O と同じ方法で実現されます。前の表において、(a) はオペレーティング・システムがアクティブ・マルチパス I/O をサポートすることを示し、(p) はパッシブ・マルチパス I/O をサポートすることを示しています。

しかし、これらの技術は、ストレージ・キャビネットが物理的に破損した場合など、同時に複数のディスクで障害が生じた場合には十分とは言えません。

そのような状況に対する対策としてまず考えられるのは、ホスト・ベースの RAID です。ホスト・ベースの RAID では、複数のストレージ・キャビネットに渡ってミラーリングやシャドウィングを行います。Linux は、Distributed Replicated Block Device (DRBD) を使用してこれを実現します。DRBD では、いずれかのシステムがローカル・ディスクに書き込みを行うと、ネットワークを介して他のシステムに更新情報が送信され、それらのシステムのローカル・ディスクにもそのデータのコピーが書き込まれます。

HP-UX は、MirrofDisk/UX を使用して、データのコピーを最大 3 つまで維持します。アクティブ・マルチパス I/O を可能にするソフトウェアは、HP-UX が使用しているストレージ・システムによって異なります。HP-UX のカーネルに組み込まれている EMC 社の製品である PowerPath は、HP の Logical Volume Manager (LVM) が XP および EVA ストレージ・サブシステムに対してアクティブ・マルチパス I/O を提供するのと同じように、EMC 社のストレージ・アレイに対しアクティブ・マルチパス I/O を提供します。

NonStop Kernel は、RAID-1 (ディスクのミラーリング)、複数の SystemNet Fabric によるアクティブ・マルチパス I/O 、複数のコントローラ・パスのほか、フォールト・トレラント Data Access Manager (DAM) に対するプロセスの二重化技術を組み合わせることで、データの高可用性を実現します。

Tru64 UNIX は、ルート・ディスクを含め、任意のディスクを保護する Logical Storage Manager で RAID-1 および RAID-5 をサポートします。LSM は、アクティブ・マルチパス I/O もサポートします。

OpenVMS は、システム・ディスクを含む任意のディスクのコピーを 3 つまで維持できるホスト・ベースのボリューム・シャドウィングで RAID-1 をサポートします。OpenVMS は、パッシブ・マルチパス I/O をサポートし、負荷のバランスはオペレータが制御します。

Windows 2000 DataCenter は、NTFS ミラーリングで高可用性を実現します。HP を含め多くのストレージ・ベンダーが、彼らのストレージ製品用に、Windows 2000 の 上位にレイヤーを形成してパッシブ・マルチパス I/O を実現するソフトウェアを提供しています。Windows では、この種のソフトウェアを SecurePath と呼んでいます。

耐災害性

耐災害性は、コンピュータの運用とデータを物理的な災害から守るために必要となるものです。たとえば、筆者が住んでいるフロリダでは、ハリケーンの心配があります。世界の他の場所では、たとえば竜巻や吹雪の心配があります。そして誰もが、地震、停電、火事を心配します。このような災害からデータを守る唯一の方法は、データを離れた場所に、可能な限りリアルタイムに保

管することです。データの複製にはさまざまな方法がありますが、主要な 2 つの方法は、物理複製と論理複製です。

物理複製は、システムまたはストレージ・サブシステムのどちらでも行えます。システムは、データの高可用性を可能にする先述のソフトウェアを使用して物理複製を行いますが、データの第 2 (場合によっては第 3 の) コピーは物理的に別の場所に置かれます。Serviceguard は MirrorDisk/UX を使用し、NonStop Kernel は Remote DataCenter Facility (RDF) を使用してこれを行います。また、TruCluster は LSM を使用し、OpenVMS Cluster Software はホスト・ベースのボリューム・シャドウィングを使用します。リモート・システムによるリモート・ボリュームへのアクセスは、まるでリモート・ボリュームとリモート・システムが、ソース・システムおよびソース・ボリュームと同じ部屋にあるのと同じように行えます。ローカルであれリモートであれ、ボリュームをシステム間で共有できるのは、TruCluster と OpenVMS Cluster Software のみです。オペレーティング・システムが、広域 FibreChannel システムをまたいでボリュームをアクセスする方法を「拡張クラスタ」と呼びます。

この場合も RAID の場合と同じように、ホストのオペレーティング・システムまたはクラスタ・ソフトウェアの機能に関係なく、ストレージ・サブシステムが物理複製のための多様な機能を提供します。EVA および XP ストレージ・アレイ向けの StorageWorks Continuous Access は、HP および競合ベンダーの大半の UNIX システムのクラスタをサポートします。

システム・ベースのデータ高可用性ソフトウェアも、ストレージ・ベースの Continuous Access も、どちらもアクティブ・アクティブ型の双方向データ複製機能を提供します。これは、どういう意味でしょう。

仮定の話として、問題なく動作している運用システムがロサンゼルスにあったとします。そのデータを守るために、ロサンゼルスから 100 キロほど離れたサン・バーナーディノに災害復旧サイトを構築するとしましょう。

まず、FibreChannel スイッチを一式導入し、FibreChannel と ATM を接続するアダプタを使用して、それらをロサンゼルスのデータセンタにある FibreChannel スイッチに接続します。その後、サン・バーナーディノに同じ構成のストレージを導入し、ロサンゼルスの運用システムからサン・バーナーディノへのデータの複製を開始します。これは、アクティブ・パッシブ複製として知られる方法です。サン・バーナーディノ側はデータを受け取るのみだからです。サン・バーナーディノ側では、複製したデータを使用して処理が行われることはありません。サン・バーナーディノ側にはそのためのシステムがない、というのが大きな理由です。

当初はこれで十分かもしれませんが、いずれ次の段階に進む必要が生じてきます。つまり、ロサンゼルスのデータセンタが壊滅的な状態になっても、サン・バーナーディノで処理を継続できるようにする必要があります。そこで、サン・バーナーディノに数台のシステムを導入して、それらを既に導入済みのストレージに物理的に接続します。しかし、オペレーティング・システムの種類に関係なく、これらのシステムは Continuous Access によるコピーが置かれたターゲット・ボリュームをマウントできないため、サン・バーナーディノのサイトは、ロサンゼルスの処理を引き継ぐように指示する合図を受け取るまで、ずっと待機していることになります。合図は自動または手動のどちらも可能ですが、いずれの場合も、ストレージ・サブシステムが Continuous Access のリンクを切断し、システムがボリュームをマウントし、アプリケーションに対して定義されていた回復処理を開始します。

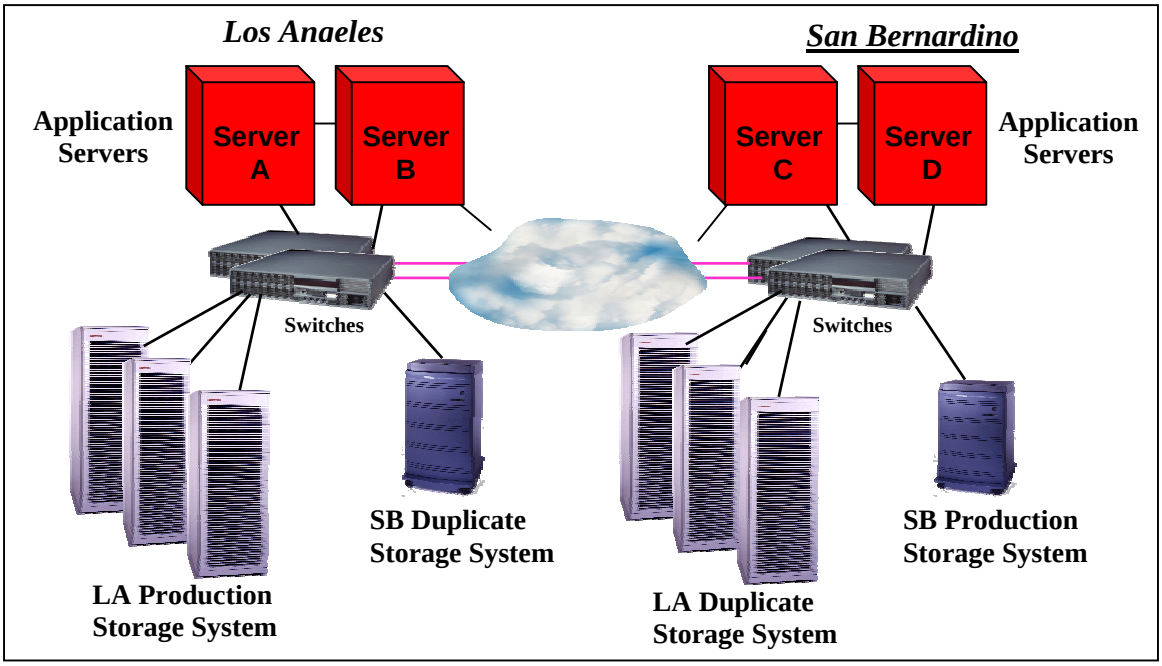
しかし、最高財務責任者が、サン・バーナーディノに一連のシステムをずっと待機状態で維持することに強い難色を示します。そこで、作業負荷を分割して、半分をロサンゼルスで、もう半分をサン・バーナーディノで担うことにします。このクラスタ・システムはマルチ・システム・ビュー・クラスタである点に注目してください。Continous Access によるコピーのターゲット・ボリュームをシステムにマウントすることはできません。そこで、ロサンゼルスのストレージをサン・バーナーディノへ複製したのと同様に、今度はサン・バーナーディノのストレージをロサンゼルスへ複製します。ストレージをロサンゼルスのシステムに接続し、サン・バーナーディノからロサンゼルスに対して Continuous Access によるコピーが行われるように設定します。

こうすることで、ロサンゼルスの運用データがサン・バーナーディノに複製され、サン・バーナーディノの運用データがロサンゼルスに複製されるようになります。どちらかのデータセンタが失われても、すべてのデータがもう一方に残り、処理を継続できます。

これをアクティブ・アクティブ型の双方向のデータ複製と言います。それぞれのストレージは一方向にのみ複製されますが、結果的にすべてのデータが組織全体にわたって複製されている状態となります。

複製が FibreChannel を通じて行われている点に注目してください。このためホストは、複製が行われていることを知らず、また気にする必要もありません。

フェイルオーバを行って稼動を続けるには、存続する側のデータセンタにおける複製プロセスを明示的に停止させ、存続データセンタのシステムにストレージ・サブシステムを明示的にマウントする必要があります。フェイルバック時にも同様の操作が必要となります。つまり、双方のストレージ・システムのデータの同期を取り、一方をソース・モードに戻し、もう一方をターゲット・モードに戻すことで、標準の構成に戻します。



システム・ベースのデータ高可用性機能の場合も、StorageWorks Continuous Access の場合も、複製されるデータは、システムまたはストレージ・コントローラによって複数回書き込まれます。そして、ソース・ディスク上のすべてのデータが、ミッション・クリティカルなデータベース・ファイルであれ、一時領域に書かれた一時ファイルであれ、ターゲット・ディスクに書き込まれます。したがって、必要なもの (たとえば、データベース・ファイルに含まれておらず、おそらく同じディスク・ボリューム上にもないデータベース・アプリケーションのための起動スクリプトなども忘れず) をすべて複製する一方で、余計なもの (/tmp など) は複製しないように、慎重に計画する必要があります。

これは、論理複製とはどのように違うのでしょう。最も大きな違いは、複製される内容と複製の方法です。

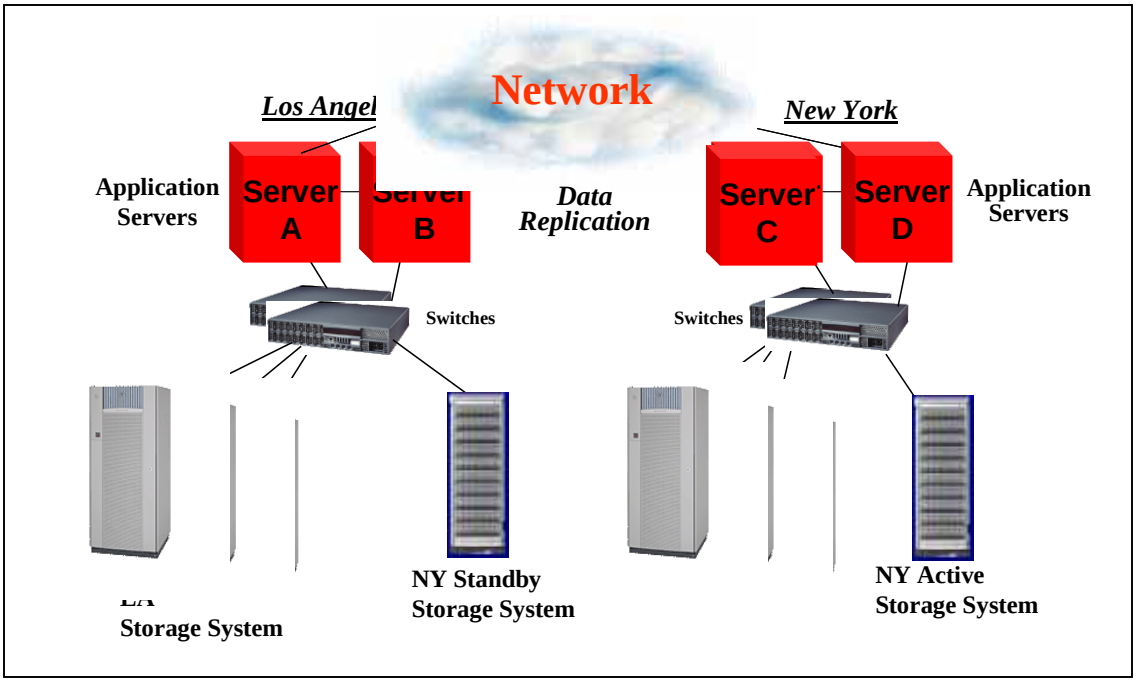
論理複製では、ディスク・ボリュームは無視され、トランザクションそのものが複製されます。物理複製では、1 つの書き込み操作を複数のディスク・ボリュームに適用しますが、それと同様に、論理複製では、1 つの更新トランザクションを複数のデータベースに適用します。そのための通信は、通常のネットワークを介して行うことが可能で、複数のデータベースを運用している

システムは、使用しているデータベース・ソフトウェアごとに、クラスタ化することも、しないこともできます。

再び仮定の話として、問題なく動作している運用システムがロサンゼルスにあったとします。今回も災害復旧サイトが必要であると判断したとします。しかし、今回の場合は、標準のネットワーク技術を使用するため、災害復旧サイトを任意の場所に構築できます。そこで、災害復旧サイトの場所としてニューヨークを選びます。

ニューヨークにデータセンタの複製を構築し、高速ネットワーク・リンクを使用して接続します。この場合、システムとストレージの両方が必要なため、データセンタとしての機能がすべて複製される点に注目してください。ただし、物理複製である必要はありません。一部のデータだけ複製すればよい場合、あるいは、フェイルオーバ発生時のパフォーマンスの多少の低下を受け入れる場合には、ニューヨークのコンピュータ・リソースは、ロサンゼルスより少なくてもかまいません。接続が完了したら、論理複製を使用して、データベース・レコードの複製を開始します。

このまま、アクティブ・パッシブ・モードにしておくこともできますが、ニューヨーク側でも何らかの運用システムを実行することもできます。ただし、システムが互いに独立しており、同じデータを参照することはできないため、ニューヨーク側で実行する運用システムはロサンゼルスと異なるものでなければなりません。しかし、双方向の物理複製があるのと同様に、双方向の論理複製も可能です。双方向の論理複製により、データ保護とニューヨークからロサンゼルスへのフェイルオーバの両方を提供できます。これは、アクティブ・アクティブ型の論理複製方式です。ただし、前の例と同様に、フェイルオーバとフェイルバックの処理は半自動化されていますが、一部は人による操作が必要となります。



何が複製されるのかを常に考慮する必要があります。物理複製の例においては、ストレージ・サブシステムによって、ディスクに書き込まれすべてのビットがレプリケーション先にコピーされました。ストレージ・サブシステムは、ファイル・システム、ファイル、データベース・レコード、REDO ログ、トランザクションをまったく認識しません。この場合の利点は、「何もかも」が複製されることです。データベース・ファイル、ログ・ファイル、フラット・ファイルなど、何もかもです。しかし不利な点は、オペレータの単純なミス ("rm -r *"や"DELETE [*...]*.*;*" など) も、リアルタイムでターゲット・ディスクに反映される点です。

これに対し、論理複製では、ボリュームではなく、データベース・トランザクションの複製を行います。セキュリティ情報、オペレーティング・システムやアプリケーションに対するパッチ、

スクリプト、その他、サイトの維持に必要なさまざまなファイルは複製されないため、それらは手作業によって複製して維持しなければなりません。しかし利点は、ファイルに対するオペレータの操作ミスがシステム全体を使用不能にすることがない点です。

物理複製と論理複製の違いについて最後に述べておくべきことが 1 つあります。物理複製は、対象とするデータの基盤構造をまったく認識しません。変更の加えられたディスク・ブロックは分かりますが、それらのブロックがどのファイルに属するのか、あるいは、それらがデータベース行なのかデータベース索引なのか分かりません。したがって、物理複製でトランザクションの原子性 (アトミック性) を保証することは不可能です。つまり、たとえば、行情報の書込み操作が正常に複製されたものの、索引情報がまだ複製されていない、という状態のタイミングが存在し得ます。トランザクションを開始したシステムで障害が発生するか、トランザクションの進行中に通信リンクに障害が発生すると、複製されているデータベースはおそらく整合性のない状態となります。

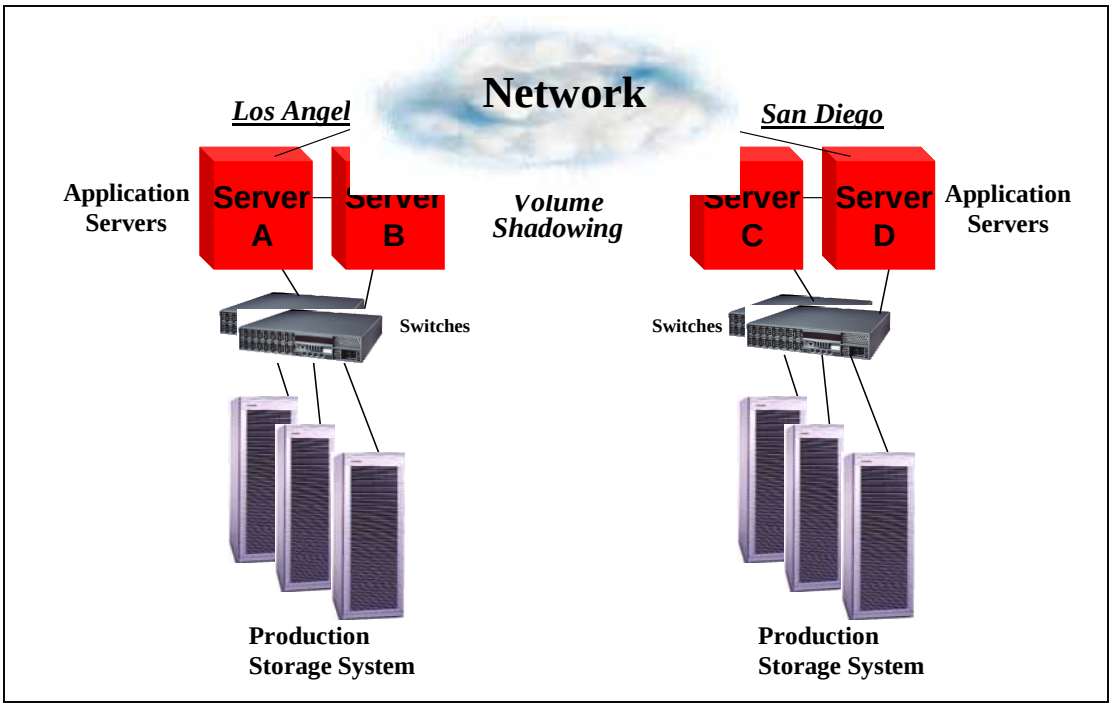
では、これと OpenVMS Cluster Software および Disaster Tolerant Cluster Services (DTCS) はどのように違うのでしょう。違いの 1 つは、物理複製が行われる場所ですが、それよりも重要なのは、ターゲット・ボリュームに広域クラスタの任意のメンバからアクセスできる点です。

再び仮定の話として、問題なく動作している運用システムがロサンゼルスにあったとします。今回も災害復旧サイトが必要であると判断したとします。しかし、今回の場合は、標準のネットワーク技術を使用するため、災害復旧サイトを 800 キロ以内の任意の場所に構築できます。そこで、災害復旧サイトの場所としてサンディエゴを選びます。

サンディエゴにデータセンタの複製を構築し、両サイトを ATM ファブリックによって相互接続します。レイテンシ (遅延) やスループットにかかわるさまざまな規則があり、クォーラムの組み立て方も非常に込み入ったものとなります。このようなサイトの設計と実装についてはぜひ弊社にご相談ください。ただし、多数のサイトがこのような構成で稼動している実績もあるので、それほど難しいわけではありません。

ここで、OpenVMS のホスト・ベースのボリューム・シャドウィングを使用してデータの複製を行う場合を例に説明します。HBVS (host-based Volume Shadowing) はホスト・ベースなので、すべてのシステムがボリュームを直接マウントすることができ、ダイレクト・アクセス入出力を通じて、ドライブに対する完全な読み込み/書き込みアクセスが可能です。任意のシステムによる任意の書き込みデータは、ロサンゼルスとサンディエゴの両方のストレージ・システムに書き込まれ、キャッシュの一貫性が完全に保たれているため、他のどのシステムもその書き込みデータを認識します。読み込みは、ローカルのディスクからとなるため、非常に効率的です。

予備システムもパッシブ・ストレージ・アレイもありません。ネットワーク障害にも対応できるように、ATM ファブリックを冗長化しておくことをお勧めします。しかし一方のサイトが壊滅的な状態になったとしても、残ったサイトはすでにディスクがマウントされていて運用されている状態なので、非常に短時間で回復することが可能です。クラスタ遷移が起こるだけですぐに業務を再開できます。



では、この方式では距離に制限があるのに対し、拡張クラスタあるいはストレージ・ベースの Continuous Access による方式では制限がないのはなぜでしょう。理由は、この方式ではサイト間の双方向の通信量が多く、光の速度にも限界があるからです。

光は真空中を秒速約 300,000 キロメートルで進みます。秒速 300,000 キロメートルということは、光は 1 ミリ秒に 300 キロ進むということです。われわれが関心を持っているのは、往復の距離なので、光は 1 ミリ秒に 150 キロ往復できると考えます。しかし、光ファイバを通る光の速度は真空中よりもいくらか遅く、スイッチによる遅延も避けられません。そこで、経験則から、80 キロの往復に 1 ミリ秒かかるとします。これに基づくと、800 キロの距離では、ディスク・アクセスのレイテンシに 10 倍× 1 ミリ秒＝ 10 ミリ秒の遅れが加わることになります。ディスク・アクセスの通常のレイテンシが 10 ～ 15 ミリ秒であるとすると、レイテンシはせいぜい 2 倍になるだけなので、OpenVMS のホスト・ベースのボリューム・シャドウィング・ソフトウェアならば十分に対応できます。しかし、それを超えるようだと、ソフトウェアはディスクがオフラインになったと間違った判断をし、シャドウ・セットを切り離してしまう可能性があります。これを回避するためにタイムアウトを延ばすと、今度は実際にアクセス不能になっているにもかかわらず、ソフトウェアはディスクが遅いだけだと判断し、本来シャドウ・セットを切り離すべきであるのに、切り離しが行われるまで容認できない遅れが生じる可能性があります。

物理複製および論理複製の場合、一方の側の各ボリュームから他方にデータを送り出しながら、最後に確認応答を受け取るのを待ちます。確認応答は時間に対する制限があまり厳しくないため、送信済みデータに対する確認応答を待ちながら、新しいデータを送信し続けることができます。他方のシステムからは対象ディスクへのデータの書き込みはできないため、競合状態を回避するための操作も必要ありません。光の速度の限界によるレイテンシの増加はそれほど大きな問題ではなく、使用するインターコネクト技術の距離の制限を超えない限り、システム間の距離の制限はないに等しいのです。

まとめ

どのオペレーティング・システムもハイ・アベイラビリティ・オプションを提供していますが、その能力はそれぞれ大きく異なります。2 つのノードで構成され、分単位で手作業によるフェイ

ルオーバを提供するシステムもあれば、16 のノードで構成され、秒単位の自動フェイルオーバを提供するシステムもあります。また、何百何千もの単位の処理をまったく透過的に障害から回復するシステムもあります。それぞれのシステムは、ますます危険が増える世の中で自身を保護する方法を知っています。いくつかのシステムは、FibreChannel を介した複製によってそれを実現し、別のシステムはデータベースのデータを全国に配置することにより、さらに別のシステムは、何百マイルもの距離に及ぶアクティブ・アクティブ型のマルチ・サイト・クラスタによりそれを実現します。

読者自身が技術者として、これらの技術を理解し、業務に適合するものを選ぶ必要があります。しかし、読者は気付いていないかもしれませんが、それ以外にもやる事があります。つまり、選択した技術の正確な能力を経営上層部に知ってもらうことです。なぜなら、耐災害性のない 2 ノード構成の手動フェイルオーバ・システムを実装しただけなのにもかかわらず、経営上層部が、主たるデータセンタが予期せず壊滅的な状態になっても待ち時間なしで回復が可能な無限に拡張できるフォールト・トレラント・システムを実装したものと考えていたとしたら、非常にまずい状態になるからです。そして、その状態が発覚するのは、実装したとおりにシステムが動作したにもかかわらず、経営上層部が考えていたようには動作しなかったと彼らが知ったときなのです。

その対策としては、選択したソリューションが何をして何をしないのかを正確に文書化し、それが経営上層部の考えているとおりの動作であるという同意を彼らから得ることです。そして、もっと多くのことができるようにしたいと言ってきたら、そのために必要な予算の確保をお願いするのです。いずれの場合も、お選びになったソリューションを設計して実装するのに必要なソフトウェアとサービスを提供することが可能ですので、弊社までぜひご相談ください。

謝辞

Kirk Bresniker、Dan Cox、Jon Harms、Keith Parris、Bob Sauers、および Wesley Sawyer の貴重な助言と本文書のレビューに対して感謝します。

関連情報

LifeKeeper for Linux に関して

- HP のサーバと Linux に関する一般的な情報については、~~hp.com/linux~~ およびhttp://www.hpe.com/jp/linux (日本語)
- HP のサーバと Linux およびハイ・アベイラビリティ・ソリューションに関する一般的な情報については、~~h18000.www1.hp.com/solutions/enterprise/highavailability/linux/index.html~~
- Lustre については、~~hp.com/hpinfo/newsroom/press/08aug02b.htm~~
- ProLiant サーバにおける LifeKeeper の仕様については、~~h18000.www1.hp.com/Solutions/enterprise/highavailability/linux/description.html#specs~~
- ~~linux-ha.org~~
    - “What Linux-HA Can Do Now”
    - “LAN Mirroring Technologies”
- Mission Critical Linux のホワイトペーパーは、~~missioncriticallinux.com/~~
- Oracle と Linux については、http://technet.oracle.com/tech/linux/
- SteelEye LifeKeeper については、http://www.steeleye.com/products/linux/#2 および ~~steeleye.com/pdf/literature/lkpr4linux.pdf~~
- Service Monitoring Daemon については、~~kernel.org/software/mon/~~

- Monit ユーティリティについては、http://www.tildeslash.com/monit/

HP-UX および Linux 向けの Serviceguard に関して

- Serviceguard とハイ・アベイラビリティに関する一般的な情報については、
  ~~hp.com/go/ha~~
- DH Brown 2002 UNIX Function Review のレポート、
  ~~hp.com/products1/unix/operating/infolibrary/reports/2002Unix_report.pdf~~
- 耐災害高可用クラスタ技術について、
  ~~docs.hp.com/hpux/onlinedocs/ha/highly_avail_clust.pdf~~
- Information Library、
  ~~hp.com/products1/unix/highavailability/ar/mcserviceguard/infolibrary/index.html~~
    - 5nines Architecture Overview
    - System Cluster Technologies and Disaster Tolerance
    - Data Protection
- MirrorDisk/UX について、~~software.hp.com/cgi-bin/swdepot_parser.cgi/cgi/displayProductInfo.pl?productNumber=B2491BA~~

NonStop Kernel に関して

- NSK の詳細情報に関する Total Information Manager (TIM) 製品へのアクセスについて、
  ~~nonstop.compaq.com/view.asp?PAGE=TIM_Prod~~
- NSKとハイ・アベイラビリティの詳細について、
  ~~h71033.www7.hp.com/object/tdnsk3pd.html~~
- Remote DataCenter Facility について、~~h71033.www7.hp.com/page/RDF_SW.html~~

TruCluster に関して

- Tru64 UNIX と TruCluster については、~~h30097.www3.hp.com/~~ および
  ~~hp.com/jp/tru64unix~~
- TruCluster V5.1b のQuickSpecs については、
  ~~h18000.www1.hp.com/products/quickspecs/11444_div/11444_div.HTML~~および
  ~~h50146.www5.hp.com/products/software/oe/tru64unix/document/v51b/TCR/TCR_SPD.PDF~~
  (日本語)
- ドキュメントに関しては、~~h30097.www3.hp.com/docs/pub_page/cluster51B_list.html~~および
  ~~h50146.www5.hp.com/products/software/oe/tru64unix/document/v51b/DOCS/HTML/tcr_lib.html~~ (日本語)
  詳細は以下を参照してください。
    - ~~h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGVETE/TITLE.HTM~~、Cluster Technical Overview
    - ~~h50146.www5.hp.com/products/software/oe/tru64unix/document/v51b/TCR/TECHOVER/TITLE.HTM~~、クラスタ概要

- ~~h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGWETE/TITLE.HTM~~、Hardware Configuration
- ~~h50146.www5.hp.com/products/software/oe/tru64unix/document/v51b/TCR/HW_CONF/TITLE.HTM~~、クラスタ・ハードウェア構成ガイド

Section 1.3.1.4 Quorum Disk / 1.3.1.4 項 クォーラム・ディスク

- ~~h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGYETE/TITLE.HTM~~、Cluster Administration
- ~~h50146.www5.hp.com/products/software/oe/tru64unix/document/v51b/TCR/AD MIN/TITLE.HTM~~、クラスタ管理ガイド

Section 4.3 Overview of Calculating Cluster Quorum / 4.3 節 クラスタ・クォーラム・ポートの計算の概要

- ~~h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHH0ETE/TITLE.HTM~~、Highly Available Applications
- ~~h50146.www5.hp.com/products/software/oe/tru64unix/document/v51b/TCR/APP_GD/TITLE.HTM~~、高可用性アプリケーション・ガイド

Chapter 1, Cluster Applications / 第 1 章 クラスタ・アプリケーション

- QuickSpecs for the Logical Storage Manager V5.1b については、~~h18000.www1.hp.com/products/quickspecs/10899_div/10899_div.HTML~~
- AlphaServer SC ホーム・ページについては、~~hp.com/techservers/index.html~~

OpenVMS Cluster Software に関して

- OpenVMS と OpenVMS Cluster Software の一般情報については、https://support.hpe.com/hpesc/public/docDisplay?docId=a00058693en_us および https://www.hpe.com/jp/openvms (日本語)
- OpenVMS Cluster Software V7.3-1 については、~~h71000.www7.hp.com/openvms/products/clusters/index.html~~
- OpenVMS Cluster Software V7.3-1 SPD については、~~h18000.www1.hp.com/info/SP2978/SP2978PF.PDF~~
- ドキュメントに関しては、~~h71000.www7.hp.com/doc/index.html~~ およびhttps://www.hpe.com/jp/openvms-technical (日本語)

  詳細は以下を参照してください。

  - ~~h71000.www7.hp.com/doc/731FINAL/4477/4477PRO.HTML~~、OpenVMS Cluster Systems
  - https://www.hpe.com/jp/openvms-cluster、OpenVMS クラスタ・システム

Section 2.3.3 Quorum Algorithm / 2.3.3 項 クォーラム・アルゴリズム

の設定と管理 Chapter 7 Setting Up and Managing Cluster Queues / 第 7 章 クラスタ・キュー

- ~~h71000.www7.hp.com/doc/731FINAL/6318/6318PRO.HTML~~、Guidelines for OpenVMS Cluster Configurations
- https://www.hpe.com/jp/openvms-clusterクラスタ構成ガイド

Chapter 8, Configuring OpenVMS Clusters for High Availability / 第 8 章 可用性を目的とした OpenVMS Cluster の構成

Appendix D, Multi-Site OpenVMS Clusters / 付録 D マルチサイト OpenVMS Cluster

- ~~h71000.www7.hp.com/doc/731FINAL/5423/5423PRO.HTML~~、Volume Shadowing for OpenVMS

- https://www.hpe.com/jp/openvms-volume-shadowing、Volume Shadowing for OpenVMS 説明書 Section 1.5 で、耐災害用の WAN ベース RAID-1 を取り上げています

Windows 2000 に関して

- Windows 2000 DataCenter の一般的な情報について、~~microsoft.com/windows2000/en/DataCenter~~
- ドキュメントについて、~~microsoft.com/windows2000/en/DataCenter/help/~~。詳細は次を参照してください。
  - 「Choosing a Cluster Model」では、Windows 2000 DataCenter がマルチ・システム・ビュー・クラスタであることを強調し、シングル・システム・ビュー機能の欠如を指摘しています。
  - 「Checklist: Preparing a Server Cluster」では、クラスタ内の各システムが個別にシステム・ディスクを持つ必要があることを提示しています。
  - 「Server Clusters」では、4 台までのシステムでサーバ・クラスタを構成できるとあります。
  - 「Cluster Hardware and Drivers」では、ネットワークが唯一にクラスタ・インターコネクトであると言っています。
  - 「Quorum Disk」では、クォーラム・ディスクの利用法を示し「Cluster Database」では、
    クォーラム・ディスク上の回復ログに各システムのクラスタ・データベースがどのように書き込まれるのかを説明しています。
  - Windows Clustering、Server Clusters、How To…、Perform Advanced Administrative Tasks
  - Choosing a RAID Method
  - 「Disaster Protection」では、WAN RAID の欠如を説明しています。

シングル・システム・イメージの Linux に関して

- この HP プロジェクトについては、http://sourceforge.net/projects/ssic-linux

StorageWorks Continuous Access に関して

- StorageWorks のハイ・アベイラビリティ・ソリューションに関する一般的な情報については、~~h18006.www1.hp.com/storage/software.html~~
- QuickSpecs for StorageWorks Continuous Access については、~~compaq.com/products/quickspecs/10281_na/10281_na.html~~
- ~~h18006.www1.hp.com/products/storage/software/conaccesseva/index.html~~、SANworks Continuous Access Overview and Features

書籍

- "Clusters for High Availability", Peter Weygant, ISBN 0-13-089355-2
- "In Search of Clusters", Gregory F. Pfister, ISBN 0-13-899709-8