Hewlett Packard Enterprise

HP-UX 特別企画

特別企画では、HP-UX メジャーアップデートのご案内や、HP-UX を利用したシステム構築・ソリューションを提供されているパートナー様にご登場いただき、技術情報・取り組みのご紹介をします。

― 特別企画 ―

- HP-UX 25 周年特別企画
- HP-UX 11i v3　新世代ミッションクリティカル OS へ
- NEC が語る「HP-UX による高可用システム構築」

《 連載期間：2005 年 11 月～2009 年 4 月 》

―目次―

HP-UX 25 周年特別企画

HP-UX11i v3　新世代ミッションクリティカル OS へ

NEC が語る「HP-UX による高可用システム構築」

HP-UX 25 周年特別企画

―目次―

HP-UX の 25 年とこれから

～HP-UX25 周年記念イベントで語られたビジョン「仮想化のその先へ」～

ミッションクリティカル環境の代名詞といえるプラットフォーム「HP-UX」が誕生して 25 周年を迎えたことを記念して、日本ヒューレット・パッカードは 2009 年 2 月に「HP-UX25 周年記念イベント」を開催した。本稿では、同イベントで行われた講演やインタビューをもとに、HP-UX が「エンジニアリング用途向けの UNIX」から「IT 社会を支えるミッションクリティカルシ

ステム」へと成長を遂げた変遷を紹介したい。そして後半では、2007 年にリリースされた最新版「HP-UX 11i v3」を中心に、その最先端のパフォーマンスや管理性、そして仮想化技術にスポットを当てる。

HP-UX の誕生、そしてミッションクリティカル環境へ

HPE は 1982 年、FA 用途とオフィスコンピューター分野の双方をターゲットとした 32 ビットワークステーション「HP 9000 500 シリーズ」をリリース。そしてこの 500 シリーズ上で動作する UNIX OS として、翌年の 1983 年に「HP-UX 1.0」がリリースされた。当時の HP-UX の用途は、CAD/CAM や計測機器制御といった汎用の技術計算分野がほとんどであったが、リアルタイム制御や製造、通信分野といったミッションクリティカル用途への導入も、すでにこの当時から少しずつ始まっていたという。

図 1：最初の HP-UX マシン「HP 9000/520」（1982 年）

80 年代後半、HPE 最初の高可用性クラスター（HA クラスター）製品である「SwitchOver UX」、（現在の Serviceguard の前身）が登場する。これにより、HP-UX マシンが何らかの原因でダウンしたとしても、数分～数 10 秒で予備マシンにフェイルオーバー（切り替え）できるようになった。1995 年にはその後継となる HA クラスター製品「Serviceguard」が HP-UX 10.0 と合わせてリリースされた。

写真 1：デビッド・フライデンダール氏

この Serviceguard と HP-UX 10.0 の登場は、HP-UX がビジネス用途やミッションクリティカル環境へ参入する大きなターニングポイントとなった。30 年前に HPE に入社して以来、500 シリーズをはじめとする HP-UX 開発に長年携わり、現在は UNIX システム開発部門を統括するデビッド・フライデンダール氏はこう語る。「オープンシステムのデータベース製品や ERP 製品、OS、そしてハードウェアをユーザーが自由に選択し、ミッションクリティカルシステムを構築するという、まったく新しい市場が生まれました。これは 90 年代初めのもっとも大きなブレークスルーでした。」（フライデンダール氏）

1990 年代後半になると、HP-UX はいよいよメインフレームの牙城へと本格的に切り込んでいった。現在も続く「メインフレームの代替」への挑戦の始まりである。こうした大きな潮流に合わせて、HP-UX もミッションクリティカル環境としての性格を一段と強めていく。1997 年には 64 ビットに対応した HP-UX 11.0 がリリースされた。

図 2：64 ビット対応 HP-UX 11.0 を搭載した HP 9000 K クラス（1997 年）

そして 2003 年にはインテル® Itanium® 2 プロセッサー（コード名 Madison）を搭載し、HP-UX 11i v2 が稼働する新しいサーバー製品「Integrity サーバー」が HPE からリリースされた。この Integrity サーバーは HP 9000 サーバーの後継機であり、HP-UX に加えて Windows と Linux が動作するマルチ OS 対応の基幹サーバーである。

図 3：Integrity サーバーのラインアップ

常に変化を続ける HP-UX のダイナミズム

つづいては、2007 年 4 月にリリースされた最新バージョン「HP-UX 11i v3」の新機能や特徴に注目してみたい。フライデンダール氏は次のように述べる。「HP-UX 11i v3 を導入されたお客様からは、パフォーマンスの向上について大きな反響をいただいています。用途や環境にもよりますが、例えば『OS をバージョンアップしただけで SAP の性能が 25%向上した』といった声をよくいただきます。」

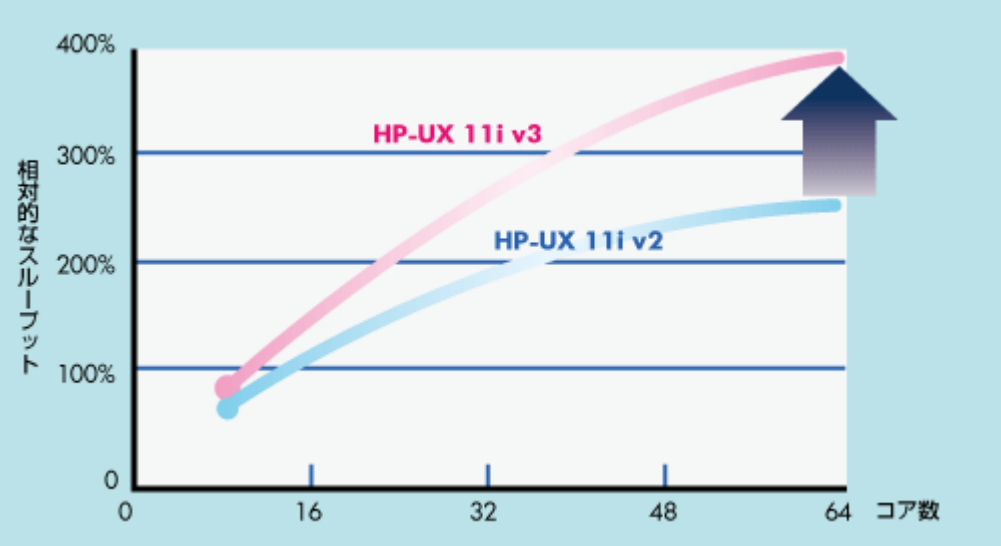


図 4：HP-UX 11i v3 によるパフォーマンス向上

また現在の HP-UX および Integrity サーバーが競合他社に対して有するもっとも大きなアドバンテージは、過去 8 年にわたって培われてきた仮想化ソリューション VSE（Virtual Server Environment）だ。HPE のエンタープライズサーバー＆ストレージ・ソフトウェア開発部門にて仮想化技術開発に携わるダン・ヘリントン氏は、「インテグレーション」の能力こそ、仮想化技術としての VSE の最大の特徴であると説明する。「例えば 1 台の Integrity サーバーの nPars 上で複数の vPars を稼働させ、数 10 個の OS インスタンスを同時に運用できます。新たに追加された Dynamic nPars 機能を使えば、それらを止めることなくサーバーのプロセッサーやメモリの交換が可能です。また Serviceguard と組み合わせ、フェイルオーバーが発生した場合には不足分のプロセッサーを瞬時に追加することも可能です。こうした複数の仮想化技術のインテグレーションによるソリューションは、業界でも HPE だけが提供するものです。」

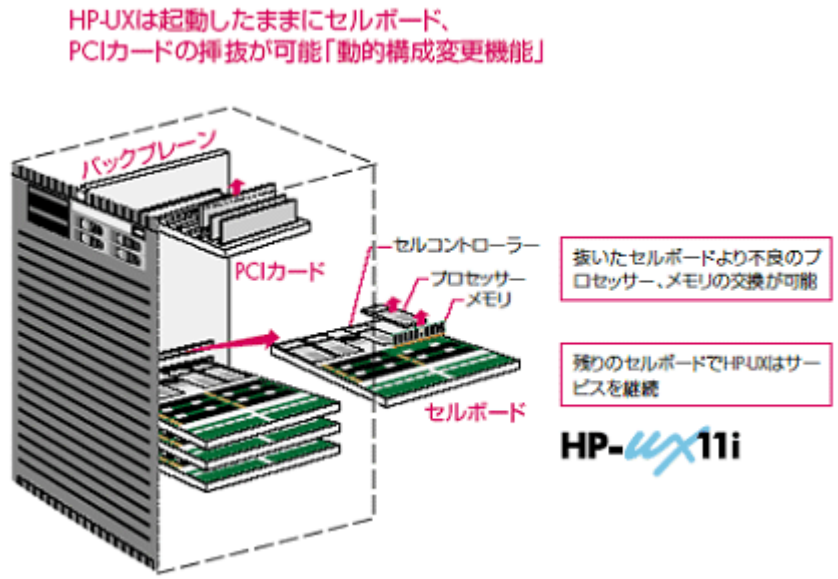


図 5：Dynamic nPars によるオンラインでのプロセッサー／メモリの交換

VSE の仮想化環境管理ツール「Insight Dynamics」


写真 2：ダン・ヘリントン氏

またヘリントン氏は、昨今の仮想化技術の普及によって仮想マシンが無数に増え続けている現状に警鐘を鳴らす。「いまや仮想マシンはクリック 1 つで作成できます。しかし多くの方は、それが"OS を管理する"というリスクとコストをともなうことに気づいていません。」とりわけ問題となるのはセキュリティリスクだという。「セキュリティパッチも適用されていない古い OS が稼働し続けており、ほかの誰もそれが何に使われていたのか知らないのです。このように、リスクの高い仮想マシンが不用意に増殖していく状況は『仮想化スプロール』と呼ばれています。」

ヘリントン氏は、この仮想化スプロールを回避する手段が、仮想化環境の管理ツールを活用することであると説明する。VSE では、こうした問題を回避するために「Insight Dynamics - VSE」（以下、「ID-VSE」）という仮想化環境の統合管理ツールを提供している。ID-VSE は、物理サーバーと仮想サーバーの違い、または HP-UX や Windows、Linux といった OS の違い、さらには ProLiant と Integrity のハードウェアの違い、VMware や Integrity VM といった仮想化技術の違いを意識せずに、すべてを一元管理できる。また、管理だけでなく、キャパシティプランニングの機能も統合されている。プロセッサーやメモリ、ネットワーク、ディスク、電源などの利用率を測定・収集し、それらのデータに基づいて、リソース配分が最適化されるような仮想マシンの構成をシミュレートできる。また、想定される複数のシナリオ間でリソースの利用率やエネルギーコストを簡単に比較可能だ。

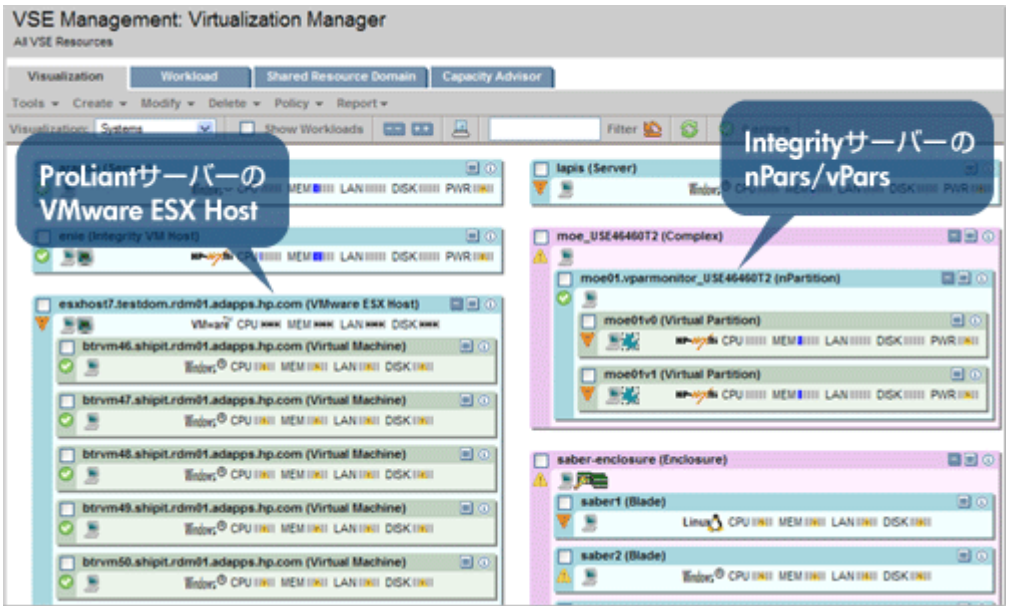


図 6：Insight Dynamics-VSE のコンポーネント： Virtualization Manager

ID-VSE の大きな特徴は、新しく導入された「Logical Server」機能により、物理サーバーと仮想サーバーを透過的に一元管理できる点だ。CPU、メモリ、ストレージといったサーバーを構成する情報を Logical Server という形式で管理し、その設置や移動を ID-VSE 上のメニューやドラッグ＆ドロップといった簡単な操作で行える。またブレードサーバーBladeSystem c-class の「バーチャルコネクト」機能とも連動し、ネットワークの MAC アドレス変更等も容易に実施できる。

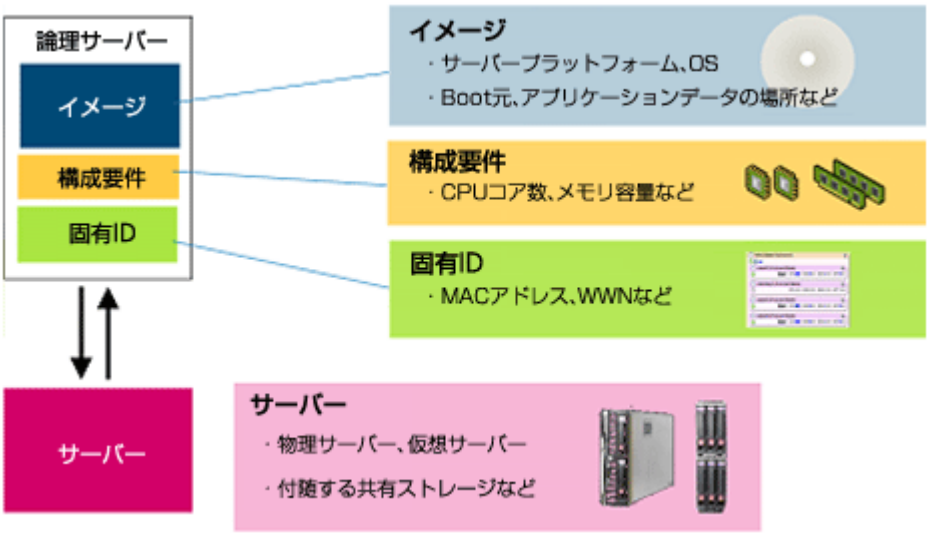


図 7：Logical Server の概念

HP-UX 11i v3 のセキュリティ

次に、HP-UX 11i v3 のセキュリティ機能に注目したい。国際的なセキュリティ資格である CISSP を有し、ミッションクリティカルシステム分野でセキュリティコンサルティングに携わるダグラス・ラモルー氏が HP-UX 11i v3 のセキュリティ機能として活用を推奨しているのが、監査機能「audsys」である。

この audsys は「システムコールレベルで監査ログを記録する」機能だ。HP-UX 上で実行されたすべてのプロセスのシステムコール（例えばファイルのオープンや読み書きなど）をリアルタイムに記録できるため、例えば「スクリプト内部で実行された処理は記録できない」といった“取りこぼし”は発生しない。また、監査ログはバイナリ形式で記録され、改ざんは困難である。

写真 3：ダグラス・ラモルー氏

この audsys で収集したログデータは、PCI DSS や SOX といった個々のセキュリティ規格に沿って分析し、レポートを生成することができる。「PCI DSS ではクレジットカード情報や個人情報といった特定のリソースについて監視や監査を求めていますが、HP-UX 上のすべての動きを監視する必要はありません。v3 の新機能である監査レポートツールを使えば、audsys が出力する監査ログから、例えば PCI DSS で必要な情報のみフィルタリングして Web ベースのレポートを生成できます。このレポートはカスタマイズも可能なので、個々のシステムに最適なレポートを組み立てることも可能です。」（ラモルー氏）

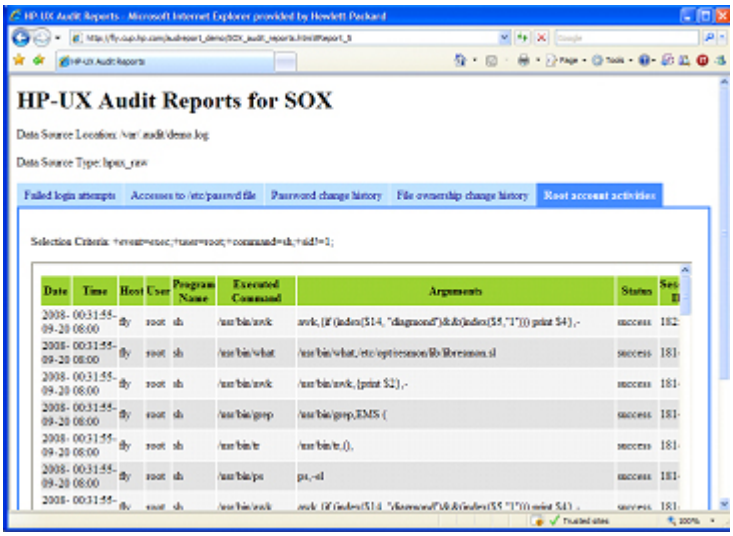


図 8：audsys の監査データから生成された SOX 対応レポート

そしてウルトラアダプティブコンピューティングへ

最後にフライデンダール氏は、「ウルトラアダプティブコンピューティング」という言葉をキーワードに、「未来の HP-UX」のビジョンを説明してくれた。「いまの企業の IT システムが抱える大きな課題は、IT システムの運用に大きなコストとスキルが要求されることです。そのため今後は、この問題を解決する手段として『自動化』が重要な意味を持つでしょう。」

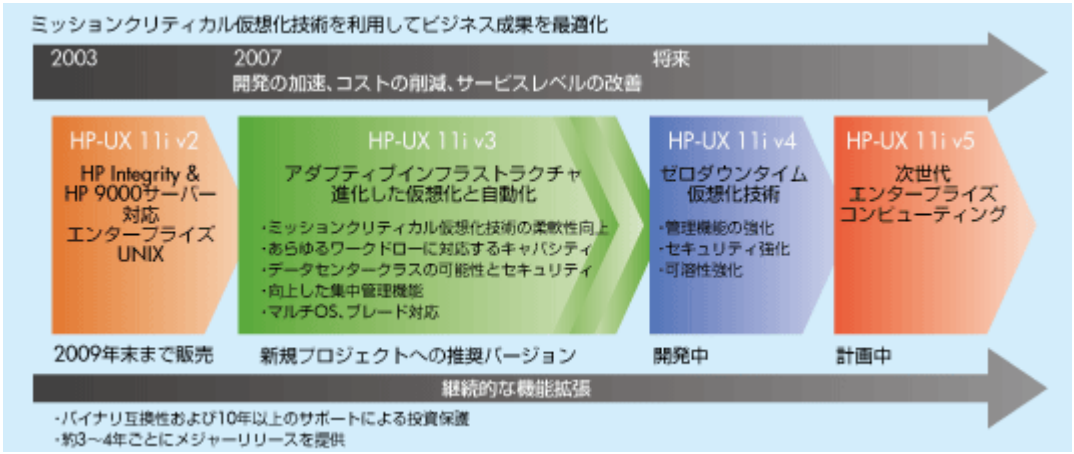


図 9：HP-UX の今後のロードマップ

「近い将来の OS は、完全な自己調整能力を獲得するはずです。『いまどのような種類のプロセスやスレッドを実行しているのか』、『どのようなスケジューリングアルゴリズムが最適なのか』といった点を動的に判断し、OS 全体を最適化する仕組みが実現されるでしょう。こうしたウルトラアダプティブコンピューティングのビジョンに基づいて、IT システムのコストとスキルの問題を解決していきたいと考えています。」（フライデンダール氏）

HP-UX25 周年特別企画「仮想化のその先に～コスト削減と可用性を求めて～」

2008 年 12 月　テクニカルライター 米田 聡

「基幹業務向けの信頼性のある UNIX」を追求し続けた HP-UX の 25 年。Serviceguard という強力な HA システムを擁することが HP-UX の大きな強みであることは、今さら言うまでもないだろう。最新の HP-UX 11i v3 には HA システムの構築を支援するさまざまなソリューションが用意されており、可用性とコスト削減の両立を実現している。今後、HP-UX がどこに向かおうとしているのか、今回は HPE 内部で行われたテストの結果などをも交えつつ、コストを抑えた HA システムを探ることにしたい。

進化する HP-UX、仮想化のその先に

今年 9 月に Integrity Virtual Machine バージョン 4.0 (以下、Integrity VM)がリリースされたのは以前の特集でお知らせした通り。ホスト OS として HP-UX 11i v3 がサポートされ、Integrity の可用性や拡張性を最大限に生かし、HP-UX の仮想は着実な進化を遂げている。また、2010 年には HP-UX の次のメジャーバージョンである HP-UX 11i v4 の投入も予定されており、可用性と仮想化の融合のさらなる進化が待っている。

そもそも HPE の目指す可用性と仮想化の融合とは何なのだろうか。また、現在 HP-UX はどこまで進化しているのだろうか。

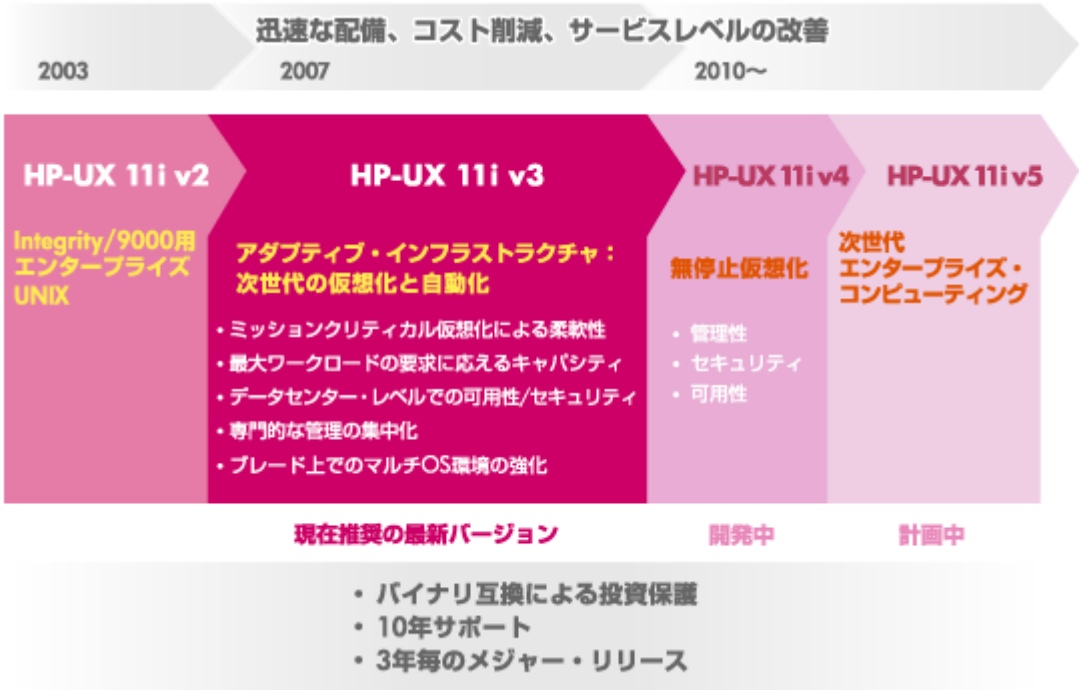


図 1：HP-UX 11i ソフトウェア ロードマップ

コスト削減と可用性の両立とは

HA システムと一口に言っても、要求される信頼性の程度はシステムによって異なる。わずかな間のサービス停止も社業の命運に関わるシステムであれば十分なコストをかけた堅牢な HA システムが求められるだろう。だが、すべてのシステムにそこまでの信

頼性が必要かと言えば、そうではないのも確かだ。たとえば、2 基のサーバを運用しているとしよう。待機系にも 2 基のサーバを用意すれば万全ではある。だが、ハードウェアや OS の信頼性が向上により、2 基のサーバが同時に障害を発生させることは極めて稀になってきた。となれば、待機系には仮想化技術を用いて 2 台分のバックアップを 1 台のサーバでまかなうという方法も、選択肢として候補に挙がってくる。ここに、コスト削減と可用性の両立への解答がある。

HP-UX では、他の OS と異なる特長のひとつ、Firmware を用いたソフトウェアレベルのパーティショニング技術 vPar が利用できる。vPar 間では CPU リソースの（また後にふれるが、最新バージョンではメモリリソースも）柔軟な再配分が可能だ。仮に 2 台のサーバに障害が生じるという万が一の事態が発生しても、vPar を用いればサービス品質の低下を最小限に抑えながらサービスの継続が可能になる。さらに、プリペイド制による CPU の拡張サービス TiCAP を組み合わせれば、さらなるコスト抑制が可能だ。待機系の CPU を最小限に抑えておき、障害発生時には TiCAP の CPU を使うことで待機系にかかるコストを抑えると同時に、サービス品質の低下も抑えられるわけだ。

このように、HPE の柔軟なソリューションを活用することでコストを抑えた HA システムの構築が可能だが、もちろん万全な HA システムに比較すればデメリットも生じうる。システムを構築する際にしっかりと押えておかなければならないのは、コストを抑えたことと引き替えに生じるデメリットが許容できる範囲であるかどうかだろう。以降で、この点にさらに踏み込んで検討していくことにしたい。

待機系に vPar＋TiCAP を活用する

まず、HA システムを一般的に検討されるシステム構成に、更に HPE の仮想化技術を取り入れ、3 つのソリューションパターンに分けてそのメリットとデメリットを考えてみた。

表 1：コスト順に 3 段階で分けた HPE のソリューション

案	ソリューション	メリット	デメリット
松	Active-Standby 間で同じリソースを用意する従来型の SG クラスタ	フェイルオーバー時にリソースの変更処理が入らないので、安定した迅速なフェイルオーバーが可能	通常運用時稼動しない余分なリソースを用意する必要がある
竹（今回の検証）	TiCAP+vPar を使用し、Active-Standby 間で通常稼動時のリソースが異なる SG クラスタ	通常運用時に稼動しない余分なリソースが比較的少なくてすむオンラインでのリソース追加によってフェイルオーバー時には Active と同等のリソースを確保できる	フェイルオーバー時にリソース追加処理を行なうことでフェイルオーバー速度が遅くなる懸念点がある（検証項目）
梅	SAN Boot 構成を使用し、サーバの障害時に複数の Active に対して 1 台以上の Cold Standby のサーバを用意する N+1 構成	通常運用時に稼動しないリソースが最小限ですむ	Cold Standby であるのでフェイルオーバーに時間がかかる

	同時障害に耐えられない可能性がある

最上位の「松」は、待機系にも稼働しているサーバと同じリソースを持たせるという、Serviceguard を用いた HA クラスタの基本的な構成だ。障害発生時にはアクティブな状態を維持する待機系へと迅速にフェイルオーバーが可能であり、また同じリソースを持つ待機系によりサービス品質の低下も発生しない。

一方、もっともコストを抑えた構成である「梅」は、SAN Boot を使って 1 台のサーバをコールドな状態で待機させておく、というもの。障害発生時にはコールド状態からブートさせ、フェイルオーバーを行うわけだ。待機サーバを 1 台に押え、コールドスタンバイさせればコストは最小にすむ。ただしフェイルオーバーには時間がかかり、サービスは一時的に停止するだろう。また、待機サーバが 1 台しかなければ複数のサーバに同時に障害が起こったときに対応できない可能性が高い。停止することが許されないようなサービスには利用しづらいだろう。

「竹」は、松と梅との間を取った HA システムだ。待機系は稼働しているサーバよりもリソースを抑えた構成にし、ホットスタンバイさせておく。また、前節でふれたように vPar＋TiCAP を使う前提で通常は最小の CPU で稼働させる。障害が発生したら待機系にフェイルオーバーすると同時に、TiCAP を使って vPar にリソースを追加、稼働しているサーバと同じ程度のリソースを与えることでサービスを継続する、というものだ。待機系にかかるコストが、うまく構築すれば梅＋α程度ですみ、なおかつ障害発生時にもサービスの品質の低下は抑えられるということで魅力的な選択肢だろう。

もちろん懸念すべきことはある。フェイルオーバーの速度が松に比べてどの程度、遅くなるのか、という点だ。最小の CPU に抑えている状態から、フェイルオーバーと同時に TiCAP を用いて増加させるのだから、松に比べればフェイルオーバーに時間がかかるはずである。そこで後半では、実際の検証結果をお目にかけることにしたい。なお、この検証は HPE が実際に顧客へシステム構築を行うにあたって、製品部隊の技術者と構築部隊の技術者が、顧客のシステムを想定して行ったフィジビリティスタディの一部となっている。

TiCAP＋vPar を使った HA システムの検証

検証したシステムは、1 台の稼働サーバに 2 つの vPar が設定され、待機サーバにも 2 つの vPar が設定されている。稼働サーバの 1 つの vPar が停止、Serviceguard によって待機サーバの vPar にフェイルオーバーさせ、その時間などをテストした。待機サーバの vPar は、通常時は最小のリソースでホットスタンバイした状態にあり、フェイルオーバーと同時に TiCAP を使って CPU を追加する。

まず、Serviceguard Extension for Faster Failover（SGeFF）を用いた場合と、用いない場合を比較した結果を示そう。ご存じの読者が多いと思うが、SGeFF は高速なフェイルオーバーを実現するオプションで、先の松のようなシステムであれば数秒にまでフェイルオーバーを短縮することができる。

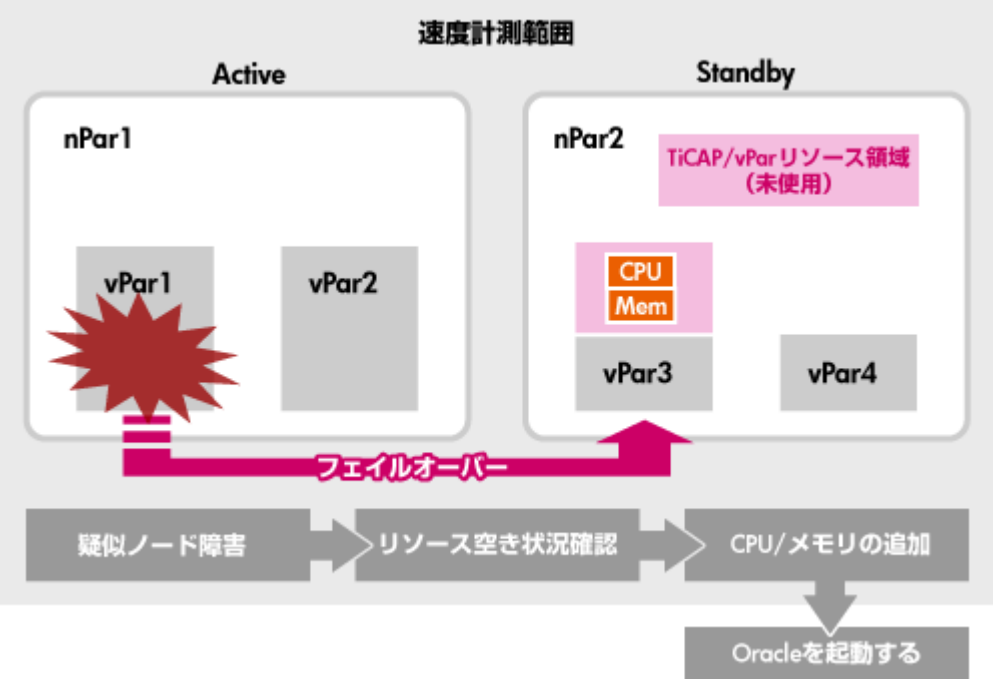


図 2：検証したシステム

表 2：SGeFF を有効にした場合と無効にした場合の検証結果

	FO 時間[s]			
	1 回目	2 回目	3 回目	平均
SGeFF 有効	23	23	22	23
SGeFF 無効	47	45	46	46

結果が示すように、SGeFF により約 23 秒の短縮が可能になっている。竹システムでも SGeFF は有効であることがわかるだろう。では、この 23 秒のうち CPU の増加には、どの程度の時間がかかるのだろうか。

表 3：CPU を増加させた場合の速度計測結果

	CPU コア数		Memory 容量[MB]	コマンド実行時間[s]			
	FO 前	FO 後		1 回目	2 回目	3 回目	平均
CPU の増加	1	4	4096	2.45	2.33	2.24	2.34
CPU の増加	1	8	4096	2.46	2.46	2.27	2.40
CPU の増加	1	12	4096	4.04	3.45	3.38	3.62
CPU の増加	1	15	4096	4.35	4.36	4.27	4.33

CPU の増加	1	4	11264	2.31	2.30	2.29	2.30
CPU の増加	1	8	11264	2.47	2.50	2.35	2.44
CPU の増加	1	12	11264	3.59	3.44	3.47	3.50
CPU の増加	1	15	11264	4.83	4.51	4.16	4.50

結果は上の通り。CPU の追加は CPU 数が増えるほど時間がかかるものの、数秒程度におさまっている。CPU がサーバの Cell をまたぐような場合、かかる時間はやや増加するが、それでも数秒以内で終了する。

最後に、vPar にメモリを追加する例を掲載しておこう。以前の vPar では CPU のみ動的な追加・削除が可能だったが、最新の vPar 5.0 ではメモリの動的な追加・削除が可能になっている。これにより、待機系の vPar の柔軟な運用が可能になる。

表 4：vPar によるメモリ追加時の計測結果

	CPU コア数		Memory 容量[MB]		コマンド実行時間[s]			
	FO 前	FO 後	FO 前	FO 後	1 回目	2 回目	3 回目	平均
メモリの増加	1	-	4096	6144	2.61	2.25	2.30	2.39
メモリの増加	1	-	4096	11264	3.74	3.98	4.03	3.92
メモリの増加	15	-	4096	6144	1.50	1.37	1.36	1.41
メモリの増加	15	-	4096	11264	2.54	2.52	2.56	2.54

vPar 5.0 を用いて仮想環境のメモリを動的に増加させた結果は上の通りだ。追加するメモリの容量が大きいほど時間がかかるが、それでも数秒以内に終了する。メモリに関しては増加させれば OS が自動的に使用するので、CPU のような再配分も必要はない。

ただし、メモリの削除にはケースバイケースで時間がかかることに注意が必要だろう。メモリを削除してもすぐにはメモリの返却は行われず、プロセスが終了するなどしてメモリを OS に返したタイミングでメモリが削減されていくためだ。

要求される可用性とのバランスが重要

以上、TiCAP＋vPar を用いた簡易クラスタの検証例を紹介した。正道の Serviceguard クラスタに比べればフェイルオーバーに時間がかかるのはやむをえないが、検証例で見た程度の遅延が許容できるシステムであれば魅力的な選択肢になるだろう。TiCAP は 30 日単位のプリペイドとなっており、障害発生時のみ稼働する待機系で利用するのであればプリペイドを使い切るという心配はほとんどないはずだ。TiCAP＋vPar を用いた HA クラスタはすでにいくつかの企業に導入されており、実績も十分にある。

ただし、迅速なフェイルオーバーが要求される HA システムには適しているとは言えないのも、また確かなことだ。システムに要求されている可用性のレベルとの兼ね合いが重要だろう。コストと可用性のバランスを十分に考えて検討する必要がある。

今回のケースは実際に顧客へのシステム提供にあたり、HPE の製品技術者と構築技術者が共同で、システムの重要度から求められる可用性とコストを念頭に最適化を行った事例である。HP-UX の仮想化技術により、可用性を保ちながらコストメリットのある柔軟な構成が組める様になったと言える。

HP-UX25 周年特別企画「HP-UX がエンタープライズ UNIX である 25 年の蓄積」

2008 年 11 月　テクニカルライター 米田 聡

2008 年、HP-UX は誕生から 25 年を迎えた。これまで多くの実績を築いてきた HP-UX だが、25 周年を機に HP-UX がたどってきた歴史を振り返ってみることにしたい。1983 年にリリースされて以来、一貫して「基幹業務向けの信頼性のある UNIX」を追求し続けた HP-UX の足跡には、最新バージョン HP-UX 11i v3 にいたるまで引き継がれてきたさまざまな長所の源流を見ることができる。

HP-UX はこうして始まった

1960 年代にベル研究所のケン・トンプソンらによって開発された UNIX は、version 7（1979 年リリース）に至るまでソースコードとともに頒布され、1980 年代初頭には教育機関や研究機関に広く普及していた。1980 年代初頭、HPE は 32 ビットワークステーション「HP 9000」をスタートさせる。9000 シリーズの当時の主要なターゲットは、おもに 2 つの方向があった。高い信頼性が求められるオフィスコンピュータ分野と、計測器の制御・計測データ収集を含むファクトリー・オートメーション（以下、FA）の分野だ。いずれも HPE 製品が得意としていた分野である。

写真 1：初期リリース時の様子

当初の 9000 シリーズは、当時のパーソナルコンピュータなどで広く普及していた BASIC や、Pascal、MODCAL などを開発言語としてサポートするコンピュータだった。MODCAL/Pascal で記述された HPE のオリジナルのカーネルがあり、その上で BASIC インタプリタが動作していたが、HPE は 9000 シリーズの将来の OS として UNIX を採用することを決定、開発をスタートさせる。そして 1982 年に発売された HP 9000 シリーズ 500 をターゲットに、HP-UX 最初のバージョン 1.0 が 1983 年に

リリースされた。初期の HP-UX は BSD のソースコードを参考に前述したオリジナルのカーネルの上で動作する OS で、ユーザは BAISC と使い分けることが可能だった。

その後、9000 シリーズの発展とともに HP-UX はやや複雑な進化を遂げていくことになる。当時の HPE では、オフィス向けのコンピュータを推進するワークステーション事業部と、FA 向けを主軸とするサーバ事業部という 2 つの事業部において、それぞれ独立して 9000 シリーズの開発が進められていた。前者は 1990 年代初頭までに CPU としてモトローラ 680x0 を採用するシリーズ 200、300、400 を展開。一方、後者では独自のプロセッサ「PA-RISC」を搭載するシリーズ 800 が開発された。

HP-UX は両事業部で開発が進められたが、採用しているプロセッサが異なること、また要求されている性能や機能が異なることもあって、カーネルのソースコードも異なるものだった。特にサーバ事業部では FA での利用が主だったことからリアルタイム性能を要求される。そのため、割り込み応答時間が保証されたリアルタイムカーネルを採用した、現在の HP-RT につながるカーネルが使用されていたのである。

写真 2：9000 シリーズ 500

だが、互換性の向上を計る動きがなかったわけではない。バージョンが進むごとに両者の統一が図られ、HP-UX 7.0 で主要コマンドが共通化された。そして、1992 年にワークステーション事業部はプロセッサに PA-RISC を採用するシリーズ 700 を投入。サーバ事業部とワークステーション事業部のプロセッサが統一されたことで、HP-UX 9.0（同年リリース）において完全な統合が果たされたのだった。

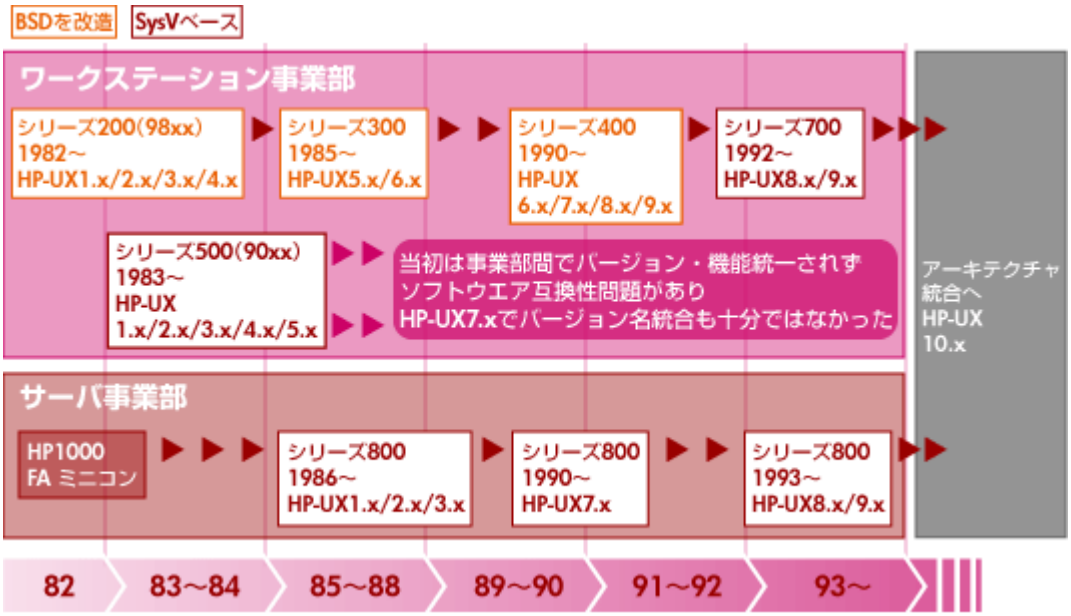


図 1：2 つの事業部で開発されていた HP-UX が統合されるまでの流れ

UNIX の戦国時代と HP-UX の立ち位置

さて、1990 年代初頭までの HP-UX がたどってきた道のりを概観したが、ここで目を外部に転じてみよう。実は HP-UX の歴史は、オープン化の歴史とも重なっているのだ。1980 年代の終わりごろ、UNIX 周辺には大きな変化が訪れていた。初期の UNIX は大きく AT&T の System V 系と、UNIX Version 7 から分岐した BSD UNIX 系とに分かれていたが、AT&T と Sun Microsystems が UNIX の統一を図るべく共同で System V Release 4（SVR4）を開発する。

だが、SVR4 を開発した両社が UNIX への支配を強めることに危機感を抱いた UNIX ベンダー各社が、1988 年に「Open Software Foundation（以下、OSF）」を設立。OSF は Mach マイクロカーネルをベースとした標準 UNIX システム OSF/1 をリリースする。同年、Sun Microsystems と AT&T もこれに対抗するかたちで「UNIX International（以下、UI）」を結成。以後、数年間にわたり UI と OSF とが激しく対立することになった。


写真 3：9000 シリーズ 712

HPE は OSF に加盟して UNIX の標準化を推進すると同時に、PA-RISC を採用する UNIX ワークステーションのベンダーによる標準化団体「Precision RISC Organization（以下、PRO）」を設立、PA-RISC 上で動作するソフトウェアの標準化を強力に推

し進めた。PRO には日立製作所、三菱電機、沖電気など HP-UX の代表的なベンダーが加盟している。現在の HP-UX の大きな特徴であり、また大きな強みでもあるマルチベンダー体制は、この PRO によって基礎が築かれたわけである。

さて、OSF と UI の両陣営が激しく対立したいわゆる「UNIX 戦争」は、1990 年代半ばに始まった環境の変化もあって、21 世紀に入る前に終息することになる。1994 年にパーソナルコンピュータの性能向上に合わせ、マイクロソフトがエンタープライズ分野を狙うオペレーティングシステム Windows NT を投入。UNIX 陣営も共同で新勢力に対抗する必要に迫られた。それを主導したのが HPE を含む OSF だ。1993 年、UI と OSF が共同で「Common Open Software Environment（以下、COSE）」を結成。COSE により UNIX の標準化が図られ、翌 1994 年に UI が OSF に合流した。さらに 1996 年、ユーザインタフェースの規格策定などを行っていた「X/Open」と OSF が合併し、現在の「The Open Group」が誕生することになった。

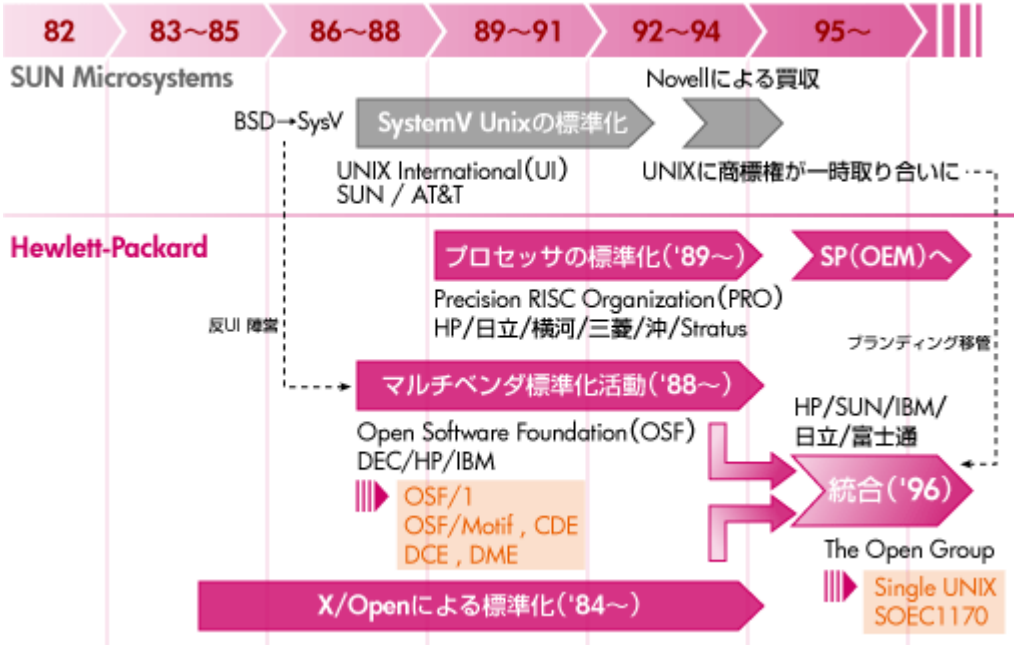


図 2：1980 年代から 1990 年代にかけての UNIX 戦国時代

基幹業務向けの UNIX を先駆けて実現した HP-UX

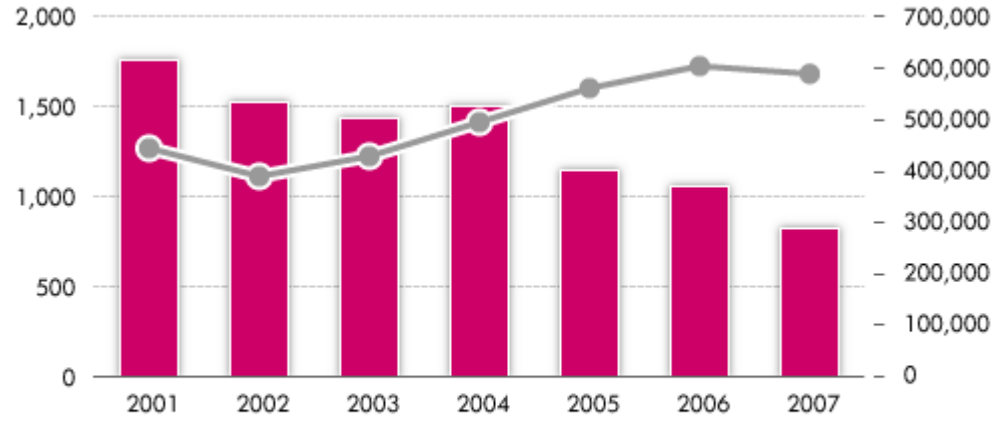
HP-UX とそれを取り巻く歴史を概観したが、初期の UNIX がソースコードと共に頒布された経緯もあり、UNIX はどちらかといえば学術色の強い OS だったと言っていいだろう。そのためか初期の UNIX では信頼性、安定性といった要素が今ほどは重視されていなかった。

実際、1980 年代に UNIX はワークステーションを中心に広がったものの、ミッションクリティカルな分野に UNIX が利用される例はほとんどなかった。UNIX はメインフレームの OS に比べて信頼性に劣るというのが当時の一般的な認識であって、UNIX はエンジニアリング分野やコンピュータサイエンス向きの OS と捕えられていた。

そうした中で、HPE の UNIX との関わり方は独特なものだったと言える。HPE は創業当時から、計測器を始めとする高い信頼性が要求される分野で実績を築いてきた。HPE のコンピュータは計測器の周辺ツールとしてスタートしており、当初から信頼性・安定性といったものを重視して設計が進められたのである。9000 シリーズの OS としてスタートした HP-UX にも、当然ながら高い信頼性が要求された。UNIX に関わる多くの企業がコンピュータベンダーとして UNIX に関わっていた中で、HPE はあくまでユーザとして UNIX に関わってきたと言い換えてもいいだろう。HPE のこの姿勢が、HP-UX の大きな特徴を形作ってゆく。

写真 4：9000 シリーズ 800

1992 年には 9000 シリーズ 800 がテレコム分野に導入され、高信頼性への要求に答えて HP-UX とともに多くの実績を築いた。1990 年代半ばにオープンシステムに注目が集まるが、それよりも早い時期に HP-UX は基幹システムへの足がかりを築いていたわけだ。1995 年にリリースされた HP-UX 10.0 では、現在の Logical Volume Manager（LVM）が導入されたほか、後にクラスタシステムがサポートされている。HP-UX 10.0 と Serviceguard は高可用性を実現する UNIX として高い評価を得、金融などさまざまな分野の基幹システムへの導入が加速した。また、2001 年に投入された Superdome では、ハードウェア仮想化としてパーティショニング技術を投入、仮想化への第一歩を踏み出している。



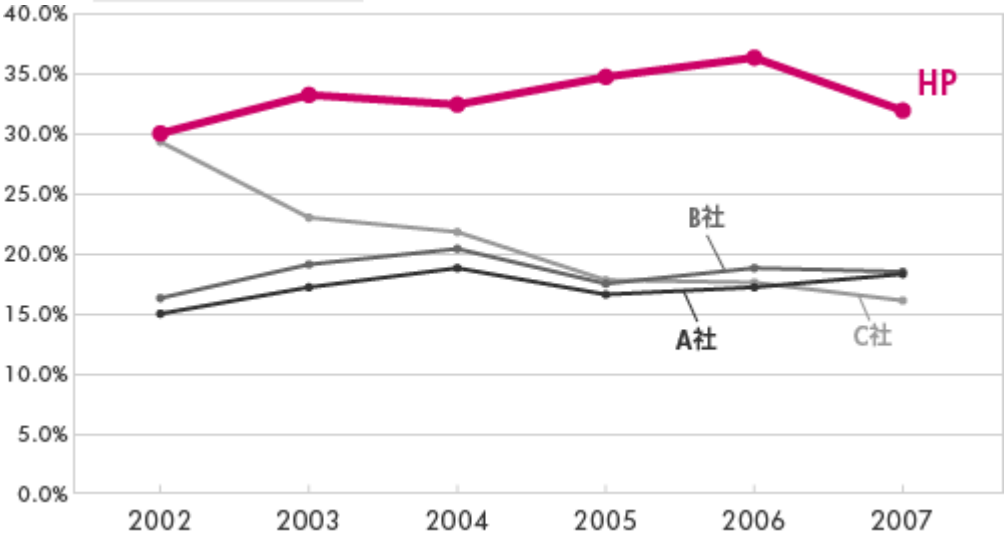


図 3：日本サーバ市場　出荷台数推移(上)と出荷金額推移(下)

出展: Unix OS/RISC & EPIC（IA-64）Server, Revenue Share IDC Worldwide Quarterly Server Tracker, Q407

以上のように、HP-UX はエンタープライズ分野に必要とされる機能を他に先駆けて取り入れてきたが、これは HPE がコンピュータを手がけ始めた当時から「コンピュータのためのコンピュータ」ではなく「使うためのコンピュータ」を一貫して追求してきたことが背景にあるわけだ。

過去から未来へ〜HP-UX のこれから

ワールドワイドで展開されている HP-UX は、多くのユーザによるフィードバックが得られるのはもちろんのこと、実は HPE の IT 部門自身が HP-UX の最大のユーザでもある。市場に出て行く HP-UX の新機能は HPE のデータセンターでも実際に使用され、信頼性・可用性等あらゆる面で厳しい要求にさらされている。それらに応えながら、HPE の「使うためのコンピュータ」の追求が続いていくはずだ。

表 1：HP-UX 9.0 リリース以降、HP-UX 11i v3 までの歩み

1992 年	HP-UX 9.0
1995 年	HP-UX 10.x　MC/Serviceguard
1997 年	HP-UX 11.0 64bit 化
2000 年	HP-UX 11i v1 ミッションクリティカル分野への本格的な対応
2002 年	HP-UX 11i v1.6 Itanium 対応
2003 年	HP-UX 11i v2 Integrity サーバでの本格的なエンタープライズ向け環境を整える
2007 年	HP-UX 11i v3 信頼性・可用性・セキュリティが向上し、仮想化環境がより充実

HP-UX11i v3　新世代ミッションクリティカル OS へ

―目次―

HP-UX のようにミッションクリティカルな分野で使用されることが多い OS では、システムの不具合やセキュリティ上の脆弱性について常に気を配る必要があります。そこで重要になるのがパッチの管理です。今回は、2007 年度に新たにリリースされた管理ツールを中心に、HP-UX におけるソフトウェア・パッチの管理について解説していきます。

稼働中のシステムイメージのダイナミックな複製を可能にする DRD（2007 年 8 月）

安定したシステム運用のためには、適切なパッチの適用やソフトウェアのアップデートなどのメンテナンス作業が不可欠です。後編では、稼働中のシステムのイメージをクローン化することで、パッチ評価用のテスト環境の構築を可能にし、さらにシステムのメンテナンス時におけるダウンタイムを短縮する Dynamic Root Disk（DRD）について紹介します。

サーバーを止めずに増強できる Dynamic nPar の離れ業（2007 年 9 月）

HPE が今回発表した新技術「Dynamic nPartitions」の最大の特徴は、アプリケーションの動作を止めることなく、Integrity サーバーのプロセッサーやメモリを増設できる点です。「サーバーの台数の追加」ではなく、1 台のサーバーのプロセッサー数やメモリ容量を、アプリケーションの稼働中に動的に増やせるので、サーバー増強にともなうサービス停止の頻度を劇的に少なくすることができます。

容易に扱える Dynamic nPar の実行例（2007 年 9 月）

ハイエンドやミドルレンジのサーバーでは、サーバー全体のリブートやサービスの停止がとりわけ困難です。そうした環境では、Dynamic nPar のオンラインでのメンテナンスやパーティション構成変更が有用です。新たに導入された概念やコマンド操作もそれほど難しくはありません。後編では、Dynamic nPar のコマンド操作を、実際の作業の流れに沿って紹介していきます。

柔軟性、信頼性、管理性を強化した HP-UX11i v3 の全貌

2007 年 4 月　テクニカルライター 秋葉けんた

2007 年 4 月 11 日、国内 UNIX サーバー市場で 6 年連続 No.1 のシェアを誇る HP-UX の最新バージョン、HP-UX 11i v3 がついに登場した。この HP-UX 11i v3 はメジャーバージョンとしてリリースされた。「v2」から「v3」へと一見マイナー・チェンジのように見えるが、それはバージョン間の互換性を意識したため、あえてこのバージョン名で市場へ投入した。その中身を見ると、パフォーマンスからシステムの信頼性を高める機能と革新的な仮想化や管理をはじめとする機能まで、いたるところに新しい技術が取り込まれ更に完成度を高めている。まさに、次世代のインフラを支えるエンタープライズ OS と言っても過言ではない。そこで HP-UX Knowledge-on-Demand では、特別企画「HP-UX11i v3 新世代ミッションクリティカル OS へ」と称して、これから 6 回に渡り、その全貌と詳細について、技術的にご紹介したいと思う。

HP-UX が提供する「実績」とは？

最新バージョンの HP-UX 11i v3 を語る前に、まず、HP-UX について触れておきたい。HP-UX は、国内 UNIX サーバー市場で 6 年連続 No.1 のシェアを誇るエンタープライズ OS で、この実績は「信頼」の代名詞と言っても過言ではない。長年企業の中核を支えるシステムとして導入されてきた歴史的な背景と、そこで培われた HPE の経験が、HP-UX の「信頼」の高さを物語っている。

「実績」の信頼技術

HP-UX は 1983 年に産声を上げ、今年で 24 年目を迎える。当時、基幹システムと言えば、メインフレームやベンダー独自 OS が主流だったが、日本ヒューレット・パッカードはそれらの自由度および将来性の低さをいち早く察知し、オープン化という潮流の草分け的な存在として「HP-UX」を開発した。それ以降、14 万ライセンスを越える販売実績を持つクラスターウェア「Serviceguard」の開発、OLTP 処理性能のへ注力など、初期段階から今日に至るまで、信頼性重視という指針はぶれることがない。

2000 年、インテル® Itanium® プロセッサー(以下、Itanium)搭載の Integrity サーバーが登場するが、これによってさらに「信頼」の技術に拍車がかかることになる。HP-UX を利用することにより、Itanium の性能を最大限に引き出してシステムとしての可用性をトータルで実現できる。業界標準のテクノロジーItanium を採用しながらも中長期の視点でポートフォリオが整備されているため、企業の IT プロジェクトでの必須条件を満たしているプラットフォームといえる。HP-UX の止まらない仕組み(DPR や DMR など)と、プロセッサーが提供するマシン・チェック・アーキテクチャ(MCA)、そしてサーバーで導入されているチップセット技術(ダブルチップスペアリングなど)は、いずれも別個に開発されたものではないからだ。それぞれ、システム障害に対する一連のアーキテクチャとして協調して設計されている。HPE がハードウェア/ソフトウェア双方のプラットフォームを有するベンダーだからこそ、別個に開発されたために発生する「オープンシステムの課題」を克服できるのだといえよう。

「安心」のサポート体制

ハードウェアと OS が別々に提供されることが多くなった今日でこそ注目を浴びているが、HPE は当初からオープンシステムのサポータビリティに力を注いできた。HP-UX は、商用だからこそできるサポータビリティを提供している。身近なクライアント PC で考えてみてほしい。例えば、OS のバージョンをアップした後、突然アプリケーションが動かなくなったことはないだろうか。パッチを当てた後で、プログラムがエラーを出すようになったことはないだろうか。不具合が出た時、パッチが出るまで待た

されたことはないだろうか。パッチを当てた後で再起動しなければならなくなることは、おそらく誰しも経験しているだろう。しかし、これが基幹システムで起こった場合、クレームだけで済む問題ではない。そこで HP-UX では、現在販売中のメジャーバージョンでリリースから 10 年のサポートを提供している。つまり、一度導入したシステムについては頻繁にアップデートの心配をしなくていい、ということになる。開発からサポートまでワンストップで提供される体制を取っており、ユーザーの視点で対応することを重視して、パッチもそれぞれユーザーの環境に合わせて適用することができる。

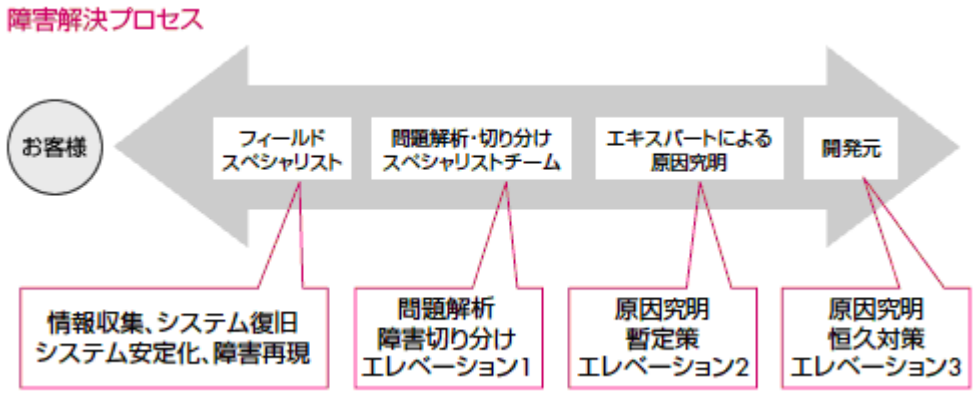


図 1：HP-UX の障害解決プロセス

「継続稼動」＋「スピーディな拡張」＝HP-UX 11i v3 のシステム

UNIX の最新バージョンと聞いて、あらためて自分の担当するサービスの OS が何なのかを確認する人も少なからずいることだろう。しかし、サービスを止めることなく運用するのは当然のこと、その上にビジネスのスピードに IT が追従していくには、インフラの構成要素として、完成度の高いオープンな OS と統合性が考えられたミドルウェアを選ぶことは不可欠。今回の HP-UX 11i v3 の登場は、次世代の基幹システムのオープンシステムのあり方を示すものだ。HPE では、ビジネスの変化に迅速・柔軟に適応し、必要な場所に必要なリソースをリアルタイムに配分できる次世代のインフラを「Converged Infrastructure」として提唱している。「Converged Infrastructure」を具象化した技術こそ、HP-UX 11i v3 ベースのシステムであると言っても過言ではない。「24 時間 365 日安心して使え」、「スピード経営に追従」をするインフラがオープンの技術で構築可能となっている。

HP-UX 11i v3 は全てが次世代スケール

まず、約 30%という大幅なパフォーマンス向上があるということに触れておかなくてならないだろう。これは、つまり HP-UX 11i v2 を使っていた環境をそのままで、OS と HP-UX 11i v3 に変えるだけでも、約 30%のパフォーマンス向上が見込めるということを意味している。性能を極限まで引き出すため、チューンアップを繰り返し、安定性、高速性を実現したのだ。また、HP-UX 11i v3 の場合、互換性を完全に維持したまま、パフォーマンスを引き出している点が特徴的だ。一例を挙げると、Integrity Superdome と HP-UX 11i v3 との組み合わせによって「TPC-C」ベンチマークや「SPECjAppServer2004」、「SAP SD 標準アプリケーション 2 階層」など、世界最速を記録している。つまり HP-UX 11i v3 と Integrity Superdome は、自他共に認める最速の組み合わせということになる。例えるならば、F1 や TGV（フランス国鉄の高速鉄道）といったところか。

対応ストレージ容量は天文学的な単位をサポート

ZB（ゼタバイト）という単位をご存知だろうか。 1 ZB は、数字で書くと「1,000, 000, 000, 000, 000, 000, 000」バイトと、0 が 21 個も続く。まさに天文学的数字である。HP-UX 11i v3 では、総ストレージ容量が 1 億 ZB 以上に対応する設計となっている。増え続けるストレージにも余裕を持って対応できる稀有な OS が、HP-UX 11i v3 なのである。

次世代インフラ技術の宝庫、HP-UX 11i v3

「24 時間 365 日安心して使え」、「スピード経営に追従」をするインフラの構築が可能である所以は、HP-UX 11i v3 に導入された新技術にある。それはまさに、「次世代インフラ技術の宝庫」である。その内容を、以下に箇条書きしてみよう。

表：HP-UX 11i v3 に導入された新技術

「柔軟性」を高める機能	「信頼性」を高める機能	「管理性」を高める機能
● インテル® Itanium® プロセッサーのハイパースレッティング対応 ● 次世代マス・ストレージ・スタック ● オンラインでのプロセッサーおよびメモリの交換 ● ネットワーク・ファイル・システム ● LVM のマルチパスとロードバランス機能	● Serviceguard の強化 ● IO カードのエラーリカバリー ● IO カードのオンライン削除 ● 動的カーネルパラメータの拡大 ● 並列化によるメモリダンプの強化 ● 暗号化ボリュームとファイルシステム ● セキュリティモジュールによる Trusted Computing Services のサポート ● Select Access Identity Management Integration (SA-IdMI) ● HP-UX の要塞化ツール、Bastille の機能強化	● 統合管理ツール、Systems Insight Manager 5.1 (SIM) ● System Management Homepage (SMH)の拡張 ● オンラインでパッチを当てる、Dynamic Root Disk (DRD) ● 新しいソフトウェア・パッチ統合管理ツール、Software Assistant (SWA) ● イベントの統合管理、Event Manager ● Modular Software Selection

では、ここからは「柔軟性」「信頼性」「管理性」と 3 つのキーポイントに分けて、それぞれの詳細を説明していこう。

次世代の「柔軟性」は未体験ゾーン

HP-UX 11i v3 で機能強化された項目の 1 つとして、「ハイパースレッディング・テクノロジへの対応」がある。このハイパースレッディングを活用することで、アプリケーションによっては最大 25%のパフォーマンスの向上が期待できる。しかし、すべてのアプリケーションでパフォーマンスの向上が期待できるわけではない。ハイパースレッディングはアプリケーションに依存する部分も大きいため、場合によっては使わない方がいいケースもあることは、その機能を持つプラットフォーム共通の注意点だ。

そこで、HP-UX 11i v3 では、同一 OS インスタンス上でハイパースレッディングを使い分けることができるように、ハイパースレッディングの可否をプロセッサごとに論理 CPU 経由で動的に設定できるようになっている。つまり、アプリケーションごとに最適なパフォーマンスを引き出すことが可能なのだ。より高いパフォーマンスを求めるユーザーには、朗報といえるだろう。
また、データの急速な拡大に対応するため、次世代マスストレージスタックの拡張を行った。これは、SCSI 論理ユニット（LUN）などの I/O デバイスの管理や、以前のバージョンと互換性を保ちつつサーバーの拡張性、適応性、性能を向上させるための機能を提供するものだ。
ネイティブ機能としてマルチパス機能とロードバランス機能が提供されており、I/O の負荷を自動的にバランスすることでアプリケーションのスループットも維持する。さらに、I/O パスやデバイスの障害に対するセルフヒーリング機能など管理性も向上している。

図 2：将来的なデータ量拡大へも備える

3 年先、5 年先に、どれくらいのストレージが必要になるか考えたことはあるだろうか。IT 部門は、劇的に増加するストレージ容量に頭を抱えている。このままデータが増加し続けた場合、「システムが対応できるかどうかもわからない」というのが正直なところではないだろうか。

現在、電子メールやビデオ、オーディオ、画像など、データ容量の大きい非構造データが飛躍的に増加している。さらに、新規ビジネス機会を探るために、企業に蓄積されているデータを活用したビジネスインテリジェンスやアーカイブニーズというものも顕著化している。こうした使い方をするのであれば、企業でやり取りされるすべてのデータを保存し、適切に管理して、必要に応じてすぐに引き出せる体制を整えなければならない。企業で必要とされているストレージは今後も爆発的に増加していくことが予想されるが、その適切な管理・運用方法はまだ見つかっていない。

前述のとおり、HP-UX 11i v3 は総ストレージ容量で 1 億 ZB（ゼタバイト）以上に対応する設計となっている。あくまでアーキテクチャ上の数値ではあるが、ほぼ無制限にストレージを接続できると言ってもいいだろう。

各ディスクについて最大 8ZB をサポートし、最大 1600 万台のストレージデバイスを接続することが可能。将来に備えたデータ量への拡大にも十分対応できる。これは、HP-UX 11i v3 が現在だけでなく将来にわたって安心して使えるだけのポテンシャルを持っていることを示している。

動的なリソース管理も、強化ポイントとして挙げられる。v2 と同様、HP-UX 11i v3 でも幅広いパーティショニング技術を採用しており、かつ、大幅な機能強化が実施されているのだ。

隔離性が高く、セルボード障害がシステム全体に波及することがない物理パーティションである「nPartitions（nPars）」、ハードウェアやソフトウェアのパフォーマンスを低下させない形でパーティション分割を実現している「Virtual Partitions（vPar）」、CPU リソースを複数の OS 環境に分割でき、エントリークラスのサーバーにも対応する「Integrity Virtual Machines（Integrity VM）」、プロセッサ・リソースの効果的な割り当てを支援する「HP-UX Processor Sets（pset）」などの仮想化機能が提供されている。これら分割度の異なる仮想化機能を上手く組み合わせることで、障害と隔離し、信頼性を確保しつつ、さらなるサーバーリソースの利用率の向上が期待できる。

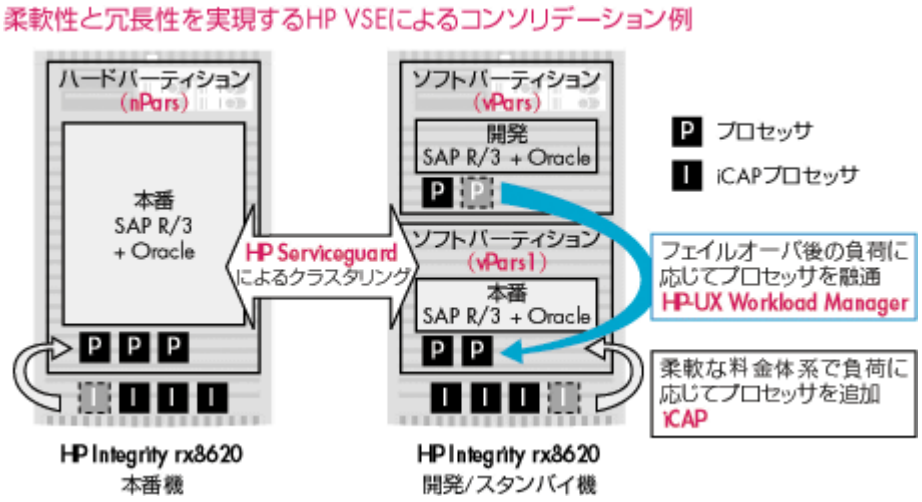


図 3：柔軟性と冗長性を実現する VSE によるコンソリデーション例

「nPars」については、オンラインでのセルボードの追加・削除の機能が追加されたことによって、柔軟性が向上している。
「vPars」については、プロセッサ・リソースの動的な移動が HP-UX 11i v1 の初期バージョンからサポートされていたが、さらに v3 では、vPars 間での動的なメモリ・リソースの移動をサポートしたことに加えて、vPars 環境内に HP-UX 11i v2 と v3 の混在も可能になった。

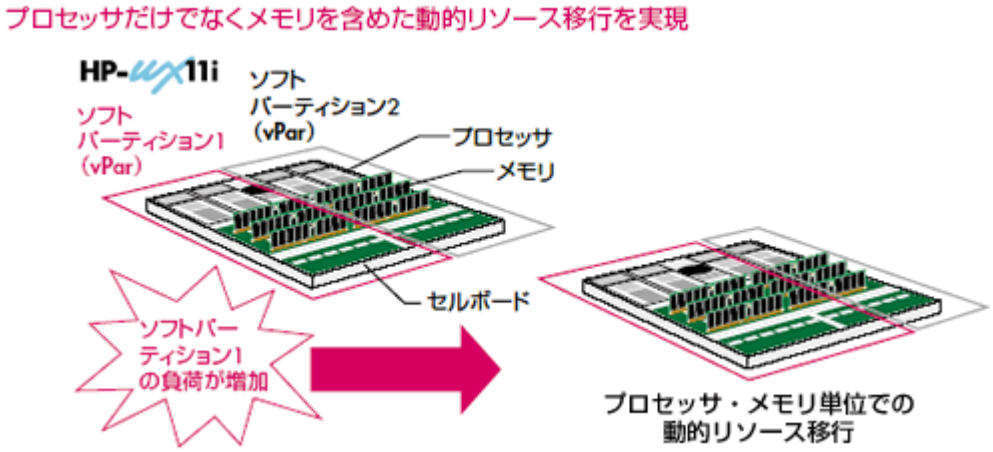


図 4：プロセッサだけでなくメモリを含めた動的リソース移行を実現

これらの機能によって、サーバーのプロセッサ・リソースやメモリ・リソースを最大限に活用することが可能になり、ビジネス処理のピークに柔軟に対応するアベイラビリティを確保できるのだ。

以上のように、HP-UX 11i v3 ではパフォーマンスの向上や次世代マス・ストレージ・スタック、パーティショニング機能の強化により、拡張性と柔軟性を強化することに成功した。次に、メンテナンス負荷の低減となる信頼性、管理性について紹介したい。

「信頼」とはこういうものだと、新たな時代を切り開く

次に、「確固たる信頼性」について見ていこう。最近は、「なにがあっても止められない」サービスが増えている。電子メールですら、いったん止まってしまうと、とたんに業務に支障をきたしてしまう。ミッションクリティカルな業務であればなおさらで、その影響は計り知れない。
企業システムに求められる信頼性・可用性・保守性の向上に対する HPE のアプローチは、「障害発生とシステムダウンの極小化」「障害発生時／システムダウン時の迅速な復旧」の 2 点に集約されている。

これは、サービスの停止を最小化してビジネスを継続させるために必要不可欠の要素であり、多くの IT 管理者のニーズと合致している。HP-UX 11i v3 で提供されているさまざまな自己監視・診断・修復機能は、そのためのものだ。
例えば、ハードウェア障害が起きた際、ビジネスの継続性を確保しながらメインテナンス作業を続けることは難しく、通常ならば少なからずダウンタイムが発生する。しかし、HPE の場合は必ずしもそうではない。メモリ、プロセッサ、I/O カードなどにホットスワップ対応のハードウェアモジュールを使えば、OS 稼働中でも追加・削除・交換ができるのである。真にビジネスを継続させながら、ハードウェア障害に対処できる環境を提供しているのだ。
PCI I/O エラーの自動回復も可能となった。PCI の I/O エラーに泣かされた経験は誰にでもあるだろう。これは計画外のシステムダウンにつながり、ビジネスの継続性を維持できない。HP-UX 11i v3 では、PCI I/O エラーによるシステムダウンを回避する機能に加え、エラーの起こったカードを自動的に初期化して再度利用できるように回復する機能が追加されている。さらに、PCI のオンライン削除（OLD）に対応したので交換が必要なカードもシステム稼働中に交換作業が可能になった。これによって、システムの信頼性を高めるばかりか、運用の柔軟性も向上している。このほかに、CPU やメモリリソースの追加や障害コンポーネントの交換を迅速に実施するため、オンラインでのセルボートの追加・交換機能が提供される予定だ。これが実現すれば、信頼性と柔軟性を大幅に向上し、新時代の「信頼性」を獲得できる。今から期待している機能強化の項目だ。

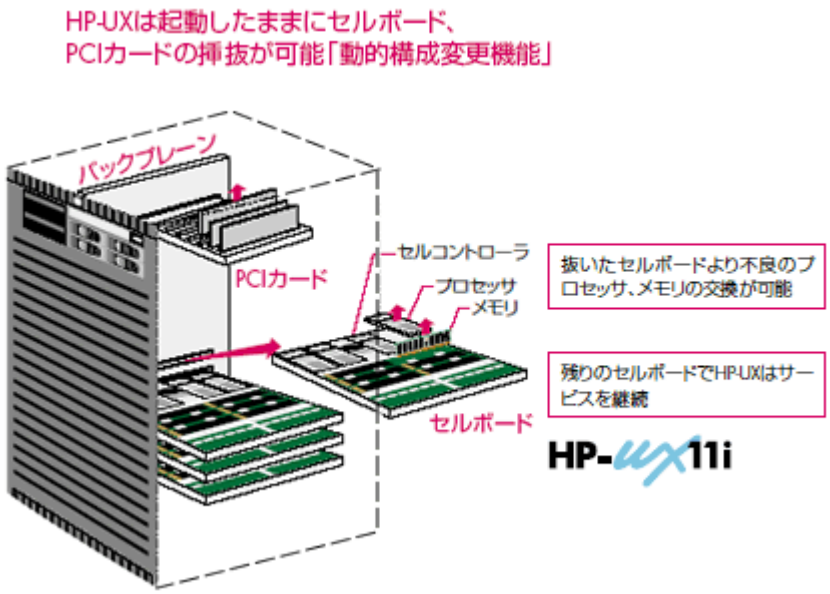


図 5：HP-UX は起動したままにセルボード、PCI カードの挿抜が可能「動的構成変更機能」

さらに、ダウンタイムを最小化するために、メモリダンプの並列処理機能を採用した。大量のメモリを積んだシステムになればなるほど、メモリダンプは意外と時間がかかるが、分散並列ダンプを行うようにマシン構成をすることで、ダンプスループットとダンプ時間を大幅に短縮できる。当たり前にできている部分にもメスをいれ、ダウンタイムを最小化しようとする試みだが、見事に成功している。
こういった HP-UX 11i v3 の新機能を見るにつけ、小さなことの積み重ねが大きな効果を生むのだと、本当に実感させられる。
HA クラスタとして定番となっている「Serviceguard」は、1995 年にリリースされた当初から、クラスタ設計においてデータの整合性を最も重要な要件として位置づけていた。整合性の失われたテラバイト単位のデータの復元には大変な時間がかかり、システムの復旧までに何日もかかってしまうからだ。Serviceguard では、複数ノードのデータベースへの書き込みが競合し、致命的な問題となる「スプリット・ブレイン」の発生を許さない設計ポリシーが取り入れられている。また、HP-UX 版の Serviceguard では「ゴースト I/O」と呼ばれる現象への対策として、障害時にディスクへの書き込み処理を直ちに停止する仕組みを実装、データの破損を防止する。ソフトウェアとハードウェアが直接連携しているからこそ実現できるメカニズムだ。さらに、Serviceguard をベースとしたディザスタリカバリ・ソリューションを提供している。

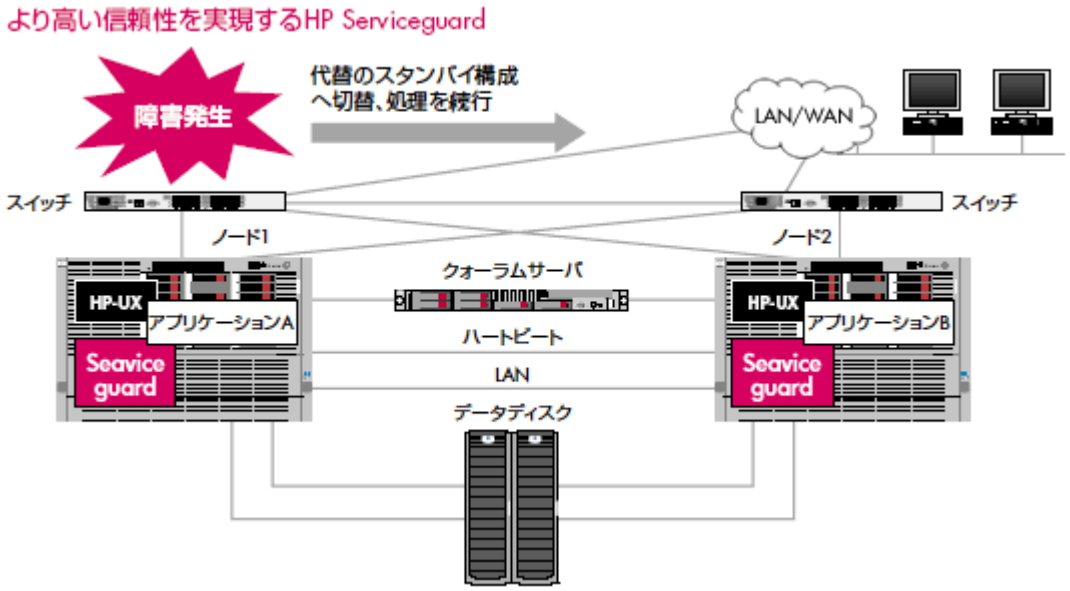


図 6：より高い信頼性を実現する Serviceguard

一方、社内外に存在するさまざまな脅威から最大限の防御を実現する統合セキュリティ技術の充実も図られている。例えば、HP-UX 11i v3 では、ファイルシステムの暗号化機能として「HP-UX Encrypted Volume & File System」が提供されている。ほかの UNIX システムでは、暗号化機能は ISV が提供しているケースもあるが、それでは OS との親和性やアプリケーションとの競合などといった検証作業が必要となる。導入によって新たなリスクが内包されるのだ。言い換えると、セキュリティ面での「安心」を選ぶか、システム面での「安心」を選ぶか、常に選択を迫られている状況である。システムとの相性が悪ければ、当然、暗号化機能を導入することもできない。「安心」を得るために何らかの「不安」を抱えるのでは、本当の意味での「信頼」を得ることはない。情報漏えいの危険性とも隣り合わせの中、神経をすり減らしギリギリの選択をし続けているのが IT 管理者である。

HP-UX 11i v3 のように、OS のコアの部分まで知り尽くしている OS ベンダーが暗号化ソリューションを提供するということは、安心して導入・活用できるだけでなく、情報漏えいのリスクも大幅に低減できることを意味している。IT 管理者は、HP-UX 11i v3 を選択することで、システムの安定性だけでなく、自分自身の安定性・信頼性も向上させられるだろう。IT 管理者をサポートする各種機能が、きちんと提供されているからだ。

「管理性」と書いて「ユーザビリティ」と読む

「シンプルな管理環境」は、管理コストを削減し、IT の生産性を向上させるため、企業にとってもメリットが大きい。管理コストの削減は、あわせて管理工数の削減も実現する。つまり IT 管理者は、煩雑な運用管理に時間を割く必要がなくなって、本来の業務に集中することができる。HP-UX 11i v3 は「攻め」の IT 投資といえよう。

その中核をなすのが、HP-UX サーバーを効率的に管理する Web ベースの管理ツール「HP-UX 11i System Management Homepage（SMH）」だろう。SIM と連携して複数のシステムを管理できるほか、ユーザーアカウント、ディスク、ファイルシステム、デバイス、ネットワークなどを GUI で容易に管理できるツールだ。直感的に作業できるのはもちろん、システムコンポーネントの稼働監視なども提供されている。既存のターミナルベースの管理も可能だが、HP-UX 11i v3 では基本的に Web ベースのツールを活用することとなる。

HP-UX 11i v3 では、Web ベースの SGM（Serviceguard Manager）や PRM（HP-UX Process Resource Manager）機能などと連携し、管理機能が強化されている。

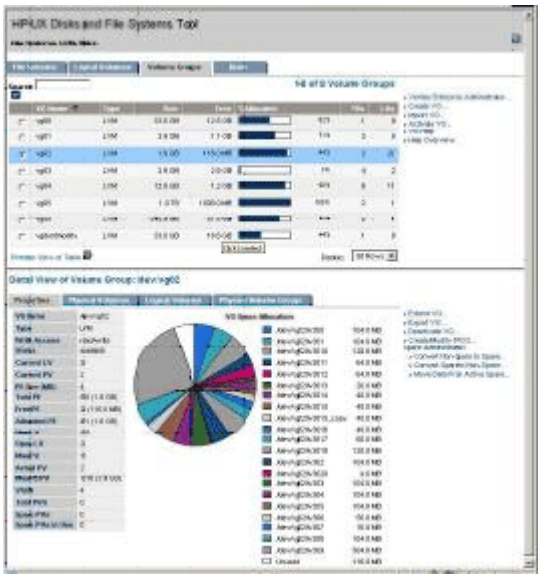


図 7：サーバー統合管理を実現する HP-UX 11i SMH

また、パッチ管理に関しても新しい技術が用いられている。通常パッチを当てるためには、稼働しているシステムのリブートを必要とするため、少なからずシステムの計画停止が必要だった。頻度は少ないものの、パッチ管理を行うためにサービスの時間帯をはずさなければならず、休日の作業となることもあった。HP-UX 11i v3 では、「ダイナミック・ルートディスク（DRD）」機能を提供している。DRD とは、稼働中のシステムのイメージを複製し、この複製した非アクティブなシステムイメージに対してパッチを当て、適当なタイミングでリブートを実施する方法だ。従来の方法と比べ、計画停止時間の大幅な削減を実現している。また、パッチをインストールしていないシステム環境が残るため、パッチ適用後のシステムで問題が起こった場合にも、パッチ適用前のシステムにすぐに戻すことができる。これだけでも復旧時間や運用に要する人的工数が違う。何か失敗したときに、「5 分前に戻りたい」と思ったことはなかっただろうか。それを実現できるのが HP-UX 11i v3 である。

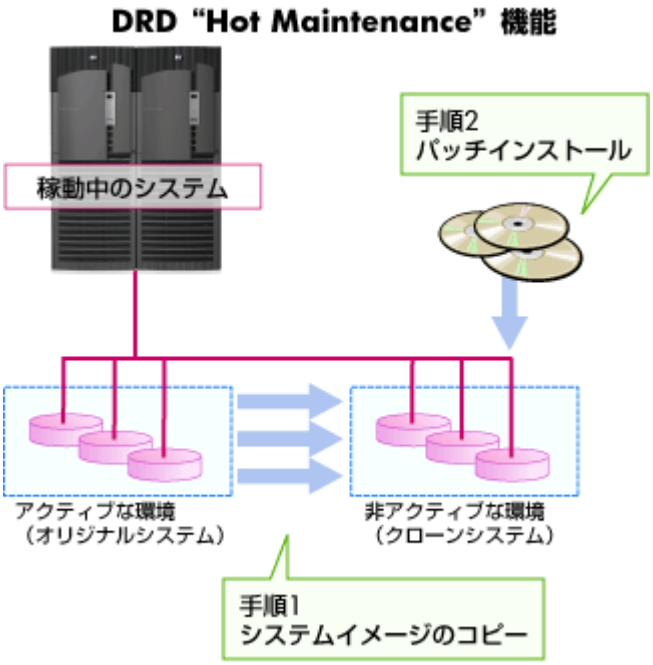


図 8：ダイナミック・ルートディスクを利用した HP-UX ソフトウェアへのオンラインパッチ適用

さらに、カーネルパラメータを動的または自動的にチューニングする「Dynamic Tunables」「Automatic Tunables」といった機能が強化されている。HP-UX 11i v3 では、カーネルのチューニングによる再起動を 6 割近く低減できるという結果も出ている。さらにストレージデバイスや I/O パスの自動的な検出と構成の機能強化により、ストレージの管理も容易となる。
HP-UX 11i v3 では、自動化とシンプルな管理環境を推し進めると同時に、システム管理ツールの強化を行っている。特にストレージ管理の大幅な改善により、SAN 環境下における管理支援業務が軽減される。
このように、HP-UX 11i v3 では、データ容量の拡大へ対応し、可用性・自動化が大幅に強化された。さらに今後登場する将来の HP-UX では、ゼロ・ダウンタイムの仮想化環境が主なテーマとなる。方向性が明確な HP-UX は、利用する企業にとっても選択する価値のあるシステムといえよう。

今回は、新機能や強化ポイントに焦点を当てて概要をご紹介した。次回からは、テーマを絞ってより詳細な HP-UX 11i v3 情報をお届けする予定だ。「こんなことを訊いてみたい」「ここはどうなったのか」など、ご要望をぜひ編集部あてにお寄せいただきたい。

仮想パーティションとユーティリティの強化ポイント・前編

2007 年 5 月　テクニカルライター 秋葉けんた

特別企画 第 2 回目となる今回は、HP-UX 11i v3 のリリースに合わせてバージョンアップされた HP-UX Virtual Partitions(vPars)に注目してみよう。「オンラインでのメモリの追加」「ハイパースレッディング(HT)のサポート」といった機能強化に加え、HP-UX 11i v2 と HP-UX 11i v3 の混在環境までサポートした最新版 vPars を紹介する。

変化自在のソフトパーティション、vPars

HPE は、1997 年オープンシステム上で仮想化を実現して以来、10 年間にわたってその市場をリードしてきた「仮想化」の先駆者らしく、仮想化製品ラインアップは群を抜いて充実している(詳しくは、過去の記事「商用 UNIX のパーティション技術最新事情」や「サーバー仮想化の"先駆者"HPE が世に問う 新技術『Integrity VM』」で確認してほしい)。そのラインアップがあるが故、様々な用途が可能だ。例えば、ソフトパーティションの HP-UX Virtual Partitions(vPars)であれば「一時的に開発用のサーバーが必要」といった急を要するユーザの要望にも瞬時に応えることもできる(過去の記事[サーバーを増やす『打ち出の小づち』」)。

vPars は、1 つのハードパーティション(nPars)を複数の仮想パーティションに分割する技術。VMware などの「仮想マシン方式による仮想化」と異なり、パフォーマンスのオーバーヘッドがほとんど発生しないという特長がある。また、仮想パーティションに専用の CPU、メモリ、I/O が割り振られるため、仮想パーティション間でのリソースの競合も発生しない。さらに、あるパーティションでカーネルパニックやパッチ適用など、そのパーティションでリブートが必要な作業が発生した場合も、ほかの仮想パーティションへの影響はない。

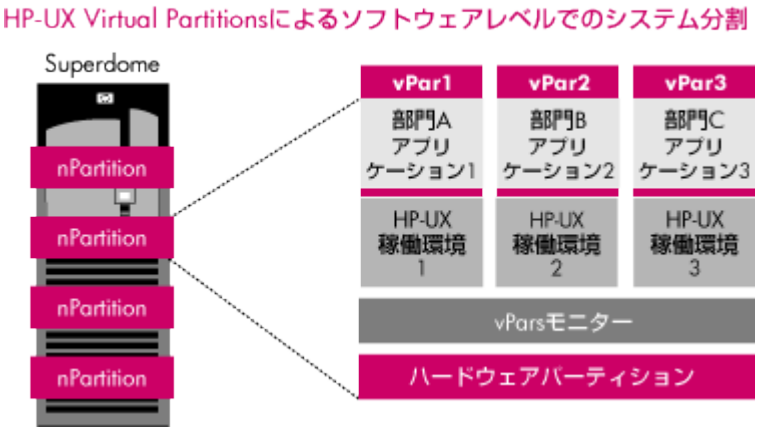


図 1：vPars の概念図

強化された最新 vPars「A.05.01」とは

HP-UX 11i v3 のリリースに合わせ、vPars も「A.04.xx」から「A.05.01」にバージョンアップされた。HP-UX 11i v3 への対応は当然として、ほかにもいくつかの機能強化が図られている。その詳細を見ていこう。

まず、メモリをオンラインで割り当てることができるようになったという点に注目したい。vPars では元から、リブートせずに、仮想パーティション間で CPU を動的に割り当てられたが、最新の vPars からは CPU だけではなくメモリもオンラインでの割り当てが可能になったのである。元々vPars は、リブートすることでメモリの割り当てを変更できた。しかし最新の vPars は、リブートの必要がない。これはつまり、柔軟性を向上させつつ、システムのメインテナンスのためのリブート削減・計画停止の最小化にも寄与する機能強化である。

今回の機能強化により、CPU とメモリを負荷状況に応じて動的に割り当てることができるようになった。たとえば、vPar1 に一般業務、vPars2 にバッチ処理のパーティションを作成しておけば、昼は一般業務、夜はバッチ処理をメインに行いたい場合、昼は vPar1 に夜は vPar2 に CPU とメモリを多く割り当てればよいのだ。このように、vPars を使いこなせば、柔軟でリソースを有効利用できるサーバー環境が実現でき、企業システムが大幅に改善されるだろう。

また、オンラインでのメモリ割り当て機能の追加にともなって、メモリには、「Base(ベース)」メモリ、「Floating(フローティング)」メモリという種別が追加されている。「Base」メモリは固定のメモリで、オンラインで追加できるが、削除するには従来の vPars 同様に、該当の vPars を停止・リブートさせなければならない。一方「Floating」メモリは、オンラインで追加・削除可能なメモリだ。最新 vPars では、「Floating」メモリを活用するケースが増えることになるだろう。追加するメモリの単位は、あらかじめ設定している「Granularity」の倍数となる。「Granularity」の値はデフォルトでは 128MB となっているので、特にこの設定を変えない限り、動的に追加・削除するメモリは 128MB の倍数に切り上げられ、丸められた値となる。もし、「Floating」メモリの追加・削除中にその操作をキャンセルしたい場合は、ステータスが“Pending”中であれば、操作のキャンセルを行える。

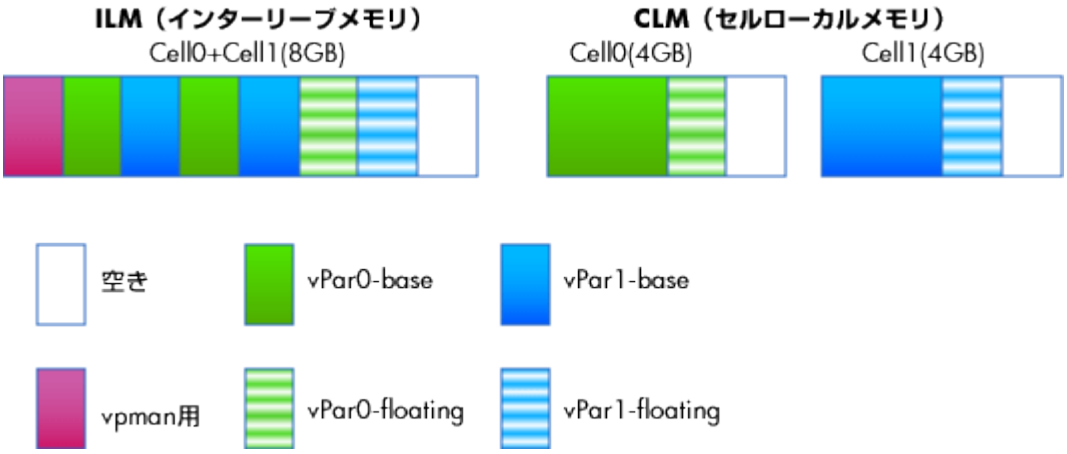


図 2：vPars のメモリ割り当ての種類

次に、HP-UX11i v3 でサポートされたハイパースレッディング(HT)の機能については、vPars 環境においても利用が可能だ。この HT を利用することで、アプリケーションによってはパフォーマンスの向上を期待することが出来るだろう。

さらに、最新の vPars では、同一 nPars 上に HP-UX 11i v2 と HP-UX 11i v3 の混在環境を構築することもできるようになった。既存の HP-UX 11i v2 環境を活用しつつ、最新の HP-UX 11i v3 を並列で稼働できるのである。ただし、混在環境で利用するにはいくつかの注意が必要となる。まず、HP-UX11i v2 を利用する場合、vPars A.04.02 以降のバージョンのみ対応となり、これ以前のバージョンの vPars では利用できない。また混在環境で利用する場合、かならず vPars A.05.01 の vPars モニタを起動し、その上で HP-UX11i v2 や HP-UX11i v3 を起動する必要があるのだ。また、HP-UX 11i v3 でサポートされた HT が HP-UX11i v2 ではサポートされていないため、HP-UX 11i v3 との混在環境下においての利用はできない。HP-UX 11i v2 と HP-UX 11i v3 との混在環境で利用する場合、HT は忘れずに OFF の設定にする必要がある。

表：vPars のバージョンによる機能の違い

機能	vPars A.05.xx	vPars A.04.xx	vPars A.03.xx
対応する OS のバージョン	HP-UX11i v3	HP-UX11i v2	HP-UX11i v1
HP-UX11iv2 と v3 の混在環境のサポート	○	×	×
HP-UX の混在環境で必要とされる vPars のバージョン	A.05.01	A.04.02	×
CPU の動的な移行	○	○	○
メモリの動的な移行	○	×	×
HT のサポート	○	×	×
セル・ローカルメモリ　(CLM)	○	○	×
CPU の I/O 割り込み処理	○	○	Bound CPU のみ

かゆいところに手が届く、vPars のこだわり

最新の vPars である「A.05.01」では、今までの vPars 同様、ILM(インターリーブメモリ)と CLM(セルローカルメモリ)の両方をサポートしている。これは、アプリケーションによってパフォーマンスを最大に引き出すための HPE の提案である。

Integrity サーバーのハイエンドおよびミッドレンジサーバーは、CPU とメモリを搭載するセルボードをサーバー全体で複数枚搭載し、それぞれを相互接続している。各 CPU は、同一セルボード上のローカルメモリはもちろん、他のセルボードのメモリにも自由にアクセスでき、広大なメモリ空間を利用できるようになっている。この手法は、非常に高い拡張性を有する反面、別のセル上にあるメモリへのアクセスが低速となり、メモリアクセス速度のばらつきが出てしまうという難点がある。メモリアクセスを優先するのであれば、別のセルボード上のメモリを使わないほうがいい。しかし、別セルボード上のメモリを使えなければ、広大なメモリ空間を利用することができない。広大なメモリ空間が必要なのか、より高速なメモリアクセスが必要なのかは、利用するアプリケーションに依存する部分が大きい。「それぞれを使い分けたい」という想いを実現するのが、ILM および CLM というメモリアクセス方法である。

ILM は、1 つのアドレス空間を接続されている複数のセルボードのそれぞれのメモリにインターリーブするため、CPU はすべてのセルボードのメモリに均等にアクセスする。広大なメモリ空間を、あるパフォーマンスを維持しながら利用できることになる。一方 CLM は、CPU と同一セルボード上のメモリを連続したアドレス空間に確保し、同一セルボード内のローカルメモリへのアクセスとなる。セルボード間の通信がないため、より高速なメモリアクセスを実現する手法となる。もちろん vPars では、この 2 つのメモリアクセス方法をサポートしている。つまりアプリケーションの特性にあわせて CLM、ILM を自由に選択し、利用できるのだ。

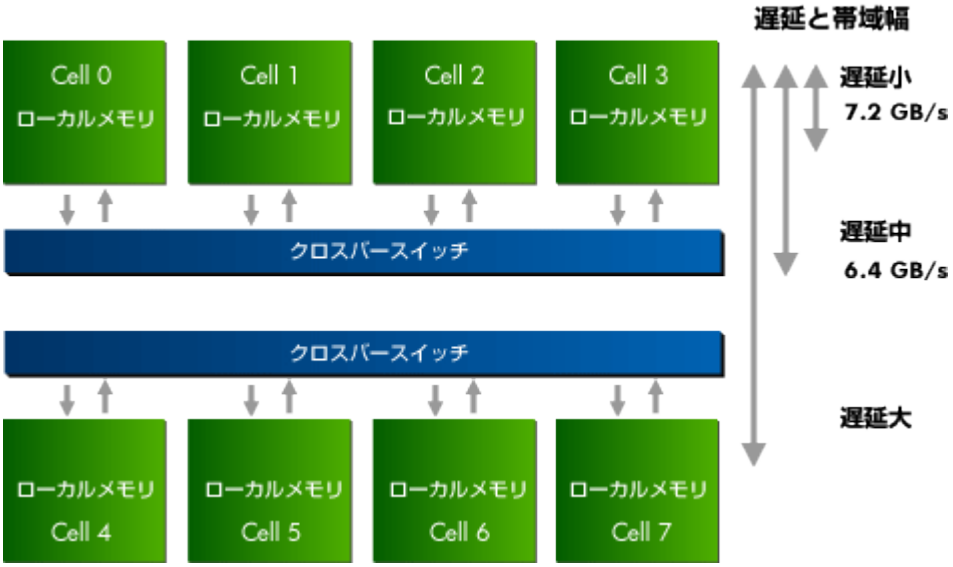


図 3：メモリアクセス速度について

TiCAP (テンポラリ・インスタント・キャパシティ)で簡単増強

普段は既存のキャパシティで間に合っているのだが、ある一時期だけキャパシティを拡張したいというケースは意外に多い。しかも、その多くは急を要する案件だ。ところが、キャパシティを拡張するための追加投資を行うには煩雑な社内手続きが多く、時間も手間もかかってしまう。急を要するのに「急」に対応できないのが現実だ。HP では、「急」を要するキャパシティの拡張を実現するため、未使用の CPU に対し 720 時間を単位とする使用権を販売し、ユーザは必要に応じて使用権の時間内で iCAP CPU を利用できる「TiCAP」を提供している。なお、iCAP CPU はライセンスされていない CPU だが、「vparstatus -A」の出力には利用可能な CPU として表示される。あらかじめ「TiCAP」を購入しておけば、社内の煩雑な手続きは不要。突発的なキャパシティ増加に対しても、適切に対応することができるようになる。もちろん、使わなければ使用権が失われることもない。最新バージョンの vPars では、その TiCAP をサポートし、顧客の投資を最小限にしながら突発的なキャパシティの拡張にも対応できるようになった。「保険」的な意味合いでも、非常に心強いソリューションであるといえよう。

さらに最新の vPars では、HP-UX 11i v3 で利用できるようになった PCI I/O エラーによるシステムダウンの回避と、エラーの起きているカードを自動回復する機能に加え、PCI のオンライン削除も vPars から利用できる。

仮想パーティションとユーティリティの強化ポイント・後編

2007 年 5 月　テクニカルライター 秋葉けんた

後半では、HP-UX11i v3 対応の vPars A.05.01 で強化された機能、メモリのダイナミックな移動について試してみよう。併せて既存の vPars でもサポートされている機能だが、CPU のダイナミックな移動も試してみよう。

vPars A.05.01 で追加されたメモリの動的な移動を試す

ここでは、以下のマシン・OS と vPars の構成を前提として説明を行う。

H/W	Integrity rx7620（CPU　8 コア、メモリ 16GB）
S/W	HP-UX11i v3、vPars A.05.01
vPars	vPar0（CPU 4 コア、メモリ 4GB）
	vPar1（CPU 2 コア、メモリ 2GB）
	vPar2（CPU 2 コア、メモリ 2GB）

まずは、メモリ移動を行う前に仮想パーティションのステータスを確認しておこう。

```
# vparstatus -P
Current Virtual Partition Version:  A.05.01
Monitor Version:  A.05.01

[Virtual Partition OS Version]

Virtual Partition Name          OS Version  State
==============================  ==========  =====
vpar0                           B.11.31     Up
vpar1                           B.11.31     Up
vpar2                           B.11.31     Up
```

HP-UX のバージョンが、B.11.31（11i v3）で、vPars のバージョンが A.05.01 であることが確認できる。

表 1：利用できる主なコマンド

vparcreate	仮想パーティションを作成
vparremove	仮想パーティションを削除
vparmodify	仮想パーティションの構成を変更
vparboot	仮想パーティションを起動
vparreset	仮想パーティションをハードリセット
vparstatus	仮想パーティションの構成を表示

次のコマンドで、各仮想パーティションの構成を確認する。

```
# vparstatus
[Virtual Partition]
                                                                  Boot
Virtual Partition Name         State Attributes   Kernel Path      Opts
============================== ===== ============ ================ =====
vpar0                          Up    Dyn,Auto,Nsr /stand/vmunix
vpar1                          Up    Dyn,Auto,Nsr /stand/vmunix
vpar2                          Up    Dyn,Auto,Nsr /stand/vmunix

[Virtual Partition Resource Summary]
                               CPU       Num   Num     Memory Granularity
Virtual Partition Name         Min/Max   CPUs  IO        ILM         CLM
============================== ========= ====  ====  ==========  ==========
vpar0                            1/  8     4     2         128         128
vpar1                            1/  8     2     2         128         128
vpar2                            1/  8     2     2         128         128

                                                Memory (MB)
                                        ILM                     CLM
                               # User                   # User
Virtual Partition Name         Ranges/MB     Total MB   Ranges/MB     Total MB
============================== ========== ==========  ========== ==========
vpar0                            0/   0        4096     0/   0            0
vpar1                            0/   0        2048     0/   0            0
vpar2                            0/   0        2048     0/   0            0
```

それでは実際にメモリの動的な移動作業を行ってみよう。まず、使用可能なメモリ量を調べることから始めよう。これには「vparstatus -A」を使う。また、「vparstatus -v」の出力では、割り当て済みの ILM および CLM の合計とフローティングメモリの量が表示されるので、こちらのコマンドを使ってメモリの割り当て状況を確認することもできる。

```
# vparstatus -A
[Available ILM (Base   /Range)]:  0x3000000/80
               (bytes) (MB)       0x20000000/1536
                                  0x178000000/2048
                                  0x2e8000000/4352
                                  0x3f8000000/112

[Available ILM (MB)]:  8128
```

```
[Available CLM (CellID Base  /Range)]:
                     (bytes) (MB)
[Available CLM (CellID MB)]:  0 0
```

メモリの空き状況は、約 8GB だ。

```
# vparstatus -p vpar0 -v
....
[Memory Details]
ILM, user-assigned [Base  /Range]:
                   (bytes) (MB)
ILM, monitor-assigned [Base  /Range]:  0x8000000/128
                      (bytes) (MB)     0x80000000/3968
ILM Total (MB):  4096  (Floating 0)

ILM Granularity (MB):  128

CLM, user-assigned [CellID Base  /Range]:
                         (bytes) (MB)
CLM, monitor-assigned [CellID Base  /Range]:
                           (bytes) (MB)
CLM (CellID MB):

CLM Granularity (MB):  128
```

vpar0 のメモリは、ILM で 4096MB（4GB）構成されていることが確認できる。

メモリをオンラインで割り当てるには、以下の 3 つの構文を利用する。ILM、CLM の追加・削除のほか、メモリアドレスを指定してメモリの追加・削除が可能だ。なお、割り当てられるメモリにはベースメモリ、フローティングメモリが利用できる。

表 2：メモリを追加・削除する構文

ILM	a|d mem::size[:b[ase]|f[loat]]
CLM	a|d cell:cell_ID:mem::size[:b[ase]|f[loat]]
アドレス	a|d mem:::base:range[:b[ase]|f[loat]]

※a は追加、d は削除、size はメガバイト単位のメモリサイズ、cell_ID はセル番号の指定となる。また「:float」を設定しなければ、デフォルトの「:base」が利用される。

ここでメモリを実際に追加してみよう。仮想パーティション「vpar0」に ILM のメモリを追加する場合には、「-a」オプションを使い、メモリを削除する場合は「-d」オプションを利用する。

では次のコマンドで、vpar0 に 2048MB のメモリを追加してみよう。

```
#  vparmodify -p vpar0 -a mem::2048:float
```

ここで、メモリが追加されていることを確認してみよう。

```
# vparstatus
....
```

```
                                          Memory (MB)
                                ILM                         CLM
                          # User                      # User
Virtual Partition Name    Ranges/MB     Total MB    Ranges/MB     Total MB
========================  ==========  ============  ==========  ============
vpar0                       0/   0          6144     0/   0             0
vpar1                       0/   0          2048     0/   0             0
vpar2                       0/   0          2048     0/   0             0
```

次に vPar0 からメモリを 2048MB 削除して、vPar1 に 2048MB 追加してみよう。

```
# vparmodify -p vpar0 -d mem::2048:float
# vparmodify -p vpar1 -a mem::2048:float
```

メモリを確認すると、

```
# vparstatus
....
                                          Memory (MB)
                                ILM                         CLM
                          # User                      # User
Virtual Partition Name    Ranges/MB     Total MB    Ranges/MB     Total MB
========================  ==========  ============  ==========  ============
vpar0                       0/   0          4096     0/   0             0
vpar1                       0/   0          4096     0/   0             0
vpar2                       0/   0          2048     0/   0             0
```

vPar0 は 2048MB 減って、vpar1 は 2048MB 増えて ILM メモリの Total MB がそれぞれ 4096MB（4GB）に変更されたことがわかる。
同様に、CLM(この場合セル 1)のフローティングメモリを追加する場合は以下の構文となる。

```
#  vparmodify -p vpar2 -a cell:1:mem::1024:float
```

なお、以上 2 つの構文を組み合わせ、ILM と CLM を 1 つのパーティションに割り当てることもできる。これらの組み合わせで、自由にオンラインでの追加・削除を実現できる。

```
#  vparmodify -p vpar2 -a cell:1:mem::1024 -a mem::1024
```

vPars A.05.01 で、CPU の動的な移動を試す

CPU の動的な移動は、今までの vPars でもサポートされていた便利な機能であるが、メモリの移動と同じようにコマンド 1 つで行えるので試してみよう。

```
# vparstatus
. . .
[Virtual Partition Resource Summary]
                          CPU       Num   Num      Memory Granularity
Virtual Partition Name    Min/Max   CPUs  IO         ILM          CLM
========================  =======   ====  ====  ==========  ==========
vpar0                       1/  8      4     2         128         128
vpar1                       1/  8      2     2         128         128
vpar2                       1/  8      2     2         128         128
```

現状は以下のとおり。

vpar0 --> CPU x 4 個
vpar1 --> CPU x 2 個
vpar2 --> CPU x 2 個

Total --> CPU x 8 個

では、次のコマンドで vpar0 から CPU を 2 つ削除してみよう。

```
# vparmodify -p vpar0 -d cpu::2
# vparstatus
[Virtual Partition Resource Summary]
                              CPU      Num   Num     Memory Granularity
Virtual Partition Name        Min/Max  CPUs  IO       ILM         CLM
============================  =======  ====  ====  ==========  ==========
vpar0                           1/  8     2     2         128         128
vpar1                           1/  8     2     2         128         128
vpar2                           1/  8     2     2         128         128
```

vpar0 の CPU 数が 2 つ減った。全体でみると 2 個余っている状態。

vpar0 --> CPU x 2 個　　ー>　4 個から 2 個に
vpar1 --> CPU x 2 個
vpar2 --> CPU x 2 個

Total --> CPU x 6 個　　ー>　2 個余り

では今度は、vpar1 に CPU を 2 個追加してみよう。

```
# vparmodify -p vpar1 -a cpu::2
# vparstatus
[Virtual Partition Resource Summary]
                              CPU      Num   Num     Memory Granularity
Virtual Partition Name        Min/Max  CPUs  IO       ILM         CLM
============================  =======  ====  ====  ==========  ==========
vpar0                           1/  8     2     2         128         128
vpar1                           1/  8     4     2         128         128
vpar2                           1/  8     2     2         128         128
```

vpar1 の CPU が 2 個増えた。

vpar0 --> CPU x 2 個
vpar1 --> CPU x 4 個　　ー>　2 個から 4 個に
vpar2 --> CPU x 2 個

Total --> CPU x 8 個

結果的には vpar0 の CPU 2 個が、vpar1 に移動したことになる。当然これらの変更には一切リブートを必要としないし、移動にも時間がかからないですぐに変更が反映される。

このように仮想パーティションを用いたシステム構成ならば、アプリケーションの負荷が急に上がったり、予想外にユーザのアクセスが増えて CPU やメモリリソースが足りなくなった場合でも、負荷が高くないマシンから一時的に CPU やメモリを移動してサービスレベルを落とさずに処理を行うことも可能である。

注意：　vPars A.04.01 から Bound/Unbound という概念がなくなり、すべてのプロセッサを動的に移動できるようになりました（ただし、ブート・プロセッサを除く）。

次世代大容量ストレージに対応した新 I/O スタック

2007 年 6 月　テクニカルライター 米田 聡

特別企画 第 3 回目となる今回は、HP-UX 11i v3 の最大の改良点と言っても過言ではない新しいストレージスタックを取り上げる。HP-UX 11i v3 では、1 億 ZB(ゼタバイト)という驚異的なストレージ容量に対応可能な設計にするなど、まさにストレージ新時代に対応した I/O に関する新技術が満載されている。前編では、増加し続けるストレージ容量への対応を可能にするスケーラビリティの大幅な拡大と、劇的に向上したパフォーマンスに注目してみた。

スケーラビリティが拡大した新ストレージスタック

オープンシステムが基幹業務や WEB サーバーなど高い信頼性や大量のデータ処理能力が必要な用途に利用されるようになった 1990 年代以降、ストレージの大規模化が急ピッチで進んでいる。大規模なデータに対応するために大容量のストレージが必要になっているのはもちろんだが、信頼性を確保するために、ストレージの冗長化がごく一般的に行われるようになっているのは、ご存じのとおりだ。

こうした傾向は当然ながら今後も続くだろう。言い換えれば、次世代システムには、増加し続け、規模の拡大を続けるストレージシステムに対応していく能力が求められているわけである。この要求に応えるべく、HP-UX 11i v3 ではストレージスタックに大規模な改良が加えられており、総ストレージ容量 1 億 ZB（ゼタバイト）以上、1 基のディスクあたり最大 8ZB の容量まで対応可能な設計が行われている。なお、1ZB とは 1,000,000,000TB にあたる。さらに、大規模なストレージ群に対応するため、新 I/O スタックでは多くの制限が取り払われている。たとえば、対応する I/O バス数は従来（HP-UX 11i v2）の 256 までという制限がなくなり、事実上、無限のバス数を扱うことができるようになった。

また、システム当たりのアクティブ LUN 数は最大 16,384 に拡大（従来は 8192）、理論的には 16M（メガ）の LUN 数に対応できるという。簡単に言ってしまえば、HP-UX 11 v3 が対応できるストレージ容量はほぼ無限になったといってもいい。増加し続けるデータ量に備え、5 年先、あるいは 10 年先に必要とされるストレージ容量にも余裕を持って対応できる設計になっている。

表：拡大したスケーラビリティ

I/O バスの最大数	理論的最大値が 4G に（11i v2 では 256）
サポートされる LUN の最大数	16384（11i v2 では 8192 アクティブ） 理論的最大は 16M

最大 LUN サイズ	ドライバに依存して 2TB 以上も可（11i v2 では 2TB）
1 つの LUN への最大パス数	32（11i v2 では 8）
ファイルシステムの最大サイズ	理論的には 8192TB 実際は使用するボリュームマネージャやファイルシステムに依存

実際の運用の例は後編で詳しく紹介するが、I/O スタックに加えられた拡張をおおまかに把握しておくことにしよう。

新しいデバイス・パスとデバイス・ファイルの表記形式の導入

HP-UX のユーザなら、従来の HP-UX 11i v2 のデバイス・パスやデバイス・ファイルの表記形式では前出の拡張に対応しきれないことに気づくだろう。たとえば、デバイス・パスは HBA へのパスとアドレスが使用されている。したがって、I/O バス数や LUN の最大数はアドレスなどにより制限されることになる。また、デバイス・ファイルの表記形式も従来のデバイス・パスを元にしたものでは前出の拡張に対応しきれない。仮に、現在の形式のまま対応できたとしても、表記が煩雑になりすぎ管理の手間が大幅に増えるだろう。

そこで、HP-UX 11i v3 ではデバイス・パスやデバイス・ファイルに新たな表記形式が導入された。それが「Agile View」や「Persistent DSF」である。具体例は後編で紹介するが、シンプルかつアドレス空間などの制約を受けない形式が導入された結果、大規模なストレージシステムに柔軟に対応することが可能になり、また管理性も大幅に向上している。

ネイティブ・マルチパス

複数のデバイス・パスを使用して負荷のバランスを最適に保ち、また故障発生時には不具合が生じたパスを切り離し、システムを落とすことなく継続稼動を可能にするマルチパスは、大規模なストレージをデータベースなどで運用するシステムでは必須のソリューションだ。

ご存じのように、HP-UX には LVM PVlink、VERITAS DMP、SecurePath、PowerPath など、サードパーティのソリューションを含め複数のマルチパス・ソリューションが用意されている。だがこれらのマルチパス・ソリューションは製品によって動作する環境が異なる。たとえば、LVM PV Link は論理ボリュームマネージャでマルチパスを提供する。一方、VERITAS DMP は独自の論理ボリュームマネージャ VxVM 上で実現されている。また、VERITAS DMP や SecurePath はオプション製品なので、追加の投資を考慮しなければならない。もちろん、その分機能が豊富で使いこなせば十分メリットがあるが、ユーザにとっては、気軽に導入したり、試しに使ってみることが簡単ではなかった。

そこで、HP-UX 11i v3 では OS そのものにマルチパスの機構・・・ネイティブ・マルチパスが組み込まれた。ネイティブ・マルチパスはデフォルトで有効であり、後述するように容易に運用が可能という特徴を持っている。

大きな改善を加えながらも完全な後方互換性を確保

以上の 2 点が HP-UX 11i v3 の新 I/O スタックの大きな特徴となっているが、読者の中には少々、心配になった方もおられるだろう。デバイス・パスやデバイス・ファイルの表記の変更、あるいは従来にないマルチパスのサポートといった大きな変更は、管理や運用に大きな違いをもたらす可能性があるからだ。また、設定等に大きな変更が必要になる恐れを抱く管理者は多いだろう。

だが、HP-UX 11i v3 の I/O スタックには、これほどの大幅な改訂が加えられながらも、ほぼ完全な後方互換性が確保されているという、もう 1 つの大きな特徴があるのだ。デバイス・パスやデバイス・ファイルの表記は従来の方式も変更せずにそのまま

利用でき、またネイティブ・マルチパスは上位レイヤからは完全に透過になっているため、ほとんどの場合は設定に特別な変更を加えなくても利用できるのである。

新ストレージスタックの卓越したパフォーマンス

新ストレージスタックの特徴は、スケーラビリティが向上していると同時に、大幅なパフォーマンスの向上が図られている点にある。百聞は一見にしかず、まずはグラフを見てほしい。

次のグラフは、I/O システムの走査をする ioscan の実行時間を計測したものだ。ご存じだと思うが、システムに接続されている I/O デバイスが増えるほど実行時間が長くなる。大規模なシステムでは、ioscan の実行時間が数分ほどかかることもめずらしくはないが、HP-UX 11i v3 ではほとんどの場合、数十秒以内で実行が終了する。

環境	
Server :	Integrity サーバー rx8620（8 プロセッサー）
	FC アダプタ 8 枚
Storage :	Storage Works XP シリーズ
	600 LUN（合計 19200 LUN パス）

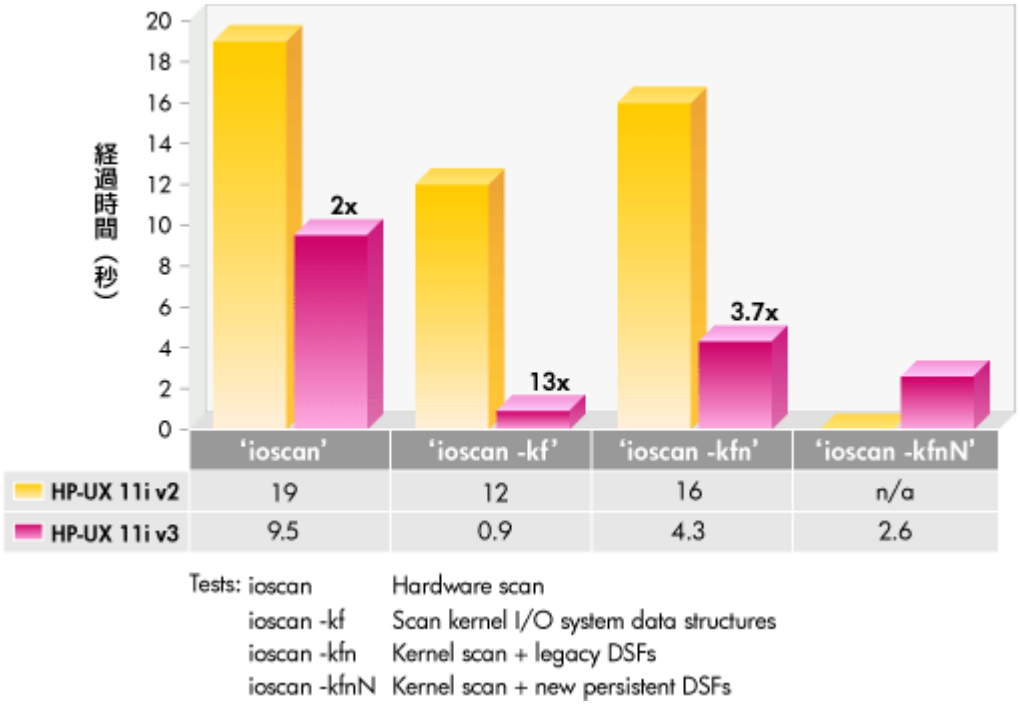


図 1：大幅に短縮された HP-UX 11i v3 での ioscan の実行時間

I/O デバイスの走査時間の短縮は、システムのブート時間の短縮にも直結している。次のグラフは起動時の初期スキャンにかかる時間およびシャットダウン＆リブートの時間を比較したものだが、大幅に時間が短縮されていることがわかるだろう。

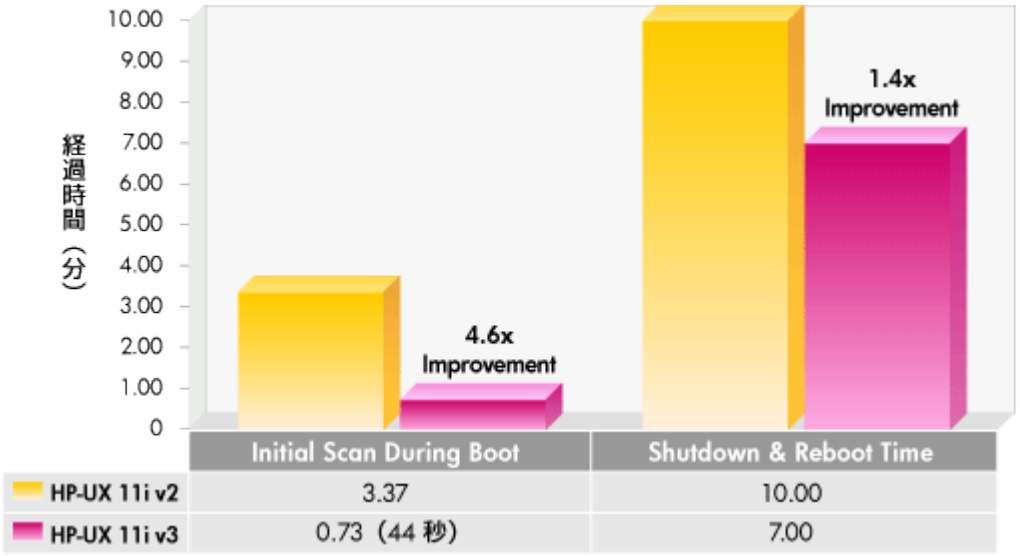


図 2：ブート時間も大幅に短縮されている

以上のように、HP-UX11i v3 のストレージスタックのパフォーマンスアップは、単にカタログスペックの数字として現れているだけではなく、利用者が実感できるほどの向上を実現しているのだ。

その理由の 1 つに、内部の大幅なブラッシュアップと、マルチスレッド化が挙げられる。特別企画の第 1 回でふれたように、HP-UX 11i v3 はデュアルコア インテル®Itanium®プロセッサのハイパースレッディング技術に対応していることが特徴だが、ストレージスタックも従来に比べて大幅に並列性が高められている。その結果が ioscan コマンドの実行速度の向上といった目に見える高速化として現れているのだ。

ストレージスタックの並列性の向上は、その他にも、さまざまな恩恵をもたらしてくれる。たとえば、クラッシュ時のダンプ時間の短縮も並列性の向上によるものだ。HP-UX 11i v3 は、同じ構成のシステムでクラッシュ時のダンプ時間が従来の 1/4 以下にまで短縮された例があるという。

続く後編では、新しいデバイスの表記形式「Agile View」と、ロードバランシングをサポートするネイティブ・マルチパスを紹介したいと思う。

ストレージ管理を容易にする Agile View とネイティブ・マルチパス

2007 年 6 月　テクニカルライター 米田 聡

大規模なストレージに対応できるスケーラビリティが大きな特徴である HP-UX11i v3。続く後編では、新しいデバイスの表記形式「Agile View」と、ロードバランシングをサポートするネイティブ・マルチパスを紹介する。大規模化によって複雑になりがちなストレージの管理・運用を容易にする HP-UX11i v3 の新機能だ。

管理を容易にする新しいデバイスの表記形式 Agile View とは

前編で紹介したように、HP-UX 11i v3 は大規模なストレージに対応できるスケーラビリティが大きな特徴となっている。しかし、ストレージの大規模化は管理・運用の複雑化に直結しがちだ。ストレージの増強によって、たとえばホスト・バス・アダプタ（HBA）やストレージの構成が変更されたとしよう。従来までの HP-UX では OS の設定やアプリケーション側にも変更が必要になり、その手間が管理者の大きな負担になっていた。

そこで HP-UX 11i v3 では、ストレージの変更に柔軟に対応する新しいデバイスの表記形式が追加された。新しい表示形式を Agile View といい、それに対応する形で従来の形式を Legacy View と呼ぶ。両者の違いを詳しく説明していこう。

従来のデバイス・パスの表示形式は多くの読者がご存じだろう。HBA までのアドレスを/で区切り、以降のアドレスを.で区切る形式が用いられている。たとえば次のような形式だ。

```
0/0/10/1/0.8.0.255.0.1.0
```

このような形式であるため、ストレージ構成の変更が HP-UX 側の変更に直結していたわけだ。

一方、HP-UX 11i v3 の Agile view では、LUN への物理的なパスを表す Lunpath hardware path と、SCSI デバイスのユニークな識別子「WWID」（Worldwide Identifier）を用い管理され、LUN への仮想的なパスを表す LUN hardware path の 2 種類の表示形式が用意されている。

Lunpath hardware path は、従来の Legacy Hardware Path を SCSI-3 のアドレッシング方式に対応させたもので、例えば次のような形式でデバイス・パスを表記する。

```
0/0/10/1/0.0x21000004cfa80f29.0x0
```

0/0/10/1/0 は従来通り HBA までのアドレスを/で区切ったものだ。その後ろに“.”で区切って FC ターゲット・ポートの Worldwide Name、デバイスの LUN ID が続く。

もう一つの LUN hardware path は、ある特定の LUN を表す「仮想的なパス」で、物理的なパスとは無関係に生成されるデバイス・パスだ。LUN hardware path は仮想的なルートノードである 64000 で始まり、たとえば次のように表記する。

```
64000/0xfa00/0x2
```

LUN hardware path は LUN そのものを指すデバイス・パスで、ストレージの構成――たとえばハードウェア的なパスなどに左右されない。これは HP-UX 11i v3 で新しく導入されたパスの概念である。

以上、HP-UX 11i v3 には、Legacy Hardware Path と Lunpath hardware Path、LUN Hardware Path の 3 種類のデバイス・パスの表記形式があるわけだ。それぞれの関係を次の図に示そう。

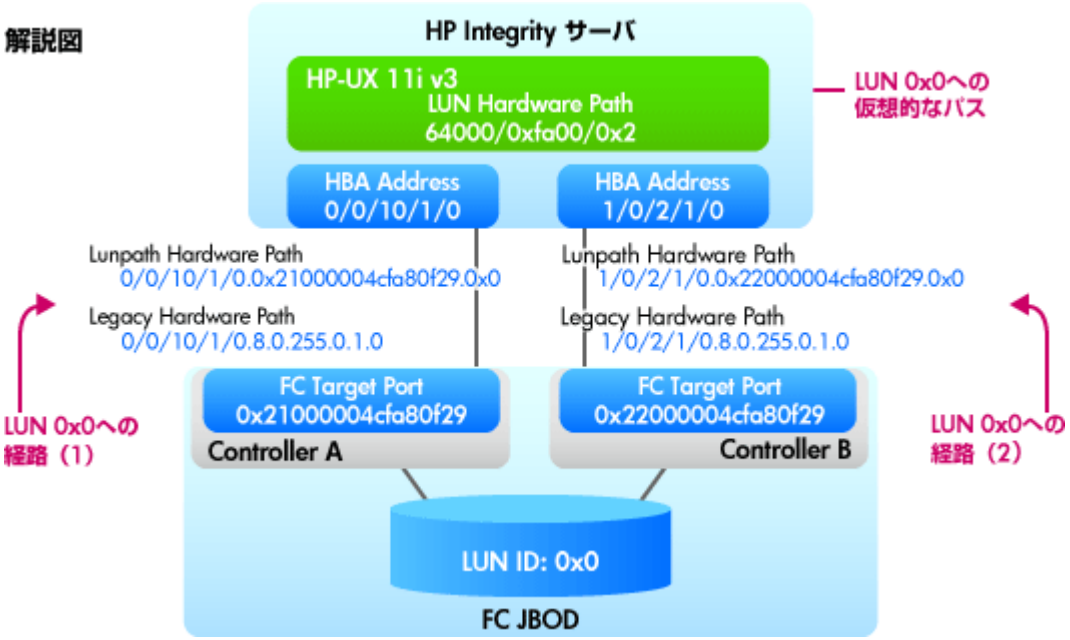


図 3：同じ LUN を表す 3 種類のデバイス・パス

図は、2 つの HBA を用いて 2 つの経路で 1 つの LUN に接続されている、ごく簡単なシステムの例だ。LUN への物理的な経路には Legacy Hardware Path と、新しい Lunpath Hardware Path という 2 種類の表記があり、さらに LUN そのものを表す（経路に左右されない）LUN hardware path がある。いずれのパス表記も同じ LUN を指していることに注意しよう。

これらのパスの対応関係は以下のようなコマンドで確認できる。

```
# ioscan -m hwpath -H 64000/0xfa00/0x2
Lun H/W Path      Lunpath H/W Path                  Legacy H/W Path
================================================================
64000/0xfa00/0x2
                  0/0/10/1/0.0x21000004cfa80f29.0x0   0/0/10/1/0.8.0.255.0.1.0
                  1/0/2/1/0.0x22000004cfa80f29.0x0    1/0/2/1/0.8.0.255.0.1.0
```

以上のデバイス・パスに加え、ストレージ構成の柔軟性を損なう要素にデバイス・ファイルの存在がある。多くの読者はご存じだろうが、従来のストレージに対応するデバイス・ファイルは HBA のインスタンス番号と LUN やターゲットのアドレスを組み合わせて生成されていた。たとえば次のような形式だ。

```
/dev/disk/c0t4d0
```

c0 の 0 が HBA のインスタンス番号、t4 の 4 がターゲットのアドレス、d0 の 0 が LUN のユニット番号だ。

このように従来のデバイス・ファイルの表記形式（Legacy DSF という）は実際の経路に基づいて生成されるので、構成の変更等でターゲットへの経路が変わればデバイス・ファイルも変わらざるをえず、ストレージ構成の変更が設定などの変更に直結してしまう。

そこで、HP-UX 11i v3 では物理的な経路に基づいてではなく、デバイスそのもの（すなわち前述の LUN Hardware Path）に対応するデバイス・ファイルの表記形式（Persistent DSF という）が追加されている。従来に比べて非常にシンプルなデバイス・ファイルの表記である。

`/dev/disk/disk13`

Persistent DSF はデバイスの LUN Hardware Path に対応して管理されているため、たとえ経路が変わっても同じデバイスには常に同じデバイス・ファイルが対応する。したがって、ストレージ構成の変更に柔軟に対応できるのである。

次の表に、ハードウェア・パス表記形式とデバイス・ファイル表記形式の新旧をまとめたので参考にしてほしい。

	Agile View（11i v3）	Legacy View（11i v2 以前）
ハードウェア・パス表記形式	Lunpath Hardware Path 例：0/0/3/0.0x6.0x0	Legacy Hardware Path 例：0/0/10/1/0.8.0.255.1.0
仮想ハードウェア・パス表記形式	LUN Hardware Path 例：64000/0xfa00/0x2	N/A
デバイス・ファイル表記形式	Persistent DSF 例：/dev/disk/disk21	Legacy DSF 例：/dev/dsk/c0t0d0

ロードバランシングをサポートするネイティブ・マルチパス

以上、HP-UX 11i v3 で導入された新しいデバイス・パス表記形式やデバイス・ファイルの表記形式を紹介したが、これらの導入はストレージの柔軟な構成変更や容易な管理を実現するだけでなく、HP-UX11i v3 からサポートされた「ネイティブ・マルチパス」とも密接に関連している。

従来から HP-UX には、ボリュームマネージャを使用する LVM PVLinks を用いたマルチパスや、サードパーティのマルチパスソリューション――たとえば VxVM Veritas DMP などが存在していた。逆に言うと、マルチパスを利用するにはこれらの導入が欠かせなかったわけだが、HP-UX11i v3 では OS そのものがロードバランシング型のマルチパスに対応した。

先に Persistent DSF は経路に関係なく生成されるデバイス・ファイルであると述べた。したがって、デバイス・ファイルを通じてストレージを操作すれば、OS が自動的に適切な経路（lunpath）を選択してデバイスへのアクセスが行われる。さらに、デフォルトでは Legacy DSF を通じてストレージを操作しても経路の選択は自動で行われるようになっている。

経路の振り分けは、LUN に設定されているロードバランシング・ポリシーに従って行われる。ネイティブ・マルチパスでは、ディスクデバイスに対して 4 種類のポリシーをサポートしている。

表：ディスクデバイスでサポートされる 4 種類のロードバランシング・ポリシー

Round Robin (round_robin)	すべてのアクティブなパスに現在の負荷とは関係なく、均等に I/O を分散
Least Command Load (least_cmd_load)	最も I/O 負荷の少ないパスを選択して I/O を実行
Cell Round Robin (cl_round_robin)	セル・ベースのサーバーに最適なロードバランス方式。I/O が発行された CPU と同じ locality に存在するパス間で Round Robin に I/O を実行

Preferred Path (preferred_path)	特定のパスに I/O を発行。ppreferred_path 属性でパスを別途指定する必要がある

以上の 4 種類に加え、テープ、チェンジャーなどシーケンシャル・デバイスをサポートするポリシーとして、ブート後、最初にオープンされたパスのみが利用される「Path Lockdown(path_lockdown)」が用意されている。

ロードバランシング・ポリシーの変更は動的に可能

ロードバランシング・ポリシーは scsimgr コマンドを用いて動的に変更することが可能だ。以下に、実際の運用例を紹介しよう。次の図のように 1 つの LUN に対して 4 つの経路を持つ、一般的な例を取り上げる。

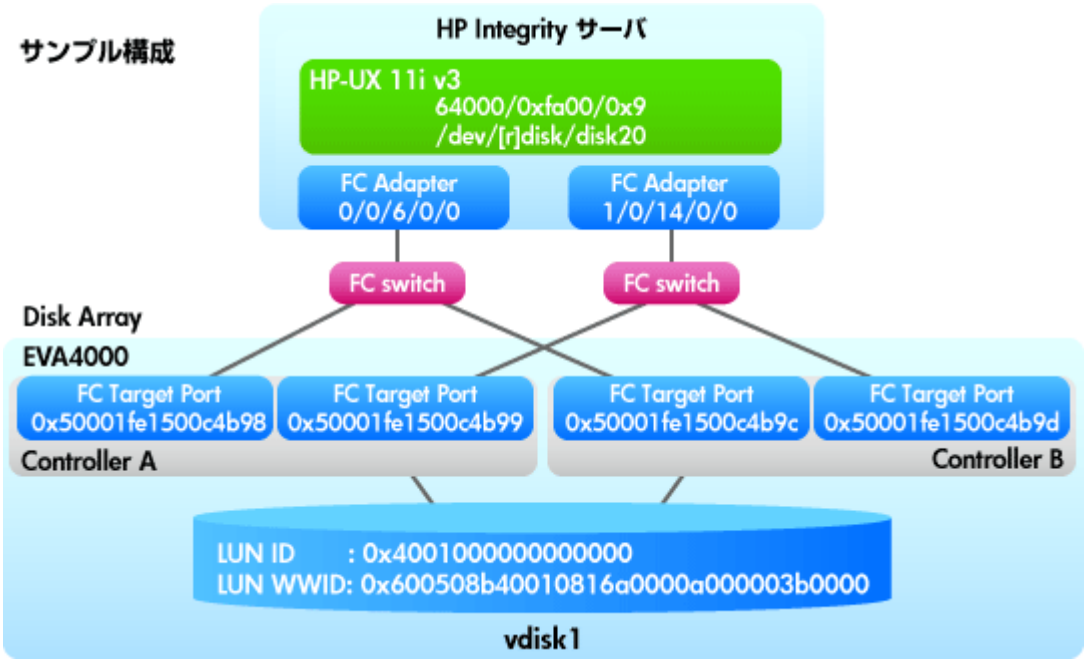


図 4：1 つの LUN に対して 4 つの経路を持つサンプル構成

この構成で、ioscan コマンドを実行すると次のように 4 つの経路（Lunpath）が確認できる。

```
# ioscan -m hwpath
Lun H/W Path        Lunpath H/W Path                          Legacy H/W Path
================================================================
64000/0xfa00/0x9
                    0/0/6/0/0.0x50001fe1500c4b98.0x4001000000000000
0/0/6/0/0.1.0.0.0.1
                    0/0/6/0/0.0x50001fe1500c4b9c.0x4001000000000000
0/0/6/0/0.1.4.0.0.1
                    1/0/14/0/0.0x50001fe1500c4b99.0x4001000000000000
1/0/14/0/0.1.0.0.0.1
                    1/0/14/0/0.0x50001fe1500c4b9d.0x4001000000000000
1/0/14/0/0.1.4.0.0.1
```

1つのLUNに対して4つのパス（lunpath）が存在

図 5：ioscan -m hwpath コマンドでマッピング情報を表示

scsimgr コマンドで現在のロードバランシング・ポリシーを確認してみよう。

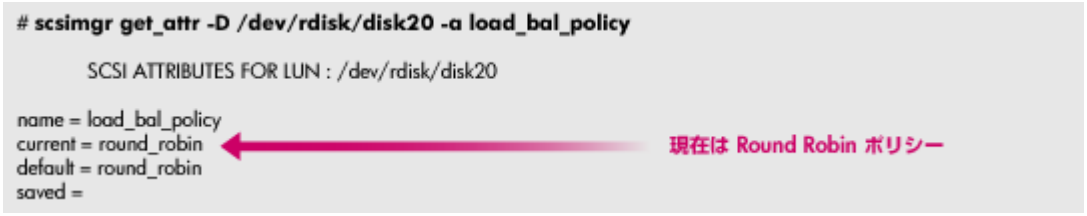


図 6：ロードバランシング・ポリシーを表示

この例のように現在のロードバランシング・ポリシーが round_robin（表参照）であると分る。変更はごく容易だ。ロードバランシング・ポリシーを least_cmd_load に切り替える例を紹介しておこう。

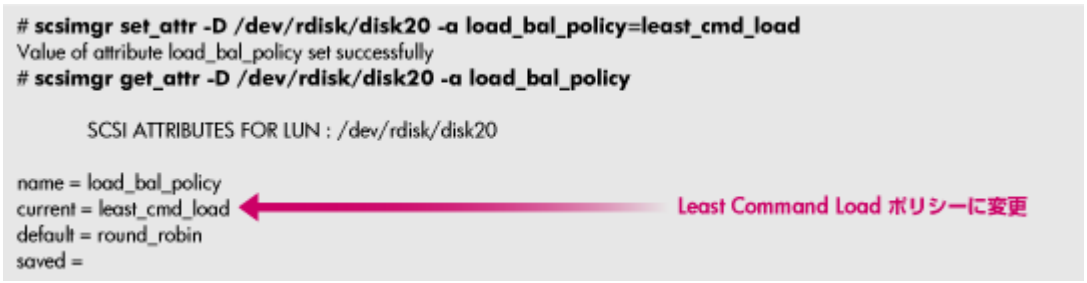


図 7：ロードバランシング・ポリシーを変更

以上の紹介でおわかりのように、HP-UX 11i v3 のネイティブ・マルチパスは、OS そのものがサポートしているため、アプリケーション側ではまったく意識する必要がない。たとえば、デバイスファイルの表記を Persistent DSF に切り替えなければならない、といった制限もない。したがって、従来のアプリケーションのまま、ロードバランシング型の高いパフォーマンスを持つ I/O の恩恵が受けられるのである。

Oracle 環境向け高速化ツール“ODM”で「raw デバイスの悩み」を解消しよう

2007 年 7 月　テクニカルライター 吉川 和巳

Oracle の構築では常識となっている raw デバイスは、同時にさまざまな「管理のしにくさ」の原因ともなっている。そうした中、「raw デバイスのパフォーマンス」と「ファイルシステムの管理性」の双方を“いいとこ取り”したソリューションが現れた。それが、「Serviceguard Storage Management Suite for HP-UX(SG SMS)」に含まれる Oracle 環境向け高速化ツール「Oracle Disk Manager(ODM)」である。Oracle データベースの構築は「raw デバイス」で――そんな“常識”も、いまや変わりつつあるようだ。

「raw デバイス」の常識が変わる？

Oracle データベース(以下、Oracle)の構築は「raw デバイス」で――そんな“常識”も、いまや変わりつつあるようだ。　ご存じのとおり、Oracle では、データを格納する方法として、以下のいずれかから選択できる。

- OS のファイルシステム
- raw デバイス

例えば前者のファイルシステムを選択した場合、Oracle はファイルシステム上に通常のファイルを作成し、その中にレコードを格納する。これに対し、後者の「raw デバイス」では、OS のファイルシステムを作成しない“生(raw)”のディスク・パーティションを利用する。つまり、OS のファイルシステムを介さずに、Oracle がディスクに直接アクセスし、データを記録する形態である。

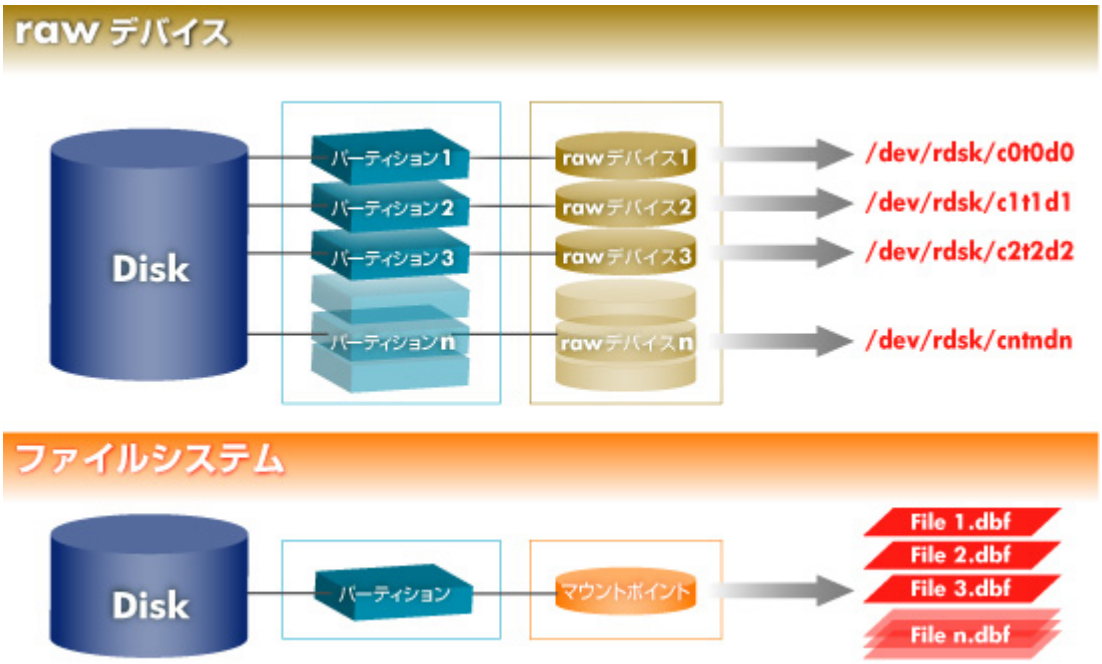


図 1：raw デバイスとファイルシステム

Oracle の構築においてファイルシステムよりも raw デバイスが好まれる理由は、リレーショナル・データベース(RDB)というソフトウェアの性質を考えれば明らかだ。RDB は、リレーショナル・モデルに基づいて、「テーブル」や「レコード」といった論理単位でデータを管理するソフトウェアである。大量のレコードを効率的に検索するためのメカニズムや、テーブルの読み書きを高速化するバッファ(Oracle で言えば SGA)をそなえる。これに対し OS のファイルシステムは、ツリー構造(ディレクトリ)に基づいて、「ファイル」という論理単位でデータを管理するソフトウェア。多数のファイルを効率よく管理し、ファイルの読み書きを高速に実行するためのバッファを提供する。

このように、データベースとファイルシステムはデータ・モデルや目的が異なるうえ、キャッシュ機構もそれぞれが個別に備えている。つまり、Oracle のデータをファイルシステム上に保存することは、「英語をいったんフランス語に訳してから日本語に訳す」ようなものなのだ。とりわけ、ファイルシステムと Oracle のそれぞれのバッファ機構が重複することによるオーバーヘッドは大きい。

そこで、ファイルシステムを介さずに、Oracle が独自のフォーマットでディスク上にデータを記録する方法が raw デバイスである。raw デバイスを用いて Oracle を構築した場合、Oracle 独自の形式でディスク・パーティション上に直接データを配置する。よって上述の二度手間を省き、パフォーマンスが大幅に向上するのである。どうしてもパフォーマンスを考慮する必要のある実運用での環境下では、raw デバイスを選択して Oracle を構築することの方が一般的であると言えるだろう。

raw デバイスは管理しにくい

しかし、raw デバイスには「管理しにくい」という大きなデメリットがある。ファイルシステムを持たないため、OS からは「ディスク上のひとつのボリューム」としか認識されない。よって raw デバイスに対してシェルから操作できることと言えば、ボリューム単位でのバックアップ/リストアくらいだ。データの一覧表示や移動、コピーといった簡単な操作も、すべて Oracle 上の管理コマンドで実施する必要が生じる。

データの管理に通常のシェルが使えないことの弊害は大きい。緊急時のリカバリ作業のみならず、日常の運用業務でさえ、Oracle 管理のスキルを持つコストの高い管理者が必要となる。また、他のアプリケーションに対しては有用な運用管理用のシェル・スクリプトやツール群も、raw デバイスに対してはほとんど役に立たない。よって、システムを構成する数あるミドルウェアやアプリケーション、ツール群の中でも、Oracle だけを特別扱いする必要が生じる。また Oracle 内部でも、ログ・ファイルや設定ファイル、Oracle のバイナリ・ファイルなどはファイルシステムで管理する。よってそれらのバックアップやリストアは、raw デバイスとは個別に実施しなければならない。

そのうえ raw デバイスには、ファイルシステムのようなパーミッションの概念がない。したがって管理者のちょっとしたオペレーション・ミスによって、raw デバイス全体を上書きしたり削除したりする可能性もある。ディレクトリの概念もないので、raw デバイスの数が多い構成では管理が煩雑となる。

さらに raw デバイスの利用時には、データベースがオンラインの状態で領域を動的に拡張することができない。そのため、個々のテーブルについてレコード件数の増加にともないどの程度の領域が消費されるか、あらかじめ厳密に見積っておく必要が生じる。もし raw デバイスの領域を使い切ったら、サービスを止めての拡張作業が必要になる。これに対し、例えば HP-UX のオプション機能「OnlineJFS」では、アプリケーションの稼働中にもファイルシステム領域の動的拡張が可能だ。サービスを止めずとも、必要に応じてディスクを追加するだけで、さらに多くのデータを格納可能になる。したがって複数のテーブルをひとつのファイルシステム上に集約して一括管理でき、上述のような厳密な見積もりや面倒なメンテナンス作業は不要だ。

このように、Oracle の構築では常識とされてきた raw デバイスも、振り返ってみればさまざまな弊害を生み出していることがわかる。そうしたなか、「raw デバイスのパフォーマンス」と「ファイルシステムの管理性」の双方を“いいとこ取り”したソリューションが現れた。それが、「Serviceguard Storage Management Suite for HP-UX(SG SMS)」に含まれるツール「Oracle Disk Manager(ODM)」である。

SG SMS と Oracle Disk Manager

「Serviceguard Storage Management Suite for HP-UX(SG SMS)」は、クラスタウェアである Serviceguard と、シマンテックのストレージ管理ツール VERITAS Storage Foundation のバンドル製品である。HP-UX 標準のボリュームマネージャやファイルシステムを上回る可用性や管理性を実現するツール群から構成されている。以前の特集記事「Serviceguard＋クラスタ・ファイルシステムでラクをする」で紹介した「Serviceguard Cluster File System」をはじめ、今回紹介する「Oracle Disk Manager(ODM)」、ストレージの I/O パスの冗長化や障害復旧機能、スナップショット作成機能などを含む。

表：Serviceguard Storage Management Suite に含まれるツール群

ツール	機能
Volume Management and File System(VxVM/VxFS)	ストレージのオンライン再構成、オンラインでのボリューム／ファイルシステム作成／サイズ変更、障害ストレージのオンライン再配置など

Cluster File System (CFS)	クラスタ・ファイルシステム
Oracle Disk Manager (ODM)	raw デバイス相当のパフォーマンスをファイルシステムで実現
Flashsnap	データベースのクローンを作成
Multi-Volume File System (MVFS)	パフォーマンスや可用性の異なる複数のボリュームから、1 つのファイルシステムを構成
Portable Data Container (PDC)	異なる OS 間でのディスク共有
Dynamic Multi-Pathing (DMP)	ホストバス・アダプタとストレージ間を複数の I/O パスで結び、負荷分散とフェイルオーバーを実現
Storage Checkpoints	ファイルシステムのロールバックを実現

さて、SG SMS のツールのひとつである ODM とは、「raw デバイスの性能とファイルシステムの管理性の両立」をめざして、1999 年以前から旧ベリタス(現シマンテック)とオラクルが共同で開発を進めてきた技術だ。オラクルが策定した ODM の API をベースにベリタスが製品を提供するかたちで、2001 年には Oracle 9i 対応の ODM が出荷開始されている。この ODM が SG SMS のツールのひとつとしてリリースされたことで、HPE が正式サポートするソリューションとして HP-UX 上での利用が可能になった。

では、ODM はどのようにしてファイルシステムによる Oracle パフォーマンスを向上させるのだろうか。簡単にまとめると、ODM は以下のような特徴を備える。

- 通常のファイルシステム（VxFS）とまったく同じ使い勝手を提供
- カーネルレベルで非同期 I/O を実装し、Oracle による複雑な I/O リクエストを一括して実行
- ファイルシステムによるバッファリングをスキップ

これらの詳細は続く後編で説明するとして、まずはその効果のほどを見てみたい。

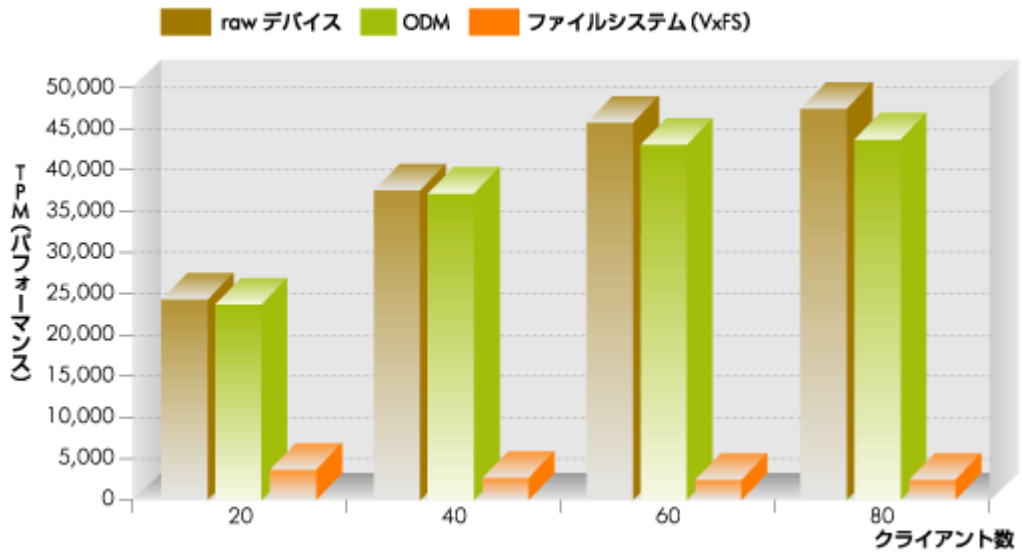


図 2：raw デバイス・ファイルシステム・ODM の Oracle 性能比較

このグラフは、先ほど紹介した raw デバイスとファイルシステムによる Oracle 性能の計測結果に、ODM による計測結果(赤色)を追加したものだ。ごらんのとおり、ODM は raw デバイスとほぼ同等のパフォーマンスを達成しており、raw デバイスと比較したオーバーヘッドは数%程度にとどまる。その一方で、HP-UX 標準装備の VxFS とまったく同じ使い勝手を提供し、管理者から見れば VxFS のファイルシステム上に Oracle を構築した場合と同様に管理できる。

このように、ODM では raw デバイスを上回る性能が得られるわけではないが、ここまでオーバーヘッドが低いのであればファイルシステムの管理性のメリットをぜひ享受したいと考える管理者は少なくないだろう。

つづく「Oracle 環境向け高速化ツール“ODM”はなぜ速いのか？」では、ODM がパフォーマンスと管理性の両立をいかにして成し遂げているか、メカニズムの詳細に迫ってみたい。

Oracle 環境向け高速化ツール“ODM”はなぜ速いのか？

2007 年 7 月　テクニカルライター 吉川 和巳

「Oracle Disk Manager(ODM)」の大きな特徴は、HP-UX のカーネルレベルで非同期 I/O を実装している点だ。またファイルシステムのバッファをスキップする Direct I/O モードを用いることで、バッファ重複によるオーバーヘッドを解消し、かつサーバーのメモリをフルに SGA に割り当て可能になる。さらに通常のファイル記述子を使わず、プロセス間で共有可能な ODM 識別子によりファイルを管理し、カーネルリソースの消費が減少する。こうした数々の工夫によって、ODM では raw デバイス並みの性能をファイルシステムで実現することに成功している。

ODM が速い、その理由

「Oracle 環境向け高速化ツール“ODM”で「raw デバイスの悩み」を解消しよう」でも説明したとおり、Oracle Disk Manager(ODM)は以下の 3 つの特徴を備えている。

- 通常のファイルシステム（VxFS）とまったく同じ使い勝手を提供
- カーネルレベルで非同期 I/O を実装し、複数の I/O リクエストを一括実行
- ファイルシステムによるバッファリングをスキップ

まず管理者から見て ODM は、ファイルシステム（VxFS）上に Oracle データベース(以下、Oracle)を構築して運用するケースとまったく同じ使い勝手を提供する。つまり ODM の導入にともない、Oracle の構築手順や運用手順が大きく変わる部分はない。よって、前編で述べたような raw デバイスの管理性の低さを解消し、ファイルシステムの利便性をフルに発揮できる。

次に、ODM の大きな特徴は、HP-UX のカーネルレベルで非同期 I/O を実装している点だ。前編でも述べたとおり、ODM はオラクルが策定した専用 API に基づいて開発された製品であり、ODM が備わる環境では Oracle が同 API を通じて非同期の I/O リクエストを発行する。これについて、以下に詳しく説明しよう。

周知のとおり、Oracle のメモリバッファである SGA とディスク間のデータ読み書きを司るのは、DBWR プロセスである。よって Oracle をファイルシステム上に構築すると、この DBWR が read()/pread()や write()/pwrite()といった標準的な UNIX の I/O システムコールを繰り返し実行し、ファイルシステムを介したデータアクセスを実行する。

こうした標準的なファイル I/O を通じて Oracle を動作させる場合、次のような問題が生じる。

- **多数の同期・非同期 I/O が発生**
  さまざまな種類の I/O リクエストが多数発生するため、ファイルシステムが介在することによるオーバーヘッドが生じる。同期 I/O も多いため、データが実際にディスク上に書き出されたり、ディスク上から読み込まれたりするまで、リクエスト元のスレッドは待ち状態となる。非同期 I/O についても、処理完了を確認するポーリングが必要となる。
- **データのフラグメンテーション**
  データベースのように大規模の連続したデータに特化していないので、データがディスク上の各所に分散して記録される。
- **ファイルオープンの競合**
  あるプロセスが書き込み中のファイルは OS によってロックされるため、他のプロセスが同時に書き込むことができない。
- **バッファの重複**
  Oracle とファイルシステムのそれぞれが個別にバッファを持ち、オーバーヘッドが生じるほか、サーバーのメモリ・リソースがムダに分割されてしまう。

そこで ODM では、標準のシステムコールの代わりに、Oracle によるファイルアクセス専用の非同期 I/O システムコールである odm_io()を利用する。これにより、以下のようなメリットが得られる。

- **I/O リクエストを少数の非同期 I/O に集約**
  odm_io()システムコールは非同期であるため、DBWR のスレッドはディスクの読み書き完了を待つ必要がなく、直ちに次の処理に取りかかれる。また、odm_io()では DBWR が発する複数の I/O リクエストをひとつに集約し、まとめて OS に指示できる。そのため I/O 処理の回数が減り、オーバーヘッドが減少する。
- **フラグメンテーションの防止**
  ディスク上の連続した領域にファイルを記録するため、大規模の連続したデータに効率よくアクセスできる。
- **ファイルロックのバイパス**
  ファイルオープン時の競合を解消する。
- **Direct I/O モードの利用**
  ファイルシステムのバッファをスキップする Direct I/O モードを用いることで、バッファ重複によるオーバーヘッドを解消し、かつサーバーのメモリをフルに SGA に割り当て可能になる。

- **ODM 識別子の利用**
  通常のファイル記述子を使わず、プロセス間で共有可能な ODM 識別子によりファイルを管理し、カーネルリソースの消費が減少する。

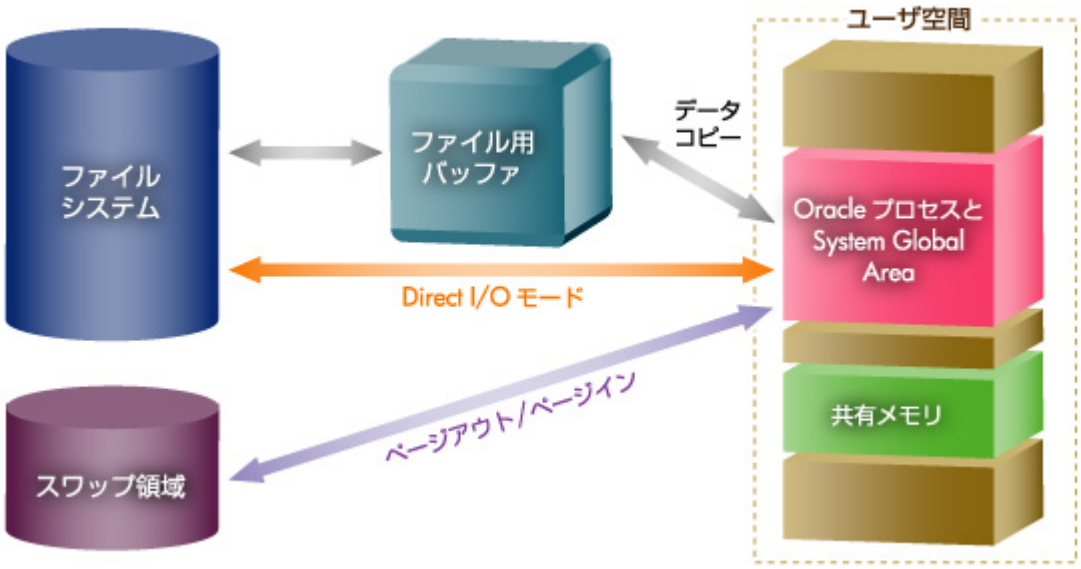


図 1：Direct I/O モードによるバッファ重複の解消

このように、ODM では Oracle に特化したシステムコールや I/O メカニズムを用意することで、ファイルシステムを扱いながらもオーバーヘッドをきわめて低く抑えることに成功している。

ODM の実力を探る

HP では、この ODM が実際にどの程度の能力を発揮できるか検証すべく、パフォーマンス・テストを実施した。このテストでは、以下のような検証システムが用意され、実運用環境に近い形でのパフォーマンスが計測されている。

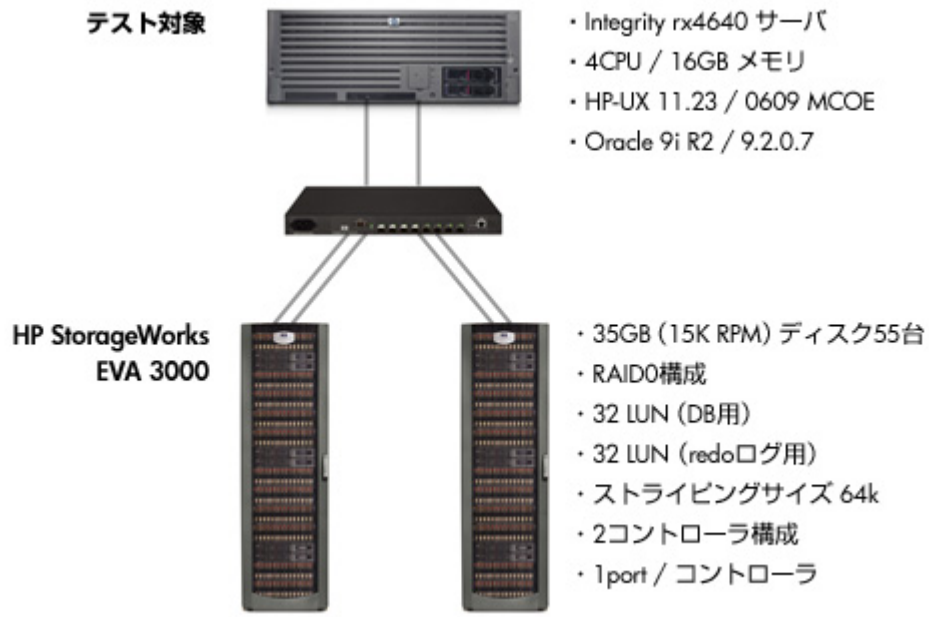


図 2：ODM 検証テストのシステム構成

この図が示すように、テスト対象となる Oracle 9i R2 が稼働する Integrity rx4640 に対して、テスト・クライアントから大量のリクエストを発行する構成である。またストレージについては、2 式の EVA3000 を用意し、RAID0 構成で動作する 55 台のディスクを装着。32 の LUN に対して LVM ストライピングを行う設定とした。以上の構成にて、以下の 5 種類のテストを実施した。

VxVM でのテスト

- raw デバイス
- ODM＋VxFS
- VxFS のみ

LVM でのテスト

- raw デバイス
- VxFS のみ

このように、HP-UX 標準の 2 種類の論理ボリュームである VxVM と LVM のそれぞれについて、raw デバイスと ODM、ファイルシステム（VxFS）による Oracle パフォーマンスを計測する。なお、ODM は LVM をサポートしないため、LVM では ODM による値は計測されていない。

以下のグラフは、これらすべてのテスト結果をまとめたものである。

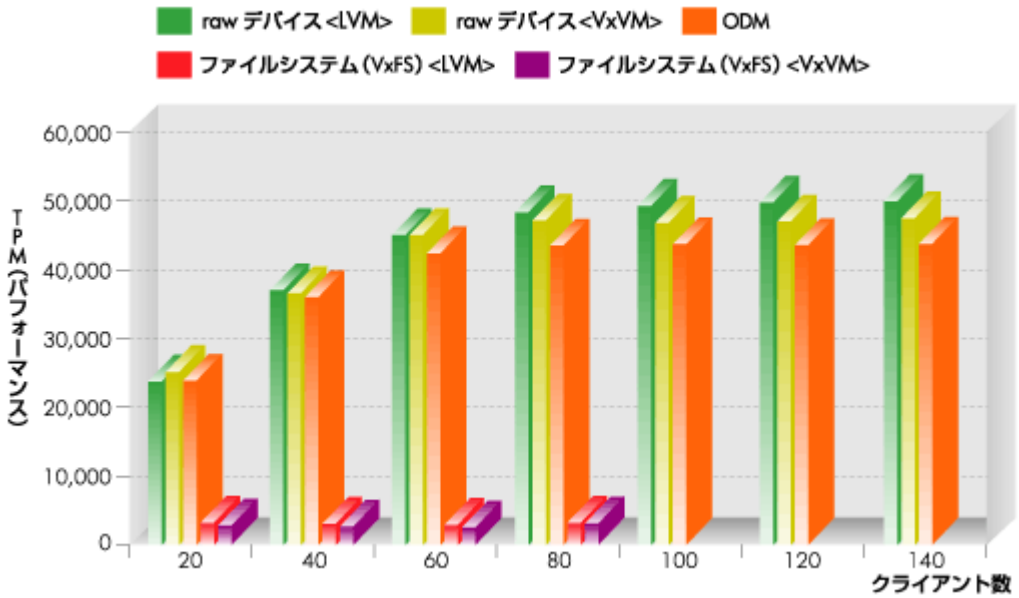


図 3：Oracle パフォーマンスのテスト計測結果

ここで、縦軸は 1 分間あたりのトランザクション数、そして横軸は同時接続クライアント数を指す。このテスト結果から導かれる結論について、HP では以下のようにまとめている。

- ODM は、ファイルシステム上の Oracle パフォーマンスを劇的に改善する
- Oracle パフォーマンスを最大限に引き出すには、従来通り raw デバイスの利用が必要だが、ODM が提供する優れた管理性とのトレードオフとなる
- ODM と raw デバイスのパフォーマンス差は 8%以内であり、管理性を重視する場合には問題とならないレベルである

ここまで見てきたとおり、ODM はまさしく「ファイルシステムの利便性」と「raw デバイスの性能」を兼ね備えたソリューションだ。もちろんテスト結果が示すように、ODM は raw デバイスのパフォーマンスを上回るわけではない。よって、ODM が提供する優れた管理性と、raw デバイスとのわずかなパフォーマンス差のどちらを優先するかのトレードオフとなる。しかし ODM の導入により、コストの高い Oracle 管理者への依存度を下げることができ、場合によってはこのパフォーマンス差を補ってあまりある TCO 削減が実現できるだろう。Oracle 構築における「raw デバイス」の常識は、ODM の登場によって大きく変わりつつあるようだ。

ソフトウェア・パッチの管理コストを削減する柔軟な管理ツール群

2007 年 8 月　テクニカルライター 大津 真

HP-UX のようにミッションクリティカルな分野で使用されることが多い OS では、システムになんらかの不具合、あるいはセキュリティ上の脆弱性がないかについて常に気を配る必要があることはいうまでもない。そこで重要なのが、システムの修正ファイルとして提供される「パッチ」を適切に選択／適用する、いわゆる「パッチの管理」である。本稿では、2007 年度に新たにリリースされた管理ツールを中心に、HP-UX におけるソフトウェア・パッチの管理について解説していこう。

HP-UX における「パッチ」とは？

「パッチ（patch）」とは、日本語では洋服の“つぎあて”のような意味だが、コンピュータの世界では一般にソフトウェアの差分修正ファイルのことをいう。また、パッチを使用してシステムに変更を施すことを「パッチを当てる」あるいは「パッチを適用する」などという。HP-UX では、システムやソフトウェア製品の不具合を修正したり、新機能を提供したりするための、いわゆるアップデータが「パッチ」として適宜提供される。
まずは、HP-UX におけるパッチの概要について簡単にまとめておこう。

ソフトウェア製品とパッチの共通ファイル・フォーマット　SD フォーマット

HP-UX では、すべてのソフトウェア製品およびパッチ・ファイルには、「SD フォーマット」と呼ばれる統一されたファイル・フォーマットが使用される。HP-UX におけるソフトウェア管理の中核をなすツール群である Software Distributor UX（SD-UX）でそれらのファイルを操作するための、階層化されたフォーマットだ。何はともあれ、次の図を見てほしい。

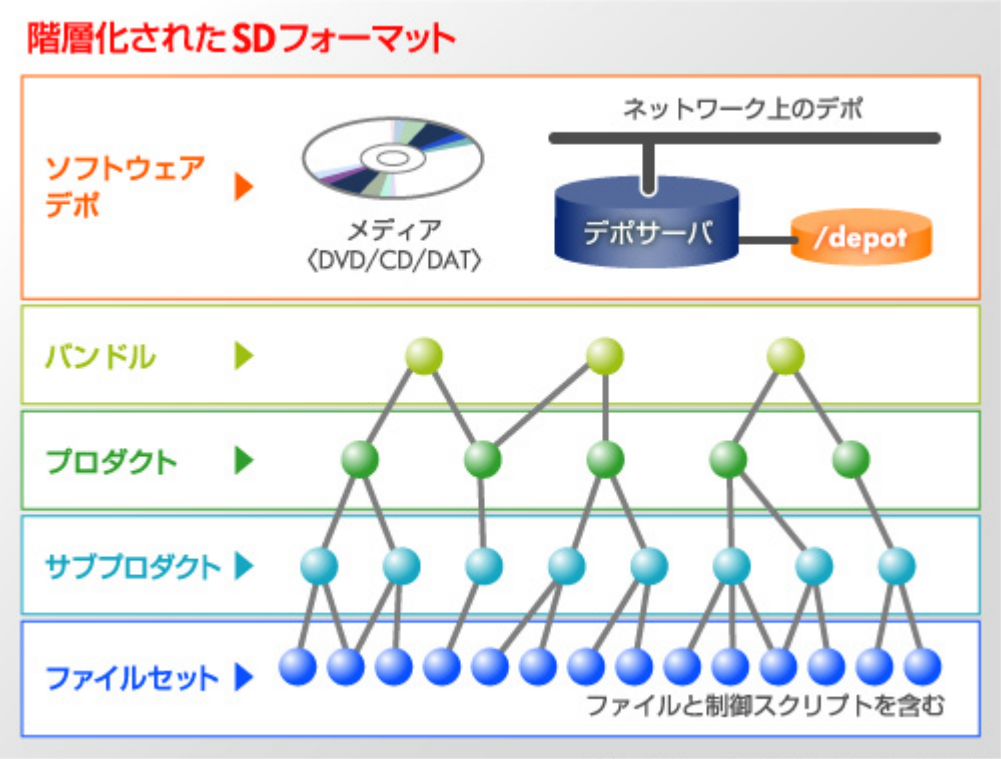


図 1：階層化された SD フォーマット

このように SD フォーマットは、パッケージ化されたソフトウェアの単位である「デポ」を頂点とする階層構造を持っている。

- **ファイルセット**　ファイルの集まり
- **サブプロダクト**　ファイルセットの集まり
- **プロダクト**　ファイルセットもしくはサブプロダクトの集まり
- **バンドル**　プロダクトもしくはファイル設定の集まり

この階層構造のおかげで、多くのファイルから構成されるソフトウェア製品およびパッチの柔軟な管理が可能になるのだ。Red Hat 系の Linux をご存じの方は、「プロダクト」は RPM パッケージ、「バンドル」がパッケージグループ、「デポ」はリポジトリに相当するものといったイメージでととらえていただければよいだろう。SD フォーマットは Red Hat 系 Linux のそれに比べて、より詳細な構造を持っているわけだ。

個別パッチとパッチバンドル

パッチは、単独の「個別のパッチ」と、複数の個別パッチをまとめてグループ化した「パッチバンドル」の 2 種類に大別される。さらに HP では、HP によってすべてのパッチをグループとして厳しく検証したパッチバンドルを、「標準 HP-UX パッチバンドル」として提供している。これは、Windows における「サービスパック」のような存在と考えて差し支えない。
以下に、標準 HP-UX パッチバンドルの主なメリットをまとめておく。

- パッチ適用の時間を短縮し、エラーのリスクを低減する。
- 依存関係を満たすために必要なすべてのパッチが標準 HP-UX パッチバンドルに含まれるように、あらゆる依存関係分析を行っているので安心。
- 複数のパッチを個別にインストールする場合は各パッチについてリブートが必要になることがあるが、標準パッチバンドルのインストールではシステムを複数回リブートする必要はない。

- バンドルを使って標準パッチデポを作成できるため、複数のシステムに簡単に導入することができる。

表 1：標準 HP-UX パッチバンドルの種類

パッチバンドル	機能
Base Quality Pack（QPKBASE）	コア HP-UX、グラフィックス、および主要なネットワークドライバ用の安定した不具合修正パッチ
Applications Quality Pack（QPKAPPS）	OE アプリケーションに対する、安定した不具合修正パッチ
Hardware Enablement（HWE）	HP-UX を使用する新しいハードウェアとレガシーハードウェアをサポートする最低限のパッチセット
Required Patch Bundle（BUNDLE11i）	新機能や新しい製品に必要とされるパッチ（HP-UX Virtual Partitions（vPars）の機能、USB-00 など）

HP-UX サポートの総本山　IT リソースセンター

HPE のサポートサイトの総本山と言えるのが、IT リソースセンター（以下 ITRC）の Web サイトである。無償のユーザアカウントを登録すれば、個別パッチ／標準 HP-UX パッチバンドルのデータベースを検索し、必要なパッチをダウンロードすることが可能だ。

個別パッチのインストール例

あらかじめ必要な個別パッチがわかっている場合には、ITRC に用意されているパッチデータベースで検索し、ダウンロードした後にインストールすればよい。以下に 「PHCO_32254」というパッチを例に、個別パッチを手動でインストールする典型的な手順を示す。パッチ名の接頭辞「PH」は「Patch HP」を、「CO」は「コマンドパッチ」であることを示し、その後ろに 5 桁のシリアルナンバーが続く。

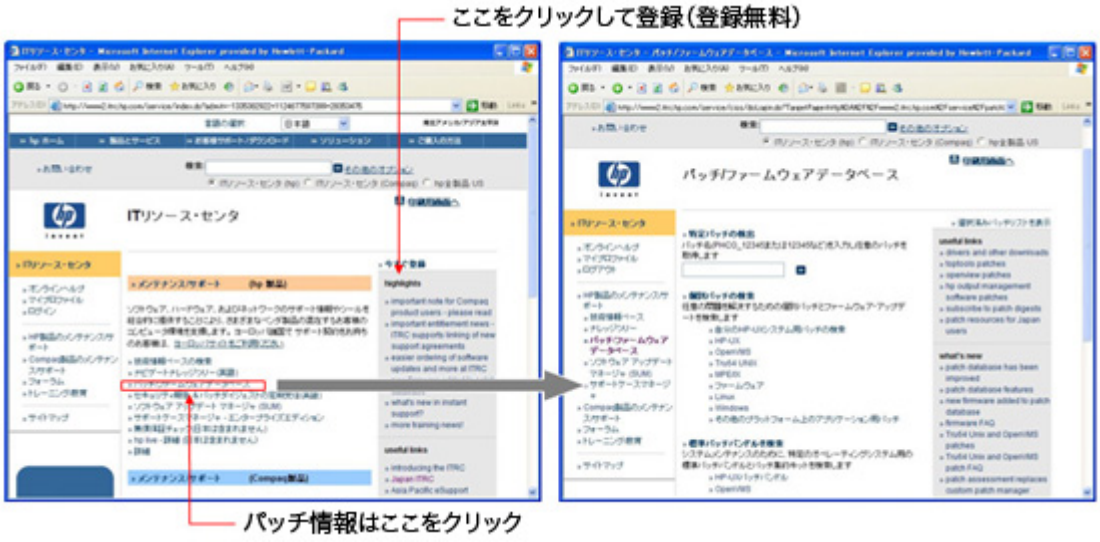


図 2：ITRC からパッチをダウンロードする

1. Web ブラウザで、ITRC に接続し、「パッチ/ファームウェアデータベース」へ移動する。「特定パッチの検索」に進み、パッチ名「PHCO_32254」を検索し、ダウンロードする。
2. ダウンロードしたパッチ「PHCO_32254」は、シェルアーカイブになっている。sh コマンドでこれを展開する。

```
# sh PHCO_32254
```

3. すると、パッチデポ「PHCO_32254.depot」が作成されるので、Software Distributor UX（SD-UX）の swinstall コマンドを実行しインストールする。

```
# swinstall -x autoreboot=true -x patch_match_target=true ¥
  -s /tmp/PHCO_32254.depot
```

ここで指定した「-x autoreboot=true」オプションは、リブートが必要なパッチを適用する際に、自動的にシステムを再起動するオプションである。ただし、リブートを必要としないパッチに対して指定してもリブートされることはない。
なお、指定したパッチに依存関係のあるパッチが設定されている場合がある。さらに依存関係に階層構造が伴う場合もある。ITRC ではこれらの依存関係をすべて解析し、ひとまとめにしてダウンロードできるという優れた機能が提供されている。複数のパッチとともに create_depot_hp-ux_11 というスクリプトも添付されているので、入手できた複数のパッチからデポのディレクトリを作成することが容易に実現できる。あとはすでに説明した swinstall の手順で -s オプションに対して create_depot_hp-ux_11 スクリプトで作成されたデポディレクトリを指定すればいいだけである。

パッチ評価 Web ページの活用

実際問題、パッチ管理においてもっともやっかいな作業のひとつが、現在のシステムに必要なパッチをどのようにして判断するかという点だろう。実は、ITRC のサイトには、システムを精査し必要なパッチを推奨してくれる「パッチ評価 Web ページ」（Patch Assessment Tool for HP-UX）が用意されている。このツールを使用することで、環境に固有のパッチを簡単にダウンロードすることが可能だ。

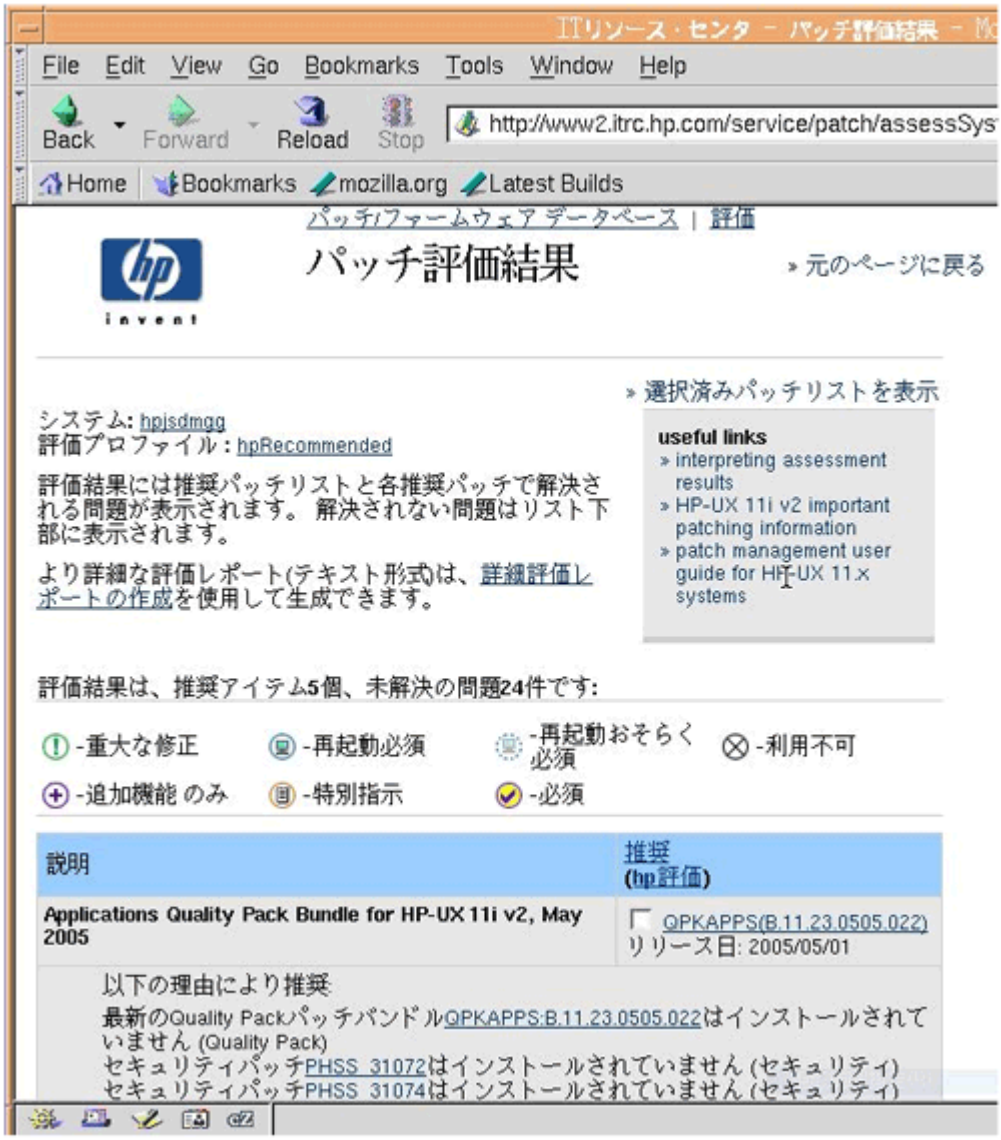


図 3：パッチを推奨してくれるパッチ評価 Web ページ

ソフトウェア・パッチの管理ツール群

HP-UX にはさまざまなソフトウェア・パッチ管理ツールが用意されている。次の表に、主要なツールの一覧を示す。

表 2：ソフトウェア・パッチ管理ツール群

ツール	用途	コマンドの種類
Ignite-UX	OS のインストール、システムイメージの配布	GUI、TUI
Software Distributor UX（SD-UX）	ソフトウェア製品、パッチのインストール	GUI、TUI、CUI
update-ux	OS のアップデート	TUI、CUI

Software Package Builder（SPB）	SD フォーマットのカスタムパッケージの作成	GUI、CUI
Security Patch Check（SPC）	セキュリティパッチの診断	CUI
パッチ評価 Web ページ	必要なパッチの診断とダウンロード	WEB
Software Assistant（SWA）	**必要なパッチの診断とサポート情報へのオンラインアクセス、パッチのダウンロード**	**CUI**
Software Manager（SWM）	**ソフトウェア製品／パッチのインストール、オペレーティング環境のアップデート**	**TUI、CUI**
Dynamic Root Disk（DRD）	**システムイメージのクローンの作成とメンテナンス**	**CUI**

これらのツール群の中で、SWA 以降の 3 つが新たにサポートされたものである。その中で SWM は SD-UX とほとんど同じ機能を持ったツールだが、SWA と DRD は HP-UX には新しい機能で、より効率的なパッチ管理を行うのに有効なツールであるといえるだろう。DRD については後編で紹介することにして、前編では SWA のメリットと使用方法について説明しよう。

パッチ管理を簡略化する SWA

2007 年 1 月にリリースされた「HP-UX　Software Assistant（SWA）」は、なにかと煩雑になりやすいパッチ管理を効率化するコマンドライン・ツールである。HP-UX 11i (v1/v2/v3) システムで使用可能だ。

前述のように、これまでも ITRC のサイトに用意されている「パッチ評価 Web ページ」を利用することで、システム環境に応じた推奨パッチの選択／ダウンロードが可能であった。ただし、そのためには Web ブラウザを立ち上げて、ITRC のサイトにアクセスし、システム情報のアップロードやパッチ評価の実行、パッチの選択といった個別の操作をする必要があった。また、パッチをインストールするに必要なパッチ用のデポの作成は、ダウンロード後に手動で行う必要があった。さらに、セキュリティ関連パッチに関するレポートの作成に関しては、別途 Security Patch Check（SPC）ツールを使用しなければならなかった。

それに対して、SWA は「パッチ評価 Web ページ + SPC」の機能を有する、より高機能なツールだ。ターミナル上で、コマンドを 2 回実行するだけで、システムを解析し、パッチ・レポートの作成から、ダウンロード、デポの作成までが完了する。

次の図は、「パッチ評価 Web ページ」を基本にしたパッチ管理の手順と、SWA を基本にした手順の比較である。後者では、手順が大幅に簡略化されているのがおわかりいただけるだろう。

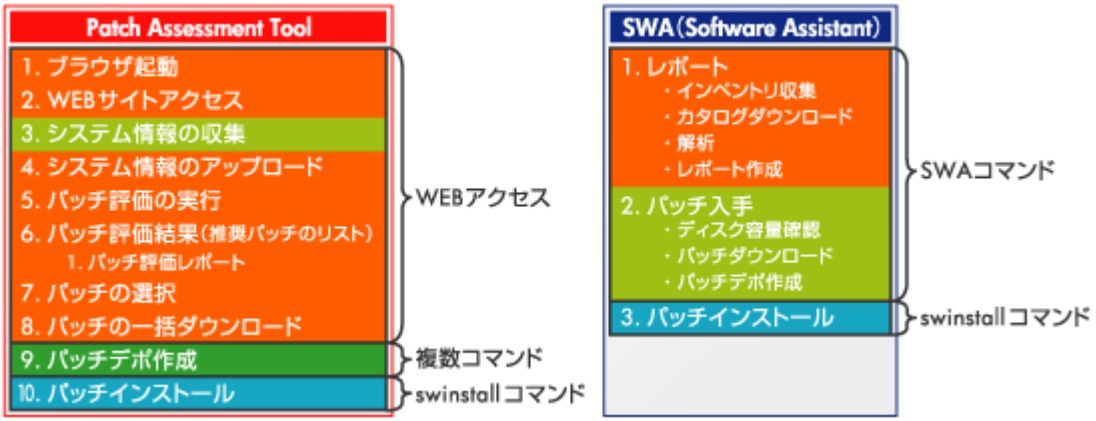


図 4：パッチ評価 Web ページと SWA との手順比較

それでは、実際に SWA を使用したパッチ・レポートの作成から、推奨パッチのダウンロード/インストールまでの流れを見ていくことにしよう。

SWA を使用したパッチ・レポートの作成

まず、「swa report」コマンドを実行し、システムのインベントリの収集/アップロード、および HPE が提供するカタログのダウンロードを（インターネット経由で）実行し、システムの解析を行う。

```
# swa report

=======  07/09/07 17:58:29 JST  BEGIN Report on Issues and New Software (user=root)
         (jobid=hpjmktg1)

       * Gathering Inventory
       * Getting Catalog of Recommended Actions and Software
       * Download complete: 8926kB
       * Performing Analysis
       * Generating Reports
NOTE:    See HTML-formatted report "/.swa/report/swa_report.html"
                             Software Assistant
                          Actions Summary Report

ASSESSMENT PROFILE

Catalog Information
   Catalog File: /.swa/cache/swa_catalog.xml
   Catalog Date: 09 July 2007 15:00:26 JST

Inventory Source
   Name: hpjmktg1
   OS: HP-UX B.11.31
   Model: ia64 hp server Integrity Virtual Machine
   Inventory File: /.swa/cache/swa_inventory_1757671707.xml
   Inventory Date: 09 July 2007 17:58:31 JST

Analysis Information
   Analysis File: /.swa/cache/swa_analysis.xml
   Analysis Date: 09 July 2007 17:58:53 JST
   Ignore File(s): /.swa/ignore
   Issues Ignored: 0

   Selected Analyzers
```

```
      QPK: latest Quality Pack patch bundle
      SEC: security bulletins
      PCW: patches with critical warnings

RECOMMENDED ACTIONS

Patch Bundles
```

swa report コマンドの実行例

「swa report」コマンドの実行が完了すると、推奨されるパッチなどの情報が記述されたレポートが HTML ファイルとして作成される。
デフォルトの出力先ファイル名は、「.swa/report/swa_report.html」。

RECOMMENDED ACTIONS AND RESOLVED ISSUES(DETAILS)

Install PHSS_36123 rating=1,cumulative patch

Issue ID	SEC:02225r1:PHSS_36123 (SEC analyzer)
Bulletin	02225r1
Bulletin Date	2007-06-12
Bulletin Title	HPSBUX02225 SSRT071295 rev.1 - HP-UX Running Xserver, Local Denial of Service (DoS)
Detection Confidence	definite
URL	http://itrc.hp.com/service/cki/docDisplay.do?docId=emr_na-c01075678-1

Install PHSS_36361 rating=1,ent Version 1.3.5.03 Cumulative patch

Issue ID	SEC:02217r2:PHSS_36361 (SEC analyzer)
Bulletin	02217r2
Bulletin Date	2007-05-25
Bulletin Title	HPSBUX02217 SSRT071337 rev.2 - HP-UX running Kerberos, Remote Arbitrary Code Execution
Detection Confidence	definite
URL	http://itrc.hp.com/service/cki/docDisplay.do?docId=emr_na-c01056923-2

Install CIFS-Server revision A.02.03.02 or subsequent

Issue ID	SEC:02204r1:UPD_003 (SEC analyzer)
Bulletin	02204r1
Bulletin Date	2007-04-04
Bulletin Title	HPSBUX02204 SSRT071341 rev.1 - HP-UX Running CIFS Server (Samba), Remote Denial of Service (DoS)
Detection Confidence	definite
Vulnerability Summary	A potential security vulnerability has been identified with HP-UX running CIFS Server (Samba). This vulnerability may allow a remote unauthorized user to create a Denial of Service (DoS).
URL	http://itrc.hp.com/service/cki/docDisplay.do?docId=emr_na-c00943462-2
Issue ID	SEC:02218r1:UPD_003 (SEC analyzer)
Bulletin	02218r1
Bulletin Date	2007-06-18

SWA を使用したパッチのダウンロード、デポの作成

続いて、「swa get」コマンドを実行し、推奨パッチのダウンロード（インターネット経由で実施）、およびデポの作成を行う。

```
# swa get -v -t /tmp/patch

=======  07/09/07 20:40:18 JST  BEGIN Get New Software From HP Preview (user=root)
         (jobid=hpjmktg1)

       * Analyzing Required Disk Space
       * System Information
         Name: hpjmktg1
         OS: HP-UX 11.31
```

```
        Model: ia64 hp server Integrity Virtual Machine
        Inventory File: /.swa/cache/swa_inventory_1757671707.xml
        Inventory Date: 09 July 2007 17:58:31 JST
      * Analysis Information
        Analysis File: //.swa/cache/swa_analysis.xml
        Analysis Date: 09 July 2007 17:58:53 JST
        Catalog: /.swa/cache/swa_catalog.xml
        Catalog Date: 09 July 2007 15:00:26 JST
        Ignore File: /.swa/ignore
      * Selected Analyzers
        QPK - latest quality pack bundle
        SEC - security bulletins
        PCW - patches with critical warnings
      * Software Cache
        /var/opt/swa/cache
      * Target Depot
        /tmp/patch
      * Downloading Software from HP to Local Cache
NOTE:   Estimated total download size:  33248580 bytes.
      * Downloading PHSS_36361 (1 of 2)
      * Successfully downloaded PHSS_36361
      * Downloading PHSS_36123 (2 of 2)
      * Successfully downloaded PHSS_36123
      * Copying Software into Target Depot
      * Processing /var/opt/swa/cache/hp-ux_patches/11.X/PHSS_36123
      * Processing /var/opt/swa/cache/hp-ux_patches/11.X/PHSS_36361
      * Registering /tmp/patch

=======  07/09/07 20:40:23 JST  END Get New Software From HP Preview succeeded.
         (user=root)  (jobid=hpjmktg1)

NOTE:    More information may be found in the Software Assistant logfile
         "/var/opt/swa/swa.log".
```

swa get コマンドの実行例

上の例では、/tmp/patch ディレクトリにパッチ用のデポが作成される。

パッチのインストール

後は、SD-UX に用意されている swinstall コマンドを実行して、作成されたデポからパッチのインストールを行えばよい。

```
# swinstall -x autoreboot=true -s /tmp/patch ¥*

=======  07/20/07 18:09:40 JST  BEGIN swinstall SESSION
         (non-interactive) (jobid=hpjmktg1-0016)

      * Session started for user "root@hpjmktg1".

      * Beginning Selection
      * Target connection succeeded for "hpjmktg1:/".
      * Source connection succeeded for "hpjmktg1:/tmp/patch".
      * Source:                /tmp/patch
      * Targets:               hpjmktg1:/
      * Software selections:
            PHSS_36123.OEM-SERVER,r=1.0,a=HP-UX_B.11.31_IA/PA,v=HP,fr=1.0,fa=HP-
UX_B.11.31_IA
            PHSS_36361.KRB5-64SLIB,r=1.0,a=HP-UX_B.11.31_IA/PA,v=HP,fr=1.0,fa=HP-
UX_B.11.31_IA
＜以下略＞
```

SWA のコマンドリファレンス

ここまでの説明で、SWA を使用することにより、システムに応じた推奨パッチのダウンロード/インストールが簡単に行えることがおわかりいただけただろう。前編の最後として、SWA のコマンドの概要をまとめておこう。

表 3：SWA のコマンドリファレンス

コマンド	機能
swa report	パッチレポートを作成する
swa get	パッチのダウンロードとパッチデポを作成する
swa clean	ディスクにキャッシュされたパッチを削除する
swa step	パッチ管理作業をステップごとに実行する

表 4：swa report コマンドの解析オプション

オプション	機能
CRIT	重大な問題の修正パッチ
PCW	重大な警告に対するパッチ
PW	警告に対するパッチ
QPK	最新のクオリティパック
SEC	セキュリティパッチ

稼働中のシステムイメージのダイナミックな複製を可能にする DRD

2007 年 8 月　テクニカルライター 大津 真

安定したシステム運用のためには、適切なパッチの適用やソフトウェアのアップデートなどのメンテナンス作業が不可欠である。後編では、現在稼働中のシステムのイメージを複製（クローン）することで、パッチ評価用のテスト環境の構築を可能にし、さらにシステムのメンテナンス時におけるダウンタイム短縮に貢献する Dynamic Root Disk（DRD）について紹介する。

メンテナンスやバックアップの際に有効活用

HP-UX の管理ツールとして新たに登場した「Dynamic Root Disk（以下 DRD）」は、稼働中のシステムのイメージを停止することなく、複製することを可能にするツールだ。このとき、現在稼働中のシステムイメージを「アクティブ・システムイメージ」、複製された側のシステムイメージを「非アクティブ・システムイメージ」と呼ぶ。複製された非アクティブ・システムイメージに対して、DRD に用意されているコマンドを使用することにより。安全にパッチを適用することが可能だ。その後、複製した非アクティブ・システムイメージをアクティベートして、そこからシステムを起動することができる。つまり、アクティブ・システムイメージと非アクティブ・イメージを"ダイナミック"に入れ替えることが出来るわけだ。
次の図は、DRD による複製の流れである。

図 1：DRD による複製の流れ

DRD の機能は、複製した非アクティブ・システムイメージをどのように扱うかによって、「Hot Recovery」と、「Hot Maintenance」に大別することができる。

代替システムイメージへのワンポイント切替を可能にする Hot Recovery

DRD の用途としてまず考えられるのが、非アクティブ・システムイメージを緊急時のバックアップとして使用する「Hot Recovery」である。オリジナルのシステムイメージでなんらかの問題が発生した場合、事前に作成しておいた非アクティブ・システムイメージから瞬時にシステムを起動できる。 この場合の実行手順の概略は次の通りだ。

1. 「drd clone」コマンドで複製を作成する(-t オプションで複製先のデバイスを指定) 。

```
# drd clone -t /dev/dsk/c1t2d0
=======  01/23/07 11:16:18 JST  BEGIN Clone System Image (user=root)
         (jobid=hpjmktg1)
```

```
      * Reading Current System Information
      * Selecting System Image To Clone
      * Selecting Target Disk
      * The disk "/dev/dsk/c1t2d0" contains data which will be overwritten.
      * Selecting Volume Manager For New System Image
      * Analyzing For System Image Cloning
      * Creating New File Systems
      * Copying File Systems To New System Image
      * Making New System Image Bootable
      * Unmounting New System Image Clone
      * System image: "sysimage_001" on disk "/dev/dsk/c1t2d0"

=======  01/23/07 11:46:15 JST  END Clone System Image succeeded. (user=root)
         (jobid=hpjmktg1)
```

2. 「drd activate」コマンドを実行し、非アクティブ・システムイメージをアクティベートする（OS 起動パスを設定）。

```
# drd activate
=======  01/23/07 14:58:18 JST  BEGIN Activate Inactive System Image
          (user=root)  (jobid=hpjmktg1)

      * Reading Current System Information
      * Locating Inactive System Image
      * Determining Bootpath Status
      * Primary bootpath : 1/0/0/3/0.3.0 before activate.    <====== 現行稼働 OS のパス
      * Primary bootpath : 1/0/1/1/0/1/1.2.0 after activate. <====== 複製した OS のパス
      * Alternate bootpath : 1/0/1/1/0/4/0 before activate.
      * Alternate bootpath : 1/0/1/1/0/4/0 after activate.
      * To return the primary boot path to its value before "drd activate ",
        issue the command
        "setboot -p  1/0/0/3/0.3.0"
      * Activating Inactive System Image

=======  01/23/07 14:58:33 JST  END Activate Inactive System Image succeeded.
         (user=root)  (jobid=hpjmktg1)
```

3. システムをリブートする。

アクティブなシステムイメージが入れ替わる。つまり、これまでアクティブ・システムイメージだったものが非アクティブ・システムイメージに、非アクティブ・システムイメージだったものがアクティブ・システムイメージに入れ替わるわけだ。

より安全なパッチの適応やシステム修正を可能にする Hot Maintenance

DRD のコマンドを使用すると、オリジナルのシステムイメージを変更することなく、非アクティブ・システムイメージに対してパッチの適用などの修正を加えることが可能だ。これを「Hot Maintenance」と呼ぶ。通常、システムのリブートが必要な複数のパッチを当てる場合、その数だけシステムを立ち上げ直す必要があるが、DRD を使用するとリブートが一度で済むためダウンタイムを短くできるわけだ。さらに、パッチの適用によってなんらかの問題が生じた場合にも、オリジナルのシステムイメージには変更を加えていないので簡単に戻すことが可能である。

なお、Hot Maintenance を使用してパッチを適用するには、前述の Hot Recovery の「手順 2（drd activate）」の前に、非アクティブ・システムイメージに対してパッチを適用すればよい。それには、「drd runcmd <SD コマンド>」を実行する。ここで、「SD コマンド」とは、ソフトウェア・パッチの管理コマンド群である Software Distributor UX（SD-UX）に用意されて

いるコマンドだ。たとえば、ソフトウェアのパッチのインストールには SD コマンドとして、swinstall コマンドを指定すればよい（-s オプションで depot のパスを指定）。

```
# drd runcmd swinstall -s depot_server:/var/opt/patches ¥*
```

非アクティブ・システムイメージのマウント

なお、パッチが正しく適用されているのかを調べたい、あるいは非アクティブ・システムイメージ内の設定ファイルの中身を確認/変更したいといった理由で、非アクティブ・システムイメージ内のファイルシステムにアクセスしたいといったケースもあるだろう。
そのような場合には、次のようにして「drd mount」コマンドを実行する。

```
# drd mount
```

これで、非アクティブ・システムイメージが「/var/opt/drd/mnts/sysimage_001」といったディレクトリにマウントされ、自由にアクセスできるようになる。なお、逆に、非アクティブ・システムイメージをアクティベートし、複製されたシステムからブートしていた場合には、「drd mount」コマンドを実行するとオリジナルであったシステムイメージが「/var/opt/drd/mnts/sysimage_000」にマウントされる。

DRD の想定する環境について

HP-UX 11i では通常、複数のディスクをひとつのグループにまとめ仮想的なパーティションを切り出すことを可能にする Logical Volume Manager（LVM）でディスクを管理しているが、ルート・ファイルシステムは「vg00」という名前のボリュームグループ内に置かれる。DRD ではオリジナルのシステムイメージが、その vg00 内に存在しているものとして処理を行う。vg00 以外のボリュームグループに存在するファイルシステムは複製の対象外となるので注意してほしい。
また、複製先はオリジナルと同等以上のディスク容量がある物理ディスクである必要があるが、OS ブート可能なものであれば SAN 環境に複製を作成することも可能だ。なお、現在、DRD は HP-UX 11i v2（B.11.23）2004 年 9 月以降と HP-UX11i v3（B.11.31）2007 年 9 月以降のリリースをサポートしている。

DRD の適用パターン

DRD の概要が理解できたところで、3 つの典型的な適用パターンを示そう。

- **障害対策のための DRD 基本パターン**
  まず 1 つ目は基本パターンと言うべきもので、ミラーリングされていない環境のディスクを複製するものだ。必要に応じて個別パッチを適用し、非アクティブ・システムイメージをアクティベートすることで複製されたシステムイメージから OS を起動できるようになる。

図 2：DRD 基本パターン（HW/SW 障害対策）

- **ミラー構成での複製作成**
  DRD では、ハードディスクがミラーリング構成の環境を、複製することも可能だ。

図 3：DRD from Mirrordisk/UX パターン（HW/SW 障害対策）

- **複数のテスト環境の構築**
  テスト環境を構築したり、もしくはある時点でのシステムのスナップショットを保存するといった目的で、複数の複製を作成することもできる。この場合、DRD から直接管理することができるのは最新の複製だけである点に注意されたい。他の複製から OS を起動させたい場合には、ユーザが設定ファイルを変更する必要がある（OS の起動パスを明示的に変更すること）。

図 4：DRD 複数構成パターン（簡単なテスト環境構築）

DRD と Ignite-UX の使い分け

さて、DRD 以前から HP-UX に用意されていたシステムイメージのハンドリング・ツールに Ignite-UX がある。Ignite-UX は、複数のサーバーやパーティション環境への OS イメージの配布に威力を発揮するツールである。今回紹介した DRD は、その Ignite-UX を完全に置き換えるツールではない。両者を適材適所で使い分ける必要がある。両者の相違をおおざっぱに説明すると、DRD のほうはどちらかというと同一システム内のメインテンスを目的にしたツールであり、Ignite-UX はネットワイドでのシステムのインストール／復旧を主体にしたツールといえる。

次表に、DRD による複製されたシステムイメージと、Ignite-UX によって作成されたシステムイメージの利用局面での相違をまとめておくので参考にされたい。

表 1：DRD と Ignite-UX の適材適所

利用局面	DRD	Ignite-UX
パッチの適用管理	オリジナルを変更することなく非アクティブ・システムイメージに適用可能	ゴールデンイメージをインストールする時に適用可能
リホスティング	現時点ではなし（サポート計画はあり）	別サーバーに再構成可能
システムリカバリ	非アクティブ・システムイメージから迅速な復旧が可能（専用のディスクが必要）	ネットワーク経由でのシステム復旧が可能（専用のディスクは不要）

ファイルシステムの変更対応	手動対応が必要	ファイルシステムの再構成やサイズ変更が可能
障害時の対応	障害時に他ディスクの非アクティブ・システムイメージからの復旧	別サーバーに格納されたメディアからの復旧

システムリカバリに関する DRD の優位性

DRD と Ignite-UX の共通点は、どちらも障害発生時のシステムのリカバリツールとして有効な点であるが、単一システムのシステムリカバリ・ツールといった用途でみた場合、DRD には次のような優位点がある。

- テープ装置を必要としない
- ネットワークの負荷を発生しない
- ネットワーク上のセキュリティを考慮する必要がない

Ignite-UX は、システムディスクやルートボリュームグループに重大な故障が発生した場合は、make_tape_recovery または make_net_recovery といったコマンドを使用することにより、信頼性の高い一貫した方法でリカバリすることが可能だ。前者はテープ装置にブート可能なリカバリテープを作成しておきそこからリカバリする機能、後者はネットワーク経由でリカバリする機能だ。
それに対して、DRD のリカバリ機能は、いわば"make_disk_recovery"機能といったものだ。ローカルディスクからのリカバリを行うため、より迅速なリカバリが可能なのはもちろん、特別なテープドライブを必要とせず、ネットワークの負荷も発生しない。さらに、ネットワーク上にデータを流さないためセキュリティ面でも安心である。

DRD のコマンド・リファレンス

DRD には、現時点で、CUI コマンドである drd のみが用意されている。本稿の最後として、drd コマンドの概要をまとめておこう。drd の引数に、次に示す 5 つのサブコマンドのいずれかを指定して実行する。

表 2：drd のサブコマンド

サブコマンド	機能
drd clone	ルート・ファイルシステムを含むボリュームグループ（vg00）の複製を作成する
drd runcmd	複製された非アクティブ・システムイメージに対して、DRD-Safe コマンドを実行する
drd activete	複製された非アクティブ・システムイメージをアクティベートし、そこからブートできるようにする
drd mount	複製された非アクティブ・システムイメージをマウントする
drd umount	複製された非アクティブ・システムイメージをアンマウントする

これらの 5 つのサブコマンドの中で「drd runcmd」が多少特殊で、引数にさらに SD のコマンドを指定して実行する。たとえば、非アクティブ・システムイメージに対してパッチを適用するには「drd runcmd swinstall」を実行する。

表 3：drd runcmd の一覧

SD コマンド	機能
drd runcmd swinstall	パッチをインストールする
drd runcmd swremove	パッチを削除する
drd runcmd swlist	インストールされているソフトウェアの要素とその属性および編成を一覧表示する
drd runcmd swmodify	インストール済みのプロダクトデータベースまたはデポのカタログファイル内の情報を変更する
drd runcmd swverify	パッチのインストール内容を検証する
drd runcmd swjob	パッチのインストールや削除に関するジョブ状況を表示する

なお、これらの 6 つの SD コマンドは、オリジナルのシステムイメージに影響を与えず、非アクティブ・システムイメージにのみに作用するため DRD-Safe コマンドと呼ばれる（詳細は、オンラインマニュアル drd-runcmd（1M）を参照）。
次に、drd コマンドの主なオプションをまとめておく。

表 4：drd のオプション

SD コマンド	機能
-?	使用方法を表示する（他のオプションとは併用できない）
-p	プレビューモードで実行する
-v	詳細なメッセージを表示する
-x extended_option=value	拡張オプションを設定する
-t target_device_file	ターゲットとなるデバイスファイルを指定する

サーバーを止めずに増強できる Dynamic nPar の離れ業

2007 年 9 月　テクニカルライター 吉川 和巳

HPE が今回発表した新技術「Dynamic nPartitions」の最大の特徴は、アプリケーションの動作を止めることなく、Integrity サーバーのプロセッサやメモリを増設できる点だ。
これはもちろん「サーバーの台数の追加」ではなく、1 台のサーバーのプロセッサ数やメモリ容量が、アプリケーションの稼働中に動的に増やせるのである。

これにより、サーバー増強にともなうサービス停止の頻度を劇的に少なくすることができる。これは、一般的なサーバー製品ではまねのできない「離れ業」と言えるだろう。

物理パーティション技術「nPartitions」

Dynamic nPar について説明する前に、Integrity サーバーの以前からの特徴である仮想化技術「nPartitions(以下、nPar)」についておさらいしておこう。Integrity サーバーのミドルレンジ以上のモデルでは、「セルボード(以下、セル)」と呼ばれるモジュールを筐体内に複数搭載している。セルは、通常のサーバーで言えばマザーボードに相当するもので、1 枚のセル上には最大 4 個のデュアルコア インテル®Itanium®プロセッサを搭載できる。例えば Integrity Superdome の場合は、最大 16 枚のセル上に合計 64 個のプロセッサ(128 コア)を搭載可能だ。

Integrity サーバーでは、これら複数のセルを組み合わせて、ひとつの大きなサーバー・マシンを構成する。しかし Integrity サーバーの魅力のひとつとして、1 つの筐体内をセル単位で自由に分割して、複数台の仮想的なサーバーを設けられる点だ。これが、いわゆる物理パーティション技術と呼ばれる nPar である。

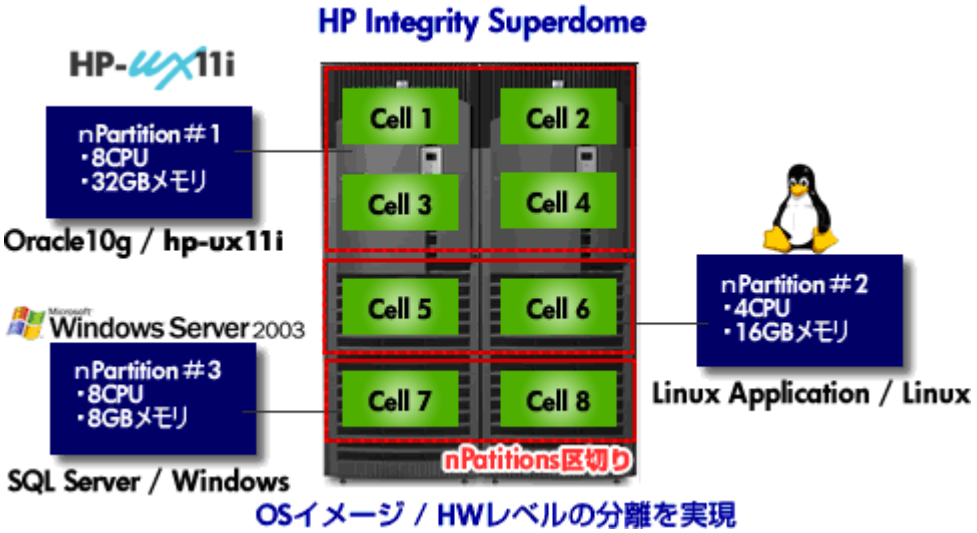


図 1：nPartitions による構成例

上図は、nPar によるサーバー分割の例である。この例では、8 枚のセルを 3 つのパーティションに分割し、それぞれで HP-UX、Windows、Linux という異なる OS を運用している。

これらの nPar は、個々のパーティションが電気的に完全に分離されており、それぞれが単体のサーバーと同等の独立性を備えている。よって万が一いずれかのパーティションでハードウェア障害が発生しても、他のパーティションは一切影響を受けずに継続して運用することができる。

また nPar では、セル単位で柔軟なリソース配分が可能であり、例えば上図の例では HP-UX サーバーの負荷が変化した場合には、nPar の構成を変更し、HP-UX サーバーに割り当てるセル数を増減できる。こうした構成変更は GUI 画面上の設定で可能で、単体のサーバーを増設する場合のように OS やアプリをインストールし直す必要もない。このように nPar は、1 台のサーバーに備わる豊富なリソースを、ビジネス・ニーズに応じて臨機応変に切り出して活用できる仮想化環境を提供する。

nPar から Dynamic nPar へ

こうした多くのメリットを備える nPar だが、これまではひとつ制約があった。それは、nPar の構成を変更する場合、その構成を変更する nPar に関してはリブートが必要になるという点だ。上述の例のように HP-UX サーバーへセルを追加するには、いっ

たんサービスを止めてその nPar をリブートしなければならない。また、セル上に何らかのハードウェア障害(例えばメモリ・モジュールのエラー多発など)が発生し、セルの交換を実施する場合にも、障害を起こしたセルを含む nPar を一度止める必要があった。

従来の nPar が抱えていたこれらの制約を解消したのが、新機能 Dynamic nPar である。Dynamic nPar とは、その名の通り、nPar によるパーティション分割を「動的」に――すなわちリブートせずに――変更可能なテクノロジーだ。そこで、続いてこの Dynamic nPar による動的なパーティション構成変更のメカニズムを紹介したい。

Dynamic nPar のメカニズム

Dynamic nPar は、現在使用しているサーバーが sx1000 もしくは sx2000 チップセットを搭載するセル・ベースの Integrity サーバー(または HP 9000 サーバー)であれば、ファームウェアのバージョンアップのみで利用可能となる。

これは Dynamic nPar の機能を利用するために、現行の Integrity サーバーだけでなく 1 世代前のサーバーも新機能を利用できることになる。新しい機能を利用するために新しい特別な筐体が必要にならない点は大きなメリットになるだろう。
ただし Dynamic nPar は最新の HP-UX 11i v3 0709 版以降でサポートしており、現状では Windows および Linux では利用できない。

Dynamic nPar が従来の nPar と異なるのは、セルが以下の 2 種類に分類される点だ。

- フローティングセル　　パーティション内で自由に追加・削除できるセル
- ベースセル　　パーティション内で固定で割り当てられたセル

フローティングセルとは、Integrity サーバー上のパーティション内で自由に追加・削除できるセルを指す。例えば、アイドル状態のパーティションから高負荷状態のパーティションへフローティングセルを 1 枚移動する、といったリソースの動的配分に利用できる。一方、ベースセルとは、パーティション内で固定で割り当てるセルを指す。フローティングセルとは異なり、ベースセルとしてパーティションに動的に追加することはできるが、動的に削除することはできない。また Dynamic nPar の個々のパーティションには、少なくとも 1 枚以上のベースセルを割り当てる必要がある。

では、このフローティングセルとベースセルの組み合わせによるリソースの動的配分の例を見てみよう。

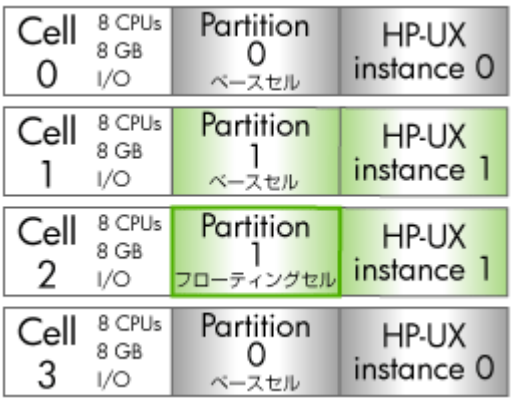


図 2：Dynamic nPar の構成例

上図は、4 枚のセルを搭載した Integrity サーバーにおいて、「Partition 0」と「Partition 1」という 2 つのパーティションを構成した例である。セル 2 のみがフローティングセルであり、残りのセルはすべてベースセルであることに注意していただきたい。

ここで、Partition 0 の負荷が上昇してプロセッサやメモリが逼迫した状況を想定する。一方の Partition 1 の負荷が高くなければ、フローティングセルである Cell 2 を Partition 1 から Partition 0 へ動的に移動することで、負荷上昇にともなうサービス品質の低下を Partition1 で動作するアプリケーションを停止することなく避けることができる(下図参照)。

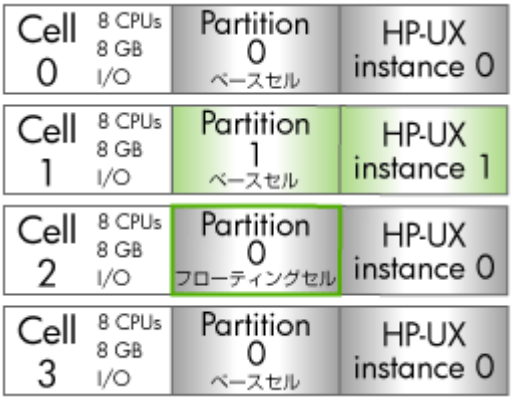


図 3：Dynamic nPar によるリソース配分例

Dynamic nPar はこんなときに便利

従来の nPar であれば、こうしたパーティション構成の変更を行うには、サービスを停止してリブートする必要があった。そのため、前もって負荷上昇が予測できるケースには対応できるものの、予期しない負荷上昇への対応は難しい。
例えば Web サイトやデータベースに一時的にアクセスが殺到しているようなケースでは、負荷対応のためにわずかな時間でもサービスが停止すると、かえって業務に混乱を招きかねないだろう。これに対し Dynamic nPar では、HP-UX が稼働したままの状態で、上述したようなフローティングセルの移動によるスムーズなリソース配分が可能となる。
よってサービスの利用者は、こうしたサーバーの増強が背後で行われていることに一切気づかないはずだ(サービス品質が向上したという変化には気づくかもしれないが)。

もっとも、Dynamic nPar も万能ではない。ひとつ考慮すべき点は、CPU やメモリといったリソースの移動をセル単位でしか実行できないことだ。前述のとおり、1 枚のセルには最大で 4 個のプロセッサが搭載されるため、あまりきめ細かなリソース調整は行えない。もし粒度の細かなリソース配分が必須であれば、vPar や Integrity VM といった他の仮想化技術を選択する必要があるだろう。

Dynamic nPar のもうひとつの用途は、アプリケーション動作中のサーバー・メンテナンスである。例えば Integrity サーバーにセルを増設したいとき、またはセル上のプロセッサやメモリ・モジュールに障害が発生して交換したいとき、これらいずれのケースでも、Dynamic nPar ではシステムをリブートする必要はない。すなわち「セルのホットスワップ」が可能である(ただし、オンラインでのメンテナンス対象はフローティングセルに限られる)。

こうしたセルの追加や交換では、Dynamic nPar から新たに追加された、セルの「アクティブ化(activation)」と「非アクティブ化(deactivation)」という機能が重要な役割を果たす。アクティブ化とは、そのパーティションに割り当てられたセルを OS から動的に利用できるようにする操作を指す。一方、非アクティブ化とは、セルを OS から利用できないようにする操作である。

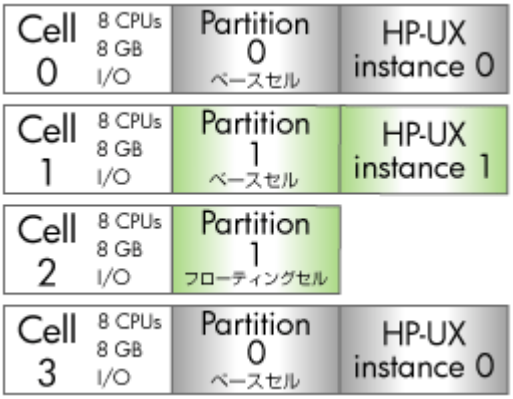


図 4：非アクティブ化されたセルの例

上図の例では、セル 2 が非アクティブ化され、Partition1 に属しているが OS からも切り離された状態となっている。こうしたセルは電源をオフにし、筐体から物理的に取り外すことができる。その後、新しいセルと入れ替えたのち、セルをアクティブ化する。こうした手順によって、故障したセルを含むパーティションで動作するアプリケーションを停止することなくハードウェアの交換やリソースの増強が可能になる。

つづく「容易に扱える”Dynamic nPar”の実行例」では、Dynamic nPar においてセルを取り扱うためのコマンドや実行例について詳しく説明する。

容易に扱える Dynamic nPar の実行例

2007 年 9 月　テクニカルライター 吉川 和巳

ハイエンドやミドルレンジのサーバーでは、サーバー全体のリブートやサービスの停止がとりわけ困難となる。そうした環境では、Dynamic nPar のオンラインでのメンテナンスやパーティション構成変更がきわめて重宝されるはずだ。
また Dynamic nPar で新たに導入された概念やコマンド操作はそれほど難しくはなく、簡単に習得が可能である。ここでは、Dynamic nPar のコマンド操作を、実際の作業の流れに沿って紹介しよう。

Dynamic nPar のコマンド操作の実際

以下では、Dynamic nPar の動作メカニズムについて、コマンドレベルで詳細に解説していきたい。まずは、Dynamic nPar で新たに導入された、フローティングセルの状態遷移について説明する。以下は、フローティングセルが取り得る各状態と、それらの間の遷移に対応するコマンドを示した図である。

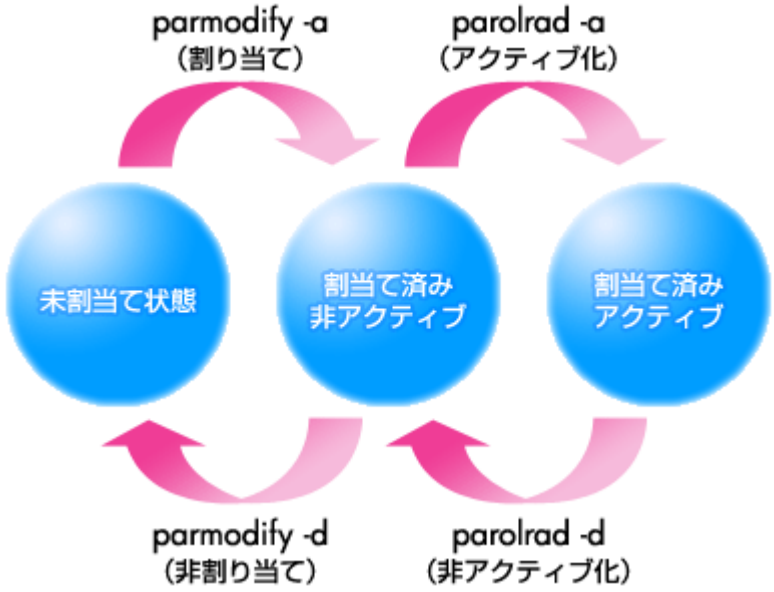


図 1：フローティングセルの状態遷移と操作コマンド

ここで示すように、フローティングセルにはおもに以下の 2 つの状態がある。

- 「未割り当て」と「割り当て済み」　セルが特定のパーティションに割り当てられているかどうか
- 「非アクティブ」と「アクティブ」　セルがパーティションの一部として稼働しているかどうか

ではここで、2 つのパーティション間で 1 枚のフローティングセルを移動するシナリオを想定して、実際のコマンドによる Dynamic nPar の操作例を見ていきたい。まず、Integrity サーバー上のパーティション構成を確認するには、parstatus コマンドを実行する。

```
[par1]# parstatus -C
[Cell]
                        CPU     Memory                                  Use
                        OK/     (GB)                            Core    On
Hardware    Actual      Deconf/ OK/                             Cell    Next Par
Location    Usage       Max     Deconf    Connected To          Capable Boot Num
==========  ==========  ======= ========= ==================    ======= ==== ===
cab0,cell0  Active Core  8/0/8   8.0/0.0   cab0,bay0,chassis0    yes     yes  0
cab0,cell1  Active Core  8/0/8   8.0/0.0   cab0,bay0,chassis1    yes     yes  1
cab0,cell2  Active Float 8/0/8   8.0/0.0   cab8,bay0,chassis0    yes     yes  1
cab0,cell3  Active Base  8/0/8   8.0/0.0   cab8,bay0,chassis1    yes     yes  0
Notes: * = Cell has no interleaved memory.
```

この例では、cell0 から cell3 まで 4 枚のセルが装着された Integrity サーバーのパーティション構成が表示されている。ここで、「Actual Usage」として「Core」または「Base」と表示されているセルはベースセルで、「Float」はフローティングセルである。また「Active」とはセルがアクティブ状態であることを示す。
また、右端にある「Par Num」の値はパーティション番号を指しており、ここではパーティション 0 とパーティション 1 の 2 つにそれぞれ 2 枚ずつセルが割り当てられているのがわかる。

つづいて、cell2 に対して parolrad -d コマンドを実行し、非アクティブ化する。ちなみに非アクティブ化では、セル上のメモリ内容が他のセルへと移動するための時間が必要となるため、コマンドの完了までに数十秒から、場合によっては数十分程度の時間を要することに注意してほしい。(セルの使用状況により操作完了までの時間は変動)

```
[par1]# parolrad -d 2
```

```
Note: Cell Online deactivation operation has been initiated.
The sequence number is: 2
Note: Cell OL* operations, may take several minutes to complete.
Killing or otherwise aborting this parolrad command will not cancel the
Cell OL* operation. See the parolrad (1M) manpage for details.
Online operation in progress ....
Online operation has completed successfully.
```

つづいて後半では、cell2 に対してパーティションの移動を行ってみたい。

パーティション移動とアクティブ化

ここまでの操作により、cell2 は非アクティブ化された状態となったが、依然としてパーティション 1 に割り当てられた状態のままである。
そこで、parmodify -d コマンドを実行し、cell2 をパーティションから外す。続けて parstatus コマンドでパーティション構成を再表示すると、以下のようになる。

```
[par1]# parmodify -p 1 -d 2
[par1]# parstatus -C
[Cell]
                         CPU     Memory                                  Use
                         OK/     (GB)                          Core      On
Hardware    Actual       Deconf/ OK/                           Cell      Next Par
Location    Usage        Max     Deconf    Connected To        Capable   Boot Num
==========  ============ ======= ========= ================== ========= ==== ===
cab0,cell0 Active Core  8/0/8   8.0/0.0   cab0,bay0,chassis0  yes       yes  0
cab0,cell1 Active Core  8/0/8   8.0/0.0   cab0,bay0,chassis1  yes       yes  1
cab0,cell2 Inactive     8/0/8   8.0/0.0   cab8,bay0,chassis0  yes       -    -
cab0,cell3 Active Base  8/0/8   8.0/0.0   cab8,bay0,chassis1  yes       yes  0
Notes: * = Cell has no interleaved memory
```

このように、cell2 は「Inactive（非アクティブ）」状態であり、かつパーティションに割り当てられていない状態であることがわかる。

つづいては、parmodify -a コマンドを実行し、cell2 をパーティション 0 に割り当てする。

```
[par0]# parmodify -p 0 -a 2:floating:y:float:100%
Note: It may take a longer time for the partition to boot if
any cell is performing power-on selftest.
To activate newly added cells, reboot the partition for reconfiguration.
or activate a Cell Online using "parolrad" command.
On HP-UX use "shutdown -R" or "parolrad -a".
If the partition is at the system firmware prompt, use the RR command on the MP.
```

このように、同コマンドのオプション-a にて「2:floating:y:float:100%」と指定することで、cell2 をフローティングセルとしてパーティション 0 に割り当てできる。ちなみに、こうしたパーティションへのセルの割り当て操作は、数秒間程度で完了する。

ここからさらに cell2 をアクティブ化するには、parload -a コマンドを実行する。

```
[par0]# parolrad -a 2
Note: Cell Online activation operation has been initiated.
The sequence number is: 3
```

```
Note: Cell OL* operations, may take several minutes to complete.
Killing or otherwise aborting this parolrad command will not cancel the
Cell OL* operation. See the parolrad (1M) manpage for details.
Online operation in progress ....
Online operation has completed successfully.
```

これにより、cell2 はアクティブ化された。では、parstatus コマンドで状態を確認してみよう。

```
[par0]# parstatus -C
[Cell]
                        CPU     Memory                                  Use
                        OK/     (GB)                            Core    On
Hardware   Actual       Deconf/ OK/                             Cell    Next Par
Location   Usage        Max     Deconf    Connected To          Capable Boot Num
========== ============ ======= ========= ==================== ======= ==== ===
cab0,cell0 Active Core  8/0/8   8.0/0.0   cab0,bay0,chassis0    yes     yes  0
cab0,cell1 Active Core  8/0/8   8.0/0.0   cab0,bay0,chassis1    yes     yes  1
cab0,cell2 Active Float 8/0/8   8.0/0.0   cab8,bay0,chassis0    yes     yes  0
cab0,cell3 Active Base  8/0/8   8.0/0.0   cab8,bay0,chassis1    yes     yes  0
Notes: * = Cell has no interleaved memory.
```

ここで cell2 の項目を見ていただくとわかるとおり、「Actual Usage」は「Active」、そして「Par Num」は「0」となっており、同セルがパーティション 0 でアクティブ状態であることが確認できた。 ちなみに、ここで説明したようなコマンド操作の他にも、GUI ベースの管理ツールである Partition Manager による Dynamic nPar 管理も可能である。

図 2：Partition Manager による Dynamic nPar 管理の画面例

以上、今回は Integrity サーバーと HP-UX の新機能である Dynamic nPar にフォーカスをあて、そのメリットやメカニズムについて説明した。ハイエンドやミドルレンジのサーバーでは、サーバー全体のリブートやサービスの停止がとりわけ困難となる。そうした環境では、Dynamic nPar のオンラインでのメンテナンスやパーティション構成変更がきわめて重宝されるはずだ。またここで見てきたとおり、Dynamic nPar で新たに導入された概念やコマンド操作はそれほど難しくはなく、簡単に習得が可能である。

他のサーバー製品にはまねのできない Integrity サーバーの本領を発揮しようと思うなら、Dynamic nPar を使わない手はないだろう。

NEC が語る「HP-UX による高可用システム構築」

―目次―

NEC が語る「HP-UX による高可用システム構築」・前編

2005 年 11 月　テクニカルライター 村上隆一

メインフレームの最後の牙城であり、文字通りのミッションクリティカル・システムである銀行の勘定系。そして、1 日あたり約 8 億件、毎秒最大 7 万 5000 件におよぶ莫大なトラフィックを 24 時間 365 日無停止で処理する大手携帯電話会社のメールやインターネット・サービス。これら世界有数の高可用システムを、HP-UX で実現したのが NEC である。ここでは、NEC が培った HP-UX での知恵、そして HP-UX による高可用システム構築のノウハウを紹介する。

NEC が培った HP-UX での知恵

NEC が HP-UX サーバの OEM 供給を受け、PA-RISC プロセッサー搭載の「NX7000 シリーズ」として販売を開始したのが 1995 年のこと。この OEM 関係を皮切りに、NEC と HPE はシステム・インテグレーション領域での緊密なパートナーシップを深めてきた。2003 年にはインテル®Itanium®プロセッサ搭載サーバ「NX7700i シリーズ」の販売も開始している。今年でちょうど 10 周年を迎えるこのパートナーシップを通じて NEC が培ってきた HP-UX によるシステム構築の"知恵"は、同社のビジョンであった「オープン・ミッションクリティカル・システム」として開花し、いまや日本の IT 社会にしっかりと根を下ろしている。

なぜ NEC は、ミッションクリティカル・システムの基盤として第一に HP-UX を選択するのだろうか。NEC で高可用性基盤ソフトウェア製品を推進する第一コンピュータソフトウェア事業部の田中幸二氏と小松大麗氏は、次のように述べる。「プラットフォームの選択では、お客様がどの程度のミッションクリティカル度を求めているかが一番の判断材料になる。例えば、99.999%の可用性を実現するために、もっとも近道となる方法はどれか。HA(高可用性)が求められるシステムをいかに効率的に構築でき、信頼感 及び安心感の高いサポートサービスを提供できるか。これらの観点から見れば、今までの実績や蓄積されたノウハウから、HP-UX が最良の選択になる」。

金融系や通信系の難しさ

HP と NEC のパートナーシップによる成果の好例が、2003 年に国内の金融機関に導入された新勘定系システムだ。このシステムでは、NEC が 2000 年にオープン勘定系システム「BankingWeb21」として発表したソリューションが採用されている。BankingWeb21 は、SI を担当する NEC を中心に、HP-UX を提供する HP、Oracle データベースを提供するオラクル、そしてミドルウェア Tuxedo を提供する BEA システムズの 4 社の共同開発で開発し、金融機関へ導入している。メインフレームの最後の牙城であり文字通りのミッションクリティカル・システムである銀行の勘定系を、HP と NEC のパートナーシップで構築されたオープンシステムにより置き換えられることを実証した。

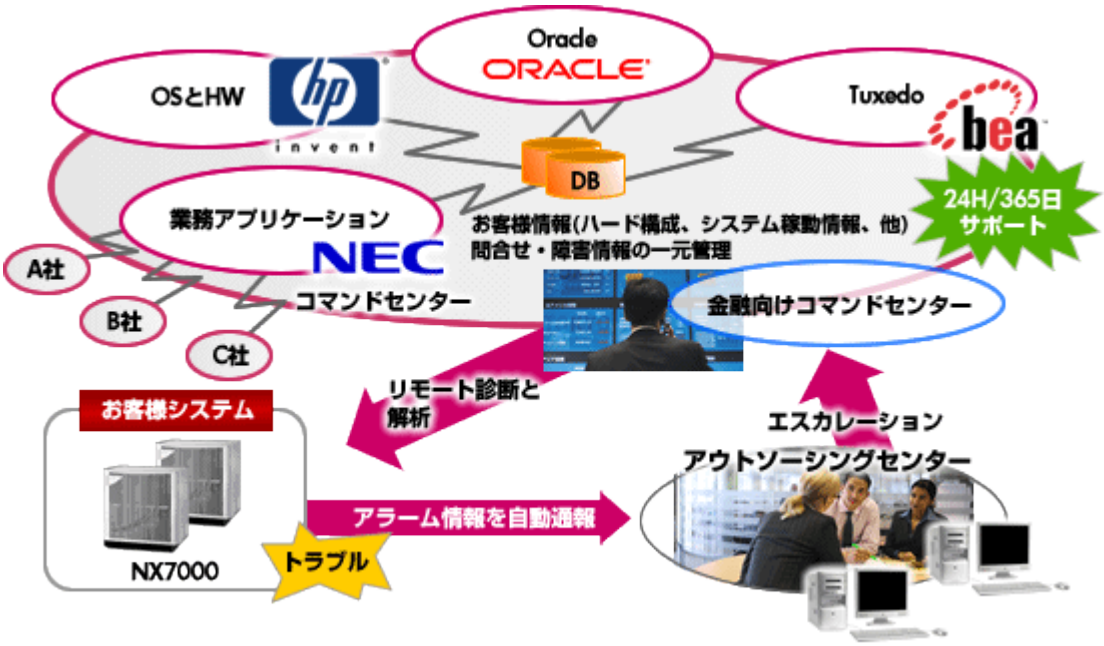


図 1：NEC、HP、オラクル、BEA システムズによる共同サポート体制

もうひとつの例として、大手携帯電話会社のメールやインターネット・サービスを支えるゲートウェイ・システムがある。このシステムは、HP-UX を搭載した数百台規模のサーバで構成されており、1 日あたり約 8 億件、毎秒最大 7 万 5000 件におよぶ莫大なトラフィックを 24 時間 365 日無停止で処理する、世界有数のミッションクリティカル・システムである。同システムにおいても、NEC をはじめ HP、オラクル、シスコシステムズなど各社のパートナーシップによるシステム・インテグレーションが実施された。

こうした金融系や通信系でのミッションクリティカル・システム構築には特有の難しさがつきまとう。「金融系や通信系では取り扱うデータがきわめて大量であり、社会インフラとして導入されているので、止めることができない。一般のお客様に比べて要求されるレベルは数段上になる。万が一システムに障害が起きた場合は、原因をすぐに解析し回復しなくてはならない」。

例えば上述の BankingWeb21 では、Oracle データベースによる高可用構成と、HP-UX のクラスタウェアである Serviceguard の組み合わせでデータベース・クラスタを構築し、これまで NEC が培ってきたシステム・インテグレーションのノウハウを投入することで、銀行の勘定系に求められる高度な可用性と柔軟性を実現可能になったという。「通常通りに構築されたクラスタでは、障害時の切り替えに分単位の時間がかかる。しかし銀行の勘定系では、システムが 1 分間止まっただけで ATM に『使用できません』という表示が出てしまう。それを防ぐために、BankingWeb21 では障害時の業務への影響を最小化する技術を実装した」と言っている。また上述の携帯電話会社のシステムでも、メールやインターネット・サービスを決して止まらないものとする厳格な可用性が求められたとのことだ。

NEC では、こうしたミッションクリティカル・システムの構築を通じて蓄積されたノウハウをシステム構築基盤のための高可用製品群として製品化している。これら製品群として提供されるに至った経緯や、ツールを利用した HP-UX による高可用システムの構築や運用のコツに関しては、後編でいくつかの例を紹介したい。

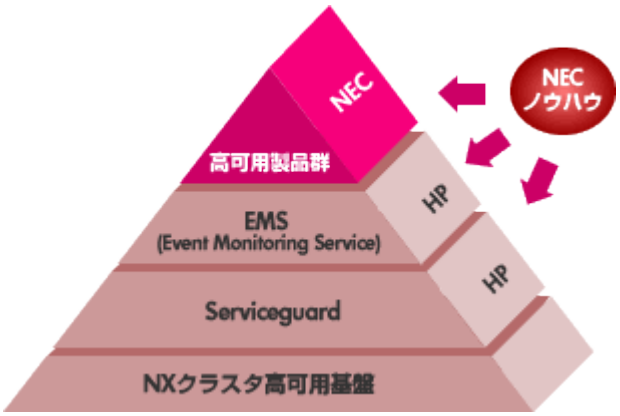


図 2：NEC が提供するシステム基盤ソフトウェアの位置づけ

無視できない障害発生のコスト

高可用システムの構築でもっとも重要視されるのが「障害発生のコスト」だという。「高可用システムでは、プラットフォームのコストよりも、障害発生時の 1 時間あたりのコストが問われる。障害を完全になくすことはできないので、いかに早く解析して復旧できるかが重要」とされる。高可用システムでは、サービスの停止を極小化するように設計されているが、それでも発生する障害に迅速に対応するために、NEC では HP との協力体制により ミッションクリティカル向けサポート (HA サポート)を提供している。HA サポートでは、24 時間 365 日の安定稼動が必須となるシステムに対し、システムの基盤となるハードウェアからソフトウェアまでの HA(高可用性)と、業務のライフサイクル(分析・企画から運用フェーズまで)を通じてミッションクリティカル性を高めるサポートを行う。特に運用フェーズでは、プロアクティブ・サービス(システムをダウンさせない予防保守)とリアクティブ・サポート(ダウン発生時の速やかな復旧)を組み合わせることで、システムの安定稼働を実現する。

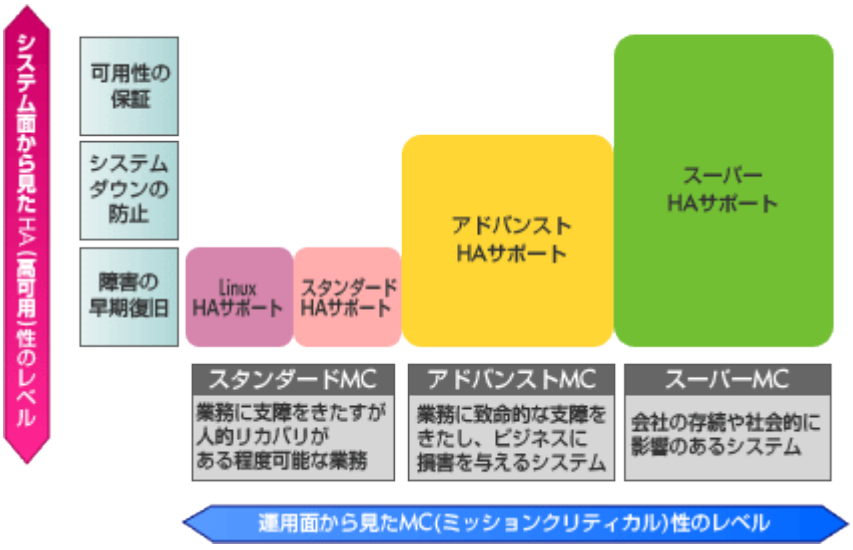


図 3：NEC が提供するミッションクリティカル(MC)向け HA サポート

障害発生のコストがプラットフォームのコストを上回るミッションクリティカル・システムでは、こうした高度なサポート体制が提供されているプラットフォームの方が総合的には“安くつく”という判断だ。

協調から生まれた業界でもユニークなソリューション、“ミッションクリティカル Java”

1995 年に新しい言語として登場した Java は、開発納期短縮、高い機能性というシステム開発における新たな要望に応える言語として利用されている。現在では、企業の基幹系システムのアプリケーションを Java で構築することは標準的となっている。特に NEC では 2000 年頃から、製造業や金融分野で Java による基幹システムの構築事例が急増しており、これまで以上にミッシ

ョンクリティカルなビジネスを支える重要な HP-UX の一つの機能として浸透しつつある。このような Java を取り巻く環境の変化から、ミッションクリティカル・システムとして利用する場合の高い可用性及び信頼性、障害時における早期解決に対応する体制などが、システムの導入、運用の現場では Java ならではの課題として浮き彫りになってきていた。

現場から NEC のサポート部隊に寄せられた「Java の課題」を整理すると、主に以下の 3 つに分類できる。

- 短いサポート期間 及び頻繁なバージョン更新(新規バージョンへの移行リスク及び高コスト)
- 原因究明に時間がかかる(顧客のビジネスへ与えるリスク増)
- 解決策の遅延(修正は次期アップデート版に依存)

この課題に対して、NEC は HP へ HP-UX 上の Java VM(仮想マシン)のサポータビリティ強化策を提案した。そして、両社の密接な協力関係により、市場における要求に応えるべく、業界で始めて体系化されたミッションクリティカル向けソリューションとして誕生したのが MC Java サポートソリューションである。

MC Java サポートソリューションでは、HP-UX の標準サポートに加えて、JVM の長期サポートが用意された。具体的には、JVM のメジャーバージョンアップ後、1～1.5 年の評価期間を経て高信頼性が確認されたマイナーバージョンを、HPE が「MC サポート対象版」として認定する。標準サポートでは同バージョンに対するサポート期間は残り 3～3.5 年となるが、MC Java サポートソリューションではさらに 3 年の期間延長を実施する。つまり合計で 6～6.5 年の長期サポートを受けられるわけだ。これにより、長期運用を前提とした IT システムでも JVM のサポート計画を立てやすくなる。

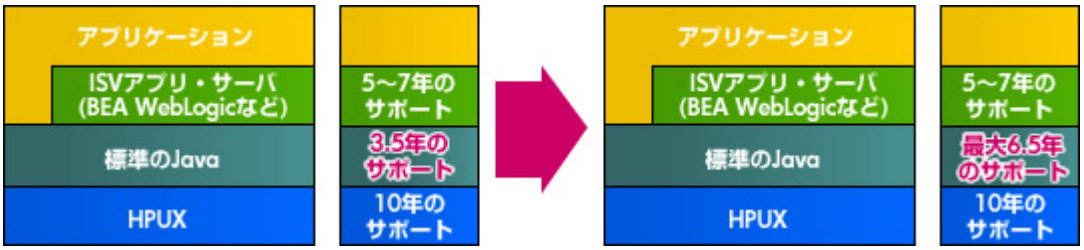


図 4：MC Java によるエンタープライズサポートライフの提供

MC Java サポートソリューションのもう一つの特徴は、JVM における個々の障害に対してミッションクリティカル・サポートとして短期間にてパッチ提供が可能になる点だ。また、提供されるパッチは、主要な ISV ソフトウェアとの動作検証を経て、ISV 各社の承認を HPE が得てから提供される。こうした信頼性の高いパッチの供給により大がかりな JVM のバージョンアップは不要となり、システムに影響を与えない範囲で 且つタイムリーな対応が可能となる。MC Java サポートソリューションではさらに、Java アプリケーションの障害発生時の解析や切り分け作業を支援する、サポータビリティ・ツールキット(SKIT)を提供する。

このように、NEC と HP による密接なコラボレーションを通じて、「Java の課題」を解決する業界でも類を見ないユニークなソリューションが生まれたのである。つづく後編では、NEC が語る HP-UX 高可用システム構築ノウハウを紹介したい。

NEC が語る「HP-UX による高可用システム構築」・後編

2005 年 11 月　テクニカルライター 村上隆一

NEC の高可用製品群は、HP-UX 環境に向けた基盤ソフトウェアであり、「ミッションクリティカル環境におけるシステム・インテグレーションのノウハウをツールとして提供することで、業務継続性を更に高める可用性を実現するとともに、同様の機能を毎回実装する労力を軽減する」という。ここでは、これら製品群として提供されるに至った経緯や、ツールを利用した HP-UX 大規模高可用システムの構築と運用のコツについて紹介したい。

大規模高可用システムの構築と運用のコツ

高可用システムの構築では、大規模システムならではのニーズに応える必要も生じる。「サーバ台数が 100 台規模になると、ソフトウェアの配布だけでも一大事になる。そこで、経験者が 1 台目だけを手作業でインストールし、そのテンプレートをベースに差分だけを変更して他のサーバに適用することで、高可用システムの構築を自動化するツールを作成した」という。この手法により、HP-UX の標準インストールツールである Ignite-UX を使いこなした上で、自社のノウハウも加えることでネットワークやストレージの設定自動化も実現している。

また、高可用システムの運用で問題となるのは、システムやアプリケーションからエラー・メッセージが出力された場合、それがどのような意味を持つのか理解し、どのような対処をすべきか判断することの難しさだ。対処方法がドキュメント化されているとはいえ、現場の運用担当者で対応できることには限界がある。そのため NEC では、「OS、データベース、ミドルウェア、アプリケーションなどのエラー・メッセージについて、いままでの経験で蓄積された運用ノウハウをデータベース化」した。いわば「システム運用の虎の巻」である。

Oracle の「ストール」に対処する

Oracle データベースの運用では、データベースの「ストール」、すなわち「Oracle インスタンスはダウンしていないが、パフォーマンスが極端に低下している状態」にどう対処するかがポイントになる。「例えば、ディスク障害などの理由により大量の SQL 処理が滞留し、タイムアウトするような状態がストールだ。こうしたトラブルは運用担当者が気づくのが難しく、検出にかなり時間がかかる。エンドユーザからの連絡で初めて明らかになることも多い。そのため、NEC 社内の Oracle のエキスパートと連携して、Oracle のストールをもっとも効率的に検出する方法を検討し、製品化(Application Monitor)した。この製品と Serviceguard の連携によって、障害検出、フェイルオーバー、障害情報収集を自動的に行うことができ、従来のストール状態での対処時間を大幅に削減することができる」という。

また、Oracle が稼働していても、クライアントとなるアプリケーション・サーバからアクセスできない状態では意味がない。そのため NEC では、BEA WebLogic などが動作するアプリケーション・サーバ上から Oracle に接続できることを常時監視する技術も上述の製品と合わせて提供している。

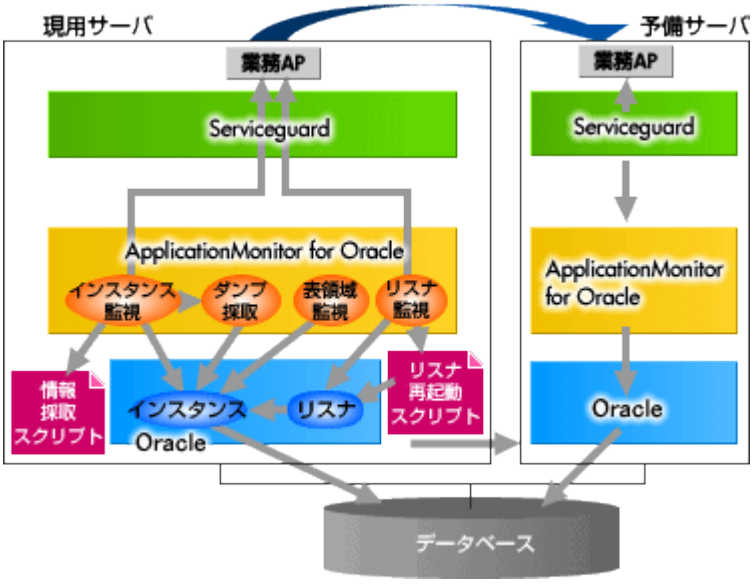


図 1：Oracle のストール監視のメカニズム

高可用システムにおける Oracle データベース構築のもう一つのポイントは、キャッシュ領域として大量のメモリを割り当てることだという。ここで生きてくるのが、HP-UX の 64 ビットのアドレス空間である。「Oracle が稼働するサーバには CPU をたくさん搭載するが、実際には CPU 負荷はあまり高くない。むしろ、負荷が集中する I/O 処理をなるべく減らすため、メモリをキャッシュとして利用する。そのため、データベース・サーバの大半は 64 ビットで構築する」。ちなみにアプリケーション・サーバでも、32 ビットから 64 ビットへの移行が進みつつあるとのことだ。

高可用システムのリソース監視

こうして慎重を期して構築される高可用システムだが、実際にシステムの障害によりクラスタのフェイルオーバーが発生するケースも無視できない。そこで、金融系や通信系では、現実にはほとんど発生しない二重障害をも想定したソリューションが求められる。例えば、プライマリ・ノード(現用系)とセカンダリ・ノード(待機系)で別々のアプリケーションを運用するアクティブ/アクティブ構成の HA クラスタの場合を考えてみよう。このとき、たまたま待機系が高負荷状態のときに現用系でトラブルが起きると、リソース不足によりフェイルオーバーが失敗することもある。よって「待機系の CPU やファイルの不足状態を検出して、今の状態ではフェイルオーバーできないことを運用担当者に通知する仕組みが不可欠」となり、NEC では高可用製品群のツールとして提供している。こうしたレアケースへの対策も事前に検討しているかどうかが、高可用システム導入時のポイントと言えるだろう。

高可用システムのリソース監視でもうひとつ問題となるポイントは、ディスクやネットワーク、OS などのリソース監視対象が数 100 か所に及ぶことだ。これだけの数になると個々のリソースをばらばらに監視するのは非効率であり、複数のリソースをグループ化して監視したいというニーズが出てくる。高可用製品群には、そうしたユーザ企業の声を受けて NEC が開発したツールがある。

高可用製品群の ResourceSaver は、Event Monitoring Service(EMS)のリソース監視と Serviceguard のパッケージ連動機能、監視対象の柔軟な組み合わせや監視対象の動的変更/再開等の運用コマンドを提供するソフトウェアだ。既存の Serviceguard や EMS と連携することで、より高度な可用性を実現できる。

例えば、監視対象の抽象化機能では、個々の監視リソースの状態を特定の条件式を基に組み合わせ、その結果得られる論理集合を仮想的なリソースとして定義し、複数の監視リソースを組み合わせた論理的な判定を行う。また「障害通知に対するリトライ機能」では、監視対象リソースの障害を検出した時、一定時間リトライを行える。装置の間欠故障や一時的なダウン状態を認識し、

不必要な待機系への切り替えを行わず、正しい処理を実行させる。さらに「運用・構成管理機能」により、Serviceguard のパッケージ切り替えを行うことなく、リソース監視に対する各種要求や設定を動的に行うことができ、運用/保守性が向上する。

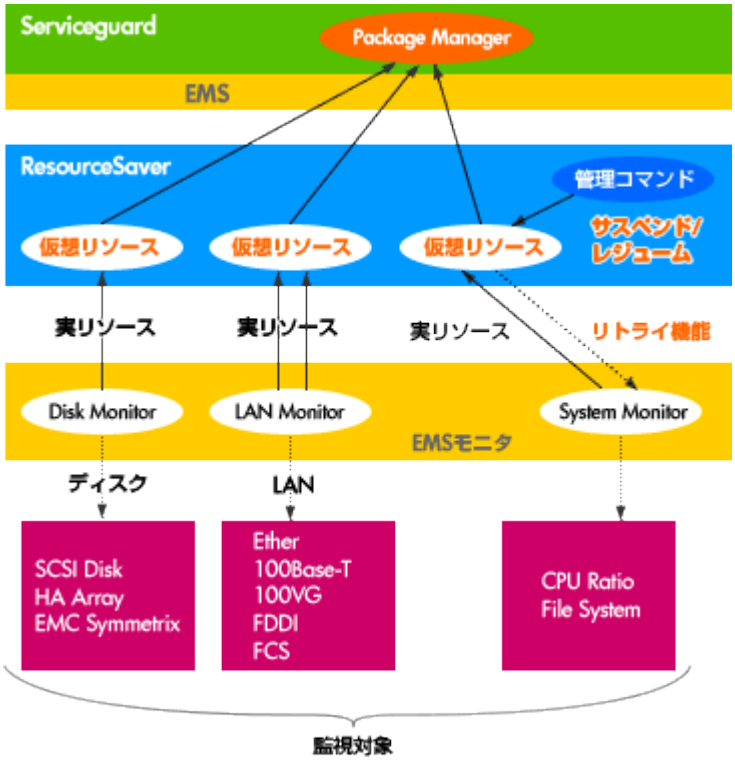


図 2：ResourceSaver による仮想リソースの監視

国内でも導入が進む仮想化技術

さらに NEC では HPE の仮想化技術を活用したシステム・インテグレーションも積極的に実施している。HP-UX の Virtual Partitions (vPars)によりビジネスニーズに応じた柔軟な構成変更を実現しつつ、同じく nPartitions(nPars)により大規模システムに求められる高可用性を実現するというアプローチだ。「サーバ・コンソリデーション(統合)の優位性に興味を持たれる方が増えているが、PA-RISC 版の vPars は既に利用されている。インテル®Itanium®プロセッサ版の vPars も出荷開始した。例えば Superdome などの大規模サーバを導入したお客様において、サーバ・リソースを分割して利用したい場合に用いられている」と説明する。もっとも、vPars の動的なパーティション変更は行わず、手作業で確実に実施するケースが多いという。「HA システムでは、障害の検出などは自動化されているが、意図的になにかを実施する場合は自動よりも手動が好まれる。大きな作業は手動でやらなくてはならない。そのため、vPars のパーティション変更そのものは運用担当者が手動で実施するケースが大半だ。例えば業務の増加に対応して、既存のサーバでそれを収容したい場合に利用されている」。

また vPars を活用する次のような例もあるという。「業務内容は同じだが、障害時の影響をなるべく最小限に抑えたいというニーズがあった。そこで、都道府県ごとに代理店を分割し、それぞれを異なる vPars 上のシステムで収容している。すべてをひとつのパーティションに一本化すれば管理は容易になるが、そうではなくパーティションを複数に分割しておくことで全国規模の障害を避けられる」という判断だ。

今回は、HP-UX による高可用システム構築の実際について、話を伺うことができた。同社が示した「オープンシステムで高可用システムはこう作る」という方法論の手応えを感じ取っていただけたはずだ。

HP-UX

www.hpe.com/jp/hpux

Hewlett Packard Enterprise