サーバ・ソリューション・マガジン

月刊サーバセレクト

SERVERS SELECT

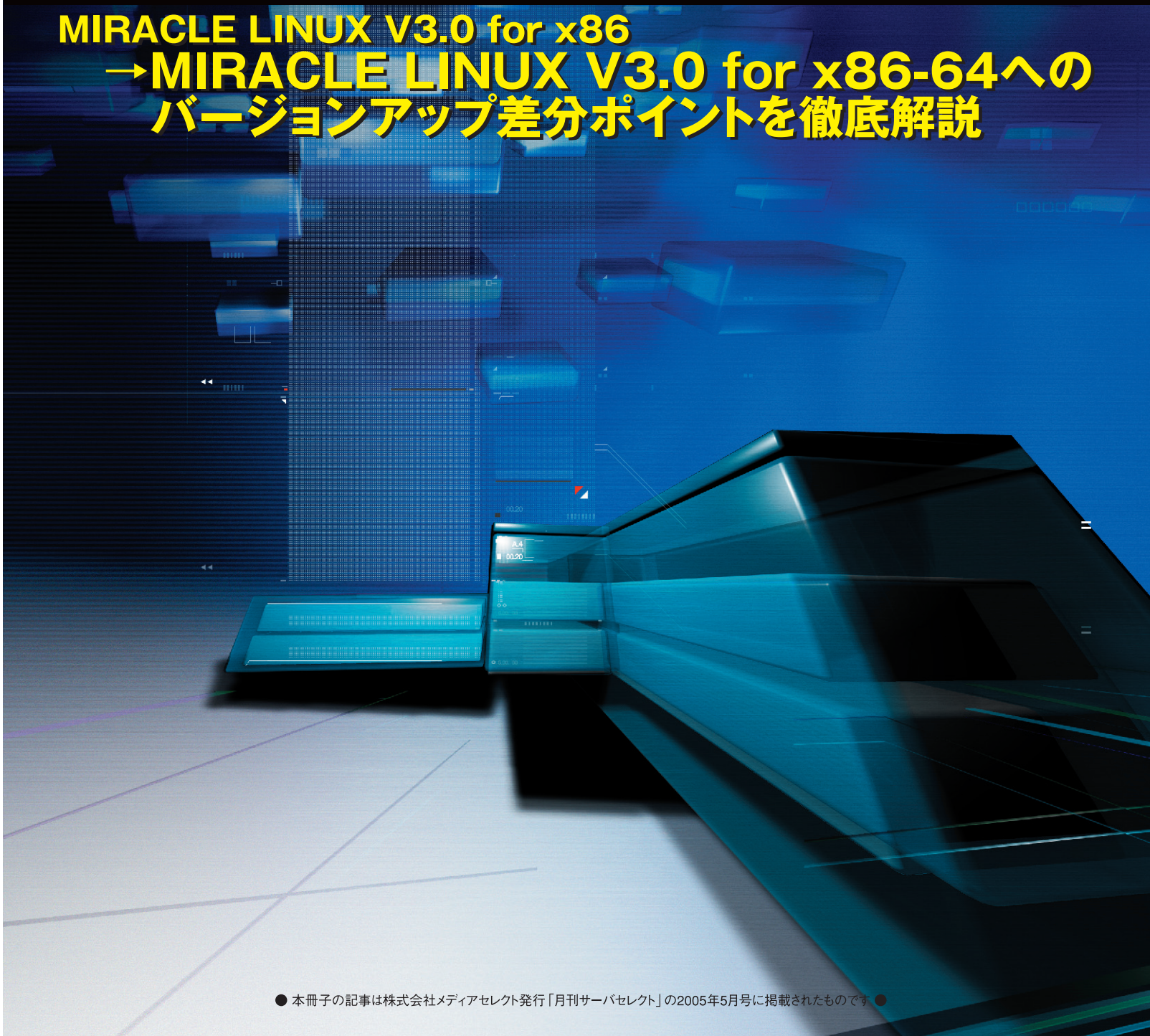
創刊2号

2005 May. 5

マルチプラットフォームのサーバ実用情報を満載!

●最新ソフトウェア［今月の差分情報］

MIRACLE LINUX V3.0 for x86 →MIRACLE LINUX V3.0 for x86-64へのバージョンアップ差分ポイントを徹底解説



● 本冊子の記事は株式会社メディアセレクト発行「月刊サーバセレクト」の2005年5月号に掲載されたものです ●

ミラクル・リナックス株式会社 〒105-0021東京都港区東新橋二丁目4番1号 サンマリーノ汐留 5階 TEL : 03-5404-5050 FAX : 03-5404-5051

最新ソフトウェア 今月の差分情報

MIRACLE LINUX V3.0 for x86-64

MIRACLE LINUX V3.0 for x86

MIRACLE LINUX V3.0 for x86 → MIRACLE LINUX V3.0 for x86-64 へのバージョンアップ差分ポイントを徹底解説

管理者の知恵“最新ソフトは差分を押えろ！”

2005年2月28日、「MIRACLE LINUX V3.0-Asianux Inside for x86-64（以下V3.0 for x86-64）」がリリースされた。同OSは既存のx86アーキテクチャとの互換性を確保しながら64bitへの対応も万全であり、Linuxベースでシステムを構築している管理者であれば、まずは押えておくべきディストリビューションである。メモリ空間のスケーラビリティ向上や64bit化によるパフォーマンスアップなど、導入効果を期待できる点も多い。今回はそのV3.0 for x86-64のアドバンテージを、具体的なベンチマーク結果も絡めつつ検証する。

文●中山一弘／エースラッシュ

64bit化で得られたMIRACLE LINUX V3.0 for x86-64のアドバンテージ

既存のシステムを生かしつつ、64bit環境へ移行することが可能となる**32bit互換モード**を備えたx86-64対応LinuxがV3.0 for x86-64だ。64bitCPUへの対応、**メモリ領域の拡大**によってハードウェアの持つポテンシャルを存分に引き出し、従来のアプリケーションもそのまま使えるのが最大の特徴である。当然、**64bitネイティブ対応**のアプリケーションを実行すればこれまでのOSとは比較にならないほどのパフォーマンス向上が期待できる。既存のシステムを運用しつつ、将来的にシステムをフル64bit化することを目標としているならば、管理者にとって、まず第一に検討する価値のあるOSである。

製品名	MIRACLE LINUX V3.0-Asianux Inside for x86-64	
動作環境	CPU	64ビット インテル Xeon プロセッサ AMD Opteron プロセッサ AMD Athlon 64 プロセッサ
	メモリ	128MB以上が必須、256MB以上を推奨
	ハードディスク	1GB（最小構成）以上の空き容量が必須、4GB以上を推奨

Point 1 既存IT資産の有効活用	Point 2 パフォーマンスの向上	
32bit互換モード	メモリ領域の拡大	64bitネイティブ対応
差分1 32bitプログラムも利用可能	差分3 スタビリティの向上と システムオーバーヘッドの減少	差分5 プロセスあたりの スレッド数増加
差分2 32bitライブラリの拡充	差分4 メモリ領域の拡大による スケーラビリティの向上	差分6 Disk I/Oおよびワークロード 実行性能を確保
		差分7 Java上での性能が向上

Point 1

スムーズな移行を実現する「既存 IT 資産の有効活用」

大容量化する物理メモリ、さらに 64bit 対応 CPU の登場と、進化を続けるハードウェア。これらのスペックをフルに発揮させるには、x86-64 対応の OS を導入することが必須となる。しかし既存業務で使われているアプリケーションは、32bit のものがほとんどであろう。業種によっては業務アプリケーションをすんなり 64bit 環境へ移行できる場合もあるだろうが、通常は部署、あるいは社内全体で一気に移行させることなど不可能に近い。V3.0 for x86-64 では、32bit 互換モードを用意することで、この問題を解決している。

差分 1 32bit プログラムも利用可能

既存 IT 資産の有効活用		パフォーマンスの向上
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

今回リリースされた V3.0 for x86-64 のメリットは 64bit に対応したというだけではない。既存の資産である 32bit プログラムを生かしつつ導入できるという特徴がある。

この OS には 32bit 互換モードが用意されており、64bit とは別に用意された 32bit ライブラリを利用することで、32bit アプリケーションもこれまでどおり運用することが可能となっている。アプリケーションライブラリも「/lib、/usr/lib（※ 64bit アプリケーションは /lib64、/usr/lib64）」といったディレクトリに格納されるので、オペレーションも従来どおり行うことができる（図 1）。既存のシステムと平行運用を行いながら 64bit へシフトしたい場合や、将来的に完全な 64bit への移行を考えている場合にも柔軟に対応することができる。

▶ライブラリの配置
　▶64bitアプリケーション
　　▶/lib64、/usr/lib64ディレクトリの64bitライブラリを利用
　▶32bitアプリケーション
　　▶/lib、/usr/libディレクトリの32bitライブラリを利用
▶RPMパッケージ
　▶64bitライブラリ、32bitライブラリのパッケージを同梱

```
# rpm  Vq  Vqf  e%{name}.%{arch}\n f glibc
glibc.x86_64
glibc.i686
```

図 1　32bit 互換モードの実装

同じ 64bit マイクロプロセッサのアーキテクチャとして存在している「IA-64」との違いは、32bit 環境での動作にある。IA-64 では 32bit 命令を 64bit 命令に変換する際のシステムにかかるオーバーヘッドが大きいため、既存の 32bit アプリケーションを動作させた場合、32bit プロセッサ上で動作させるよりもパフォーマンスが落ちる傾向にある。

V3.0 for x86-64 の 32bit 互換モードで 32bit アプリケーションを運用する場合、システムコールの実行は、32bit 互換モード用に設けられたシステムコールインターフェイスを経由して、カーネルに届けられる。

もちろん 64bit アプリケーションは、通常のシステムコールインターフェイスを利用する。また RPM パッケージは、64bit、32bit の両方が必要になる。64bit 用のパッケージは当然だが、32bit 用のライブラリも RPM パッケージとして同梱されている。この結果、32bit アプリケーションを操作するユーザーは、V3.0 for x86-64 が 64bit 対応 OS であることを特に意識することなく 32bit アプリケーションを実行することができる（図 2）。

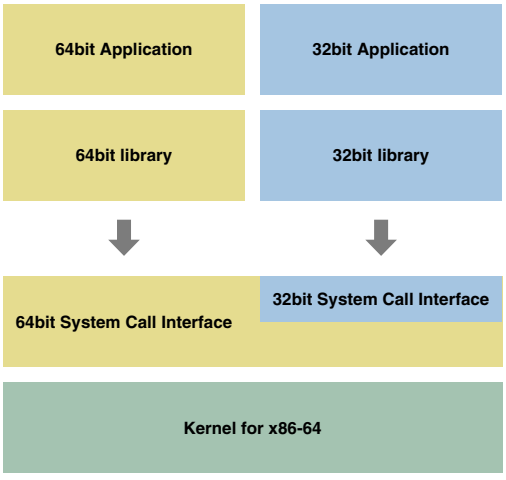


図 2　32bit 互換モード概念図

差分2

32bit ライブラリの拡充

既存 IT 資産の有効活用	パフォーマンスの向上	
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

V3.0 for x86-64 の互換モードは優れたものである。しかしそれでも、すべての 32bit アプリケーションが動作するわけではない。32bit アプリケーションに対して必要な 32bit ライブラリが不足している場合、当然動作させることができない。この場合には、必要とされる 32bit ライブラリを別途導入する必要がある。また 64bit アプリケーションとして提供されている Apache や Perl などに対して 32bit でコンパイルされたモジュールは動作させることができない。さらに 32bit のカーネルモジュールが必要なアプリケーション (一部のセキュリティソフト) なども動作させることができない。

以上の点をふまえ、業務で使う必要のある 32bit アプリケーションについては、事前の調査が必要だろう (図 3)。

図 3　32bit アプリケーション資産を使う場合、以上の 3 点については事前に調べておきたい

32bit ライブラリも提供する 64bit 対応 Linux としては、「RedHat Enterprise Linux 3」などがあるが、V3.0 for x86-64 では、RedHat に収録されているものに加え、さらにいくつかの 32bit ライブラリを追加している (表 1)。

ORBit
audiofile
beecrypt
bzip2-libs
esound
gnome-libs
gtk+
imlib
libpng10
libungif
libxml2
net-snmp
openssl
rpm-libs
tcp_wrappers

表 1　V3.0 for x86-64 で追加された追加 32bit ライブラリ例

これらのパッケージとしては、bzip2-libs や libxml2 など、32bit アプリケーションの動作検証で必要性が明らかになったものを追加しており、全部で 82 個の 32bit パッケージが同梱されている (表 2)。

主要ソフトウェア			
パッケージ名称	バージョン	パッケージ名称	バージョン
kernel	2.4.21	squid	2.5.STABLE3
glibc	2.3.2	Apache	2.0.46
gcc	3.2.3	PHP	4.3.3
XFree86	4.3.0	MySQL	3.23.58
rpm	4.2.3	PostgreSQL	7.4.6
KDE	3.1	Samba	3.0.10
openssh	3.6.1p2	ntp	4.1.2
bind	9.2.4	net-snmp	5.1
dhcp	3.0pl2	iptable	1.2.8
postfix	2.0.16	perl	5.8.0
sendmail	8.12.11	ruby	1.6.8
proftpd	1.2.10	oranavi	10.1

表 2　V3.0 for x86-64 に収録される主要ソフトウェア

Point 2

64bit ネイティブ対応による「パフォーマンスの向上」

x86 と x86-64 を比べた場合、大きく異なるのはメモリ領域の拡大と 64bit ネイティブ対応という部分になる。どちらの場合も x86-64 ではスケーラビリティ、スタビリティの向上が期待でき、フル 64bit 環境を構築できたのならば、これまでとは比較にならないほど高いパフォーマンスが発揮される。ここでは V3.0 for x86-64 を使用し、従来のアプリケーションでの動作や比較テスト結果をもとに、x86-64 対応 OS としてのパフォーマンスを検証する。

差分 3 スタビリティの向上とシステムオーバーヘッドの減少

既存 IT 資産の有効活用	パフォーマンスの向上	
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

従来の 32bit 環境で使える仮想アドレス空間は 4GB が最大値となっていた。しかしビット幅が倍の 64bit になることで、扱えるようになるメモリ領域は 16 エクサ（10^18）という巨大な物となった。

32bit 環境下における最大 4GB のメモリ空間では、カーネル空間に 1GB が割り当てられ、ユーザー空間には 3GB が割り当てられる（図 4）。このことにより、1 プロセスあたりの利用可能メモリは最大 3GB ということになり、スケーラビリティの限界を招いていた。同様に搭載メモリの拡大により、カーネル内のオブジェクトが増加してしまうことでカーネル空間が圧迫され、スタビリティの低下が問題になっていた。

なお、「4G/4G パッチ」と呼ばれる仮想アドレス空間の割り当てをそれぞれ 4GB に変更する機能が kernel-hugemem として提供されているが、この機能を採用するとオーバーヘッドによる性能低下が大きく、現実的ではない。

サーバに搭載される物理メモリの調達コストが安くなった現在、その搭載メモリサイズが仮想メモリサイズを超えることも多くなっている。従来の OS ではその能力を使い切れないという問題もあった。大容量化する物理メモリを有効に使うためにも、OS の 64bit 化は必然的な流れである。

V3.0 for x86-64 では、仮想アドレス空間についてはカーネルコードや vmalloc 領域、direct mapping 領域などに対し、理論上、それぞれ 512GB のリソースを与えることができる。さらにユーザー空間にも同様に 512GB 以上を割くことができるので（図 5）、かなり余裕のあるメモリ空間となる。

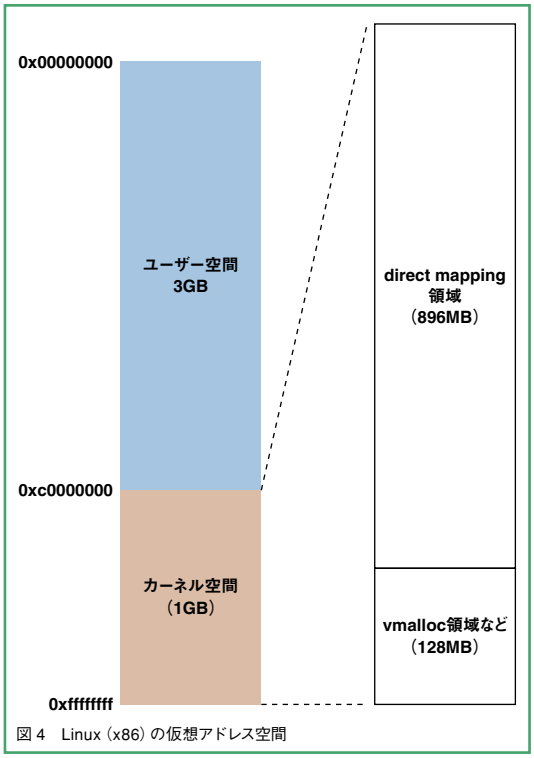


図 4　Linux（x86）の仮想アドレス空間

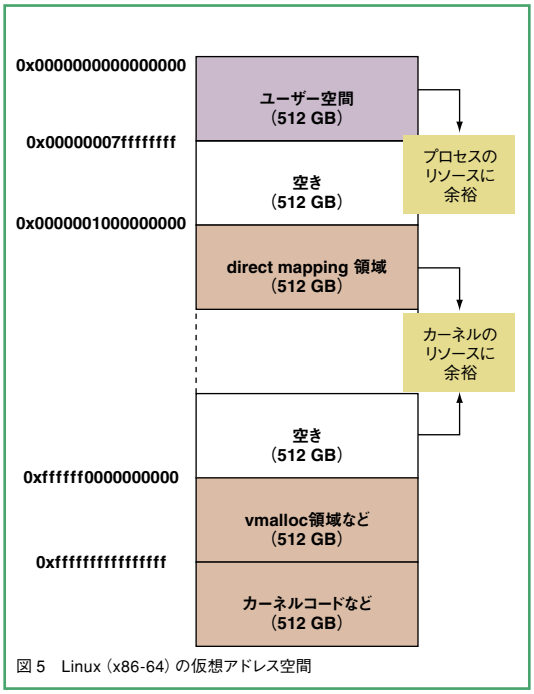


図 5　Linux（x86-64）の仮想アドレス空間

差分4 メモリ領域の拡大によるスケーラビリティの向上

既存 IT 資産の有効活用	パフォーマンスの向上	
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

ここでは、「V3.0 for x86-64 & Oracle Database for Linux x86-64」と「V3.0 for x86 & Oracle Database 10g for Linux」を使用し、メモリ領域の拡大により大規模 SGA（Software Global Area）の実装方法がどのように変化するのか検証を行う。

x86 環境で利用できる仮想メモリ空間は 4GB であることは先に述べたとおりだ。SGA はユーザー空間にある共有メモリ領域に作成されるが、テキストセグメントやライブラリマッピング、ピープ領域などに一定サイズが取られてしまうので、SGA に割り当てられる最大容量はデフォルトで約 1.7GB が上限となっている（図 6）。

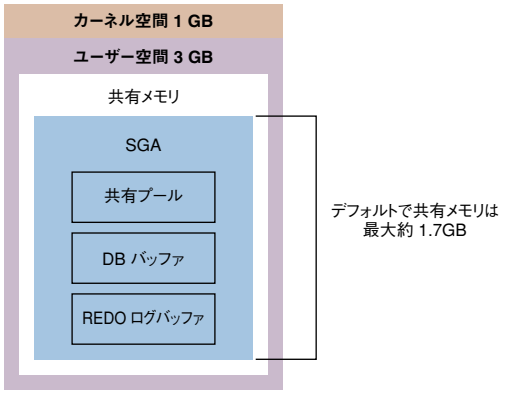


図 6　Oracle Database（x86）運用時のデフォルトでの共有メモリサイズは 1.7GB だ

SGA をさらに大きく拡張する場合には、VLM（Very Large Memory）機能を使うことになる（図 7）。これにより SGA 内のデータベースバッファキャッシュのみをメモリファイルシステムに出すことで大容量を確保し、物理メモリをデータベースバッファキャッシュで利用することが可能となる。

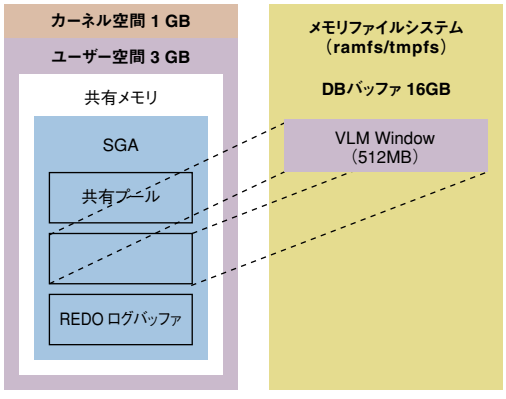


図 7　VLM を利用するとデータベースバッファは 16GB になる

このデータベースバッファキャッシュは単なるファイルとして扱われるため、共有メモリの「1.7GB」という制約を受けることはない。しかし、メモリファイルシステム上のデータベースバッファキャッシュにアクセスするためには、データベースバッファキャッシュシステム上のデータを共有メモリ上にマッピングする必要がある。VLM がマッピングする領域を VLM ウィンドウとして用意しなければならず、この通常の SGA にない作業が、システムのオーバーヘッドとなってしまうのだ（図 8）。特にデータベースバッファキャッシュ上にランダムアクセスする索引検索などの作業では、このオーバーヘッドが顕著に現れ、パフォーマンスの劣化が起こることになる。

これにより、連続データにアクセスする全表操作はパフォーマンスの向上が見られるものの、ランダムアクセスではパフォーマンスが落ちるという結果になってしまうのだ。また、SGA のデータベースバッファキャッシュ以外に拡張領域を作ることができない、という問題も大きい。

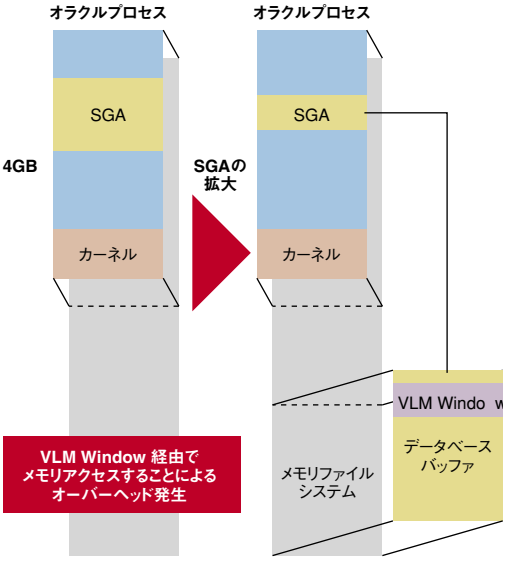


図 8　VLM Window 経由でメモリアクセスを行うとオーバーヘッドが発生。x86 環境では、これを避けるために SGA を拡大する必要がある

しかし、x86-64 対応 OS である V3.0 for x86-64 の場合、仮想メモリ空間が 4GB から 1TB へと拡大しているため、より大規模な物理メモリを SGA に割り当てることができる。ユーザー空間が 512GB となり、それに伴う共有メモリ領域も約 340GB まで確保することが可能なため、SGA の領域としては理論上、最大で 300GB 程度までは確保できる。デフォルト状態でも数 10GB 単位で簡単に確保できること

になるので、現在のハードウェア事情を考えると、ほぼ、上限はないと言ってもいいだろう（図 9）。

先ほどの x86 環境でのメモリサイズ制約から解放されるとともに、オーバーヘッドのない作業が可能になり、パフォーマンスが著しく向上する。Oracle 側、OS 側の両方において VLM 用の特殊な設定も不要となり、SGA のデータベースバッファキャッシュ以外の領域拡張も可能となる。

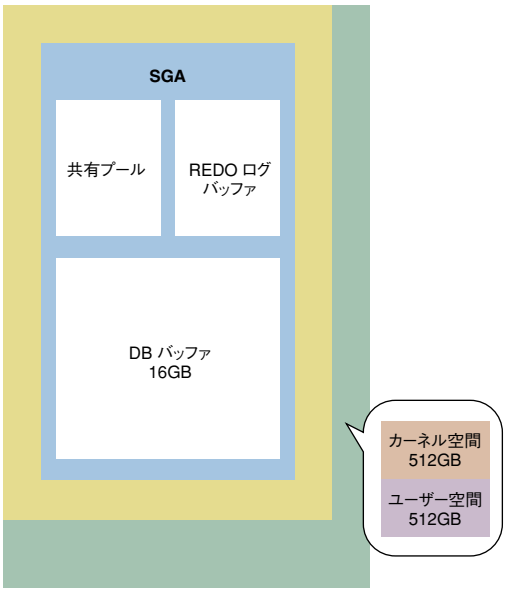


図 9　Oracle Database（x86-64）ではユーザー空間が 512GB になるので SGA サイズの上限はなくなる

表 3 は、行数が 500 万行の表を 5 つ用意し、データベースバッファキャッシュのサイズを拡大させながら索引検索の応答時間（全表検索のパフォーマンス）を計測したものだ。なお、「x86 w/tmpfs」「x86 w/ramfs」は、32 ビット環境でのそれぞれのメモリファイルシステムで VLM を使用している。

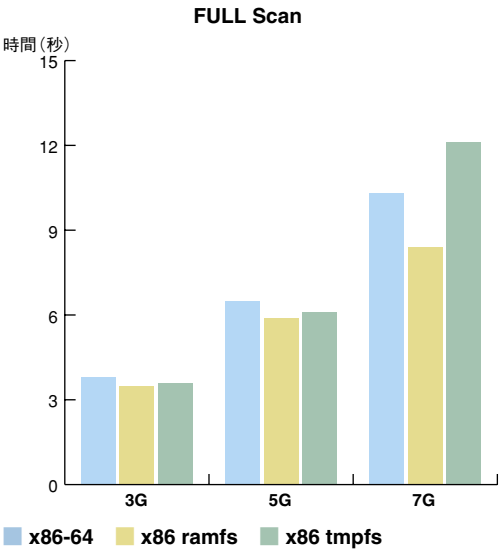


表 3　全表検索の応答時間

x86 では DB バッファサイズが 3GB のときに VLM のオーバーヘッドが影響し、若干のパフォーマンス低下を招いているが、それ以外はヒット率の上昇（表 4）と共に応答時間が短縮している。

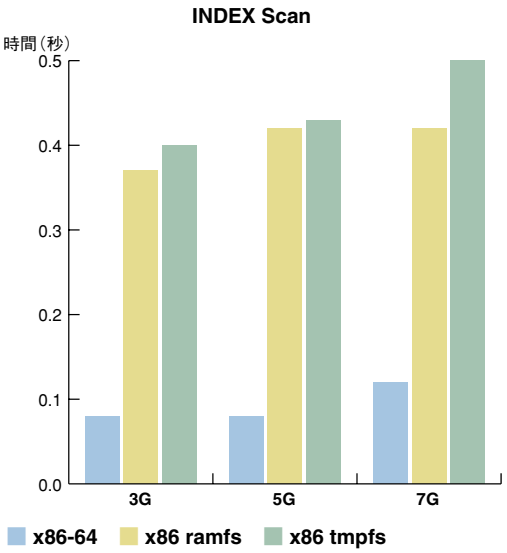


表 4　DB バッファキャッシュのヒット率

表 5 に示すのは、全表検索と同じ環境でデータベースバッファキャッシュのサイズを拡大させながら索引検索の応答時間を計測し、x86（データベースバッファキャッシュ 1GB）の応答時間を 100% とした場合の相対処理性能をグラフにしたものだ。x86-64 ではデータベースバッファキャッシュのサイズ拡大とともに応答時間が短縮されていく様子がよく分かる結果となった。

それに対し、x86 w/VLM では VLM を使わないそのほかの構成と比べ、2 ～ 3 倍もパフォーマンスが低下してしまっている。先にも述べたが、ランダムアクセスによって VLM ウィンドウの再マッピングが頻繁に発生したためである。

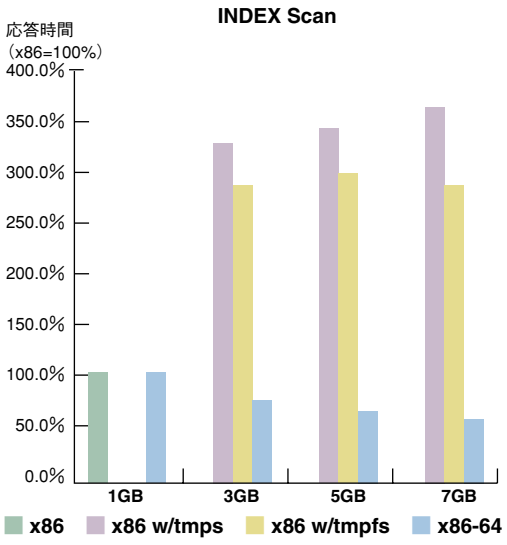


表 5　索引検索の応答時間

これらのことから分かるのは、大容量メモリを搭載したサーバで「V3.0 for x86-64 & Oracle Database for Linux x86-64」というシステムを運用する場合、x86-64 環境の方が圧倒的に高いパフォーマンスを弾き出すということだ。厳密に言えば、ハードとシステムが持つ性能をフルに引き出すことが可能になっているということであり、今までにないノンストレス環境を構築することができることを、数値が表している。また VLM を使わずにデフォルト状態で巨大な SGA 空間が持てるという操作性の良さも特筆しておかなければならない。

差分5 プロセスあたりのスレッド数増加

既存 IT 資産の有効活用	パフォーマンスの向上	
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

ここまでデータベース運用環境における V3.0 for x86-64 のパフォーマンスを見てきたが、次に「プロセスにおける最大スレッド数」を検証する（表 6）。グラフの横軸は搭載メモリ、縦軸はスレッド数となっている。メモリがフルに活用される 64bit on x86-64 に関しては、メモリが増えるにしたがい作成されるスレッド数も伸びていく。一方、32bit では x86、x86-64 とも差はなく、4GB を上限としてスレッド数も頭打ちとなってしまう。

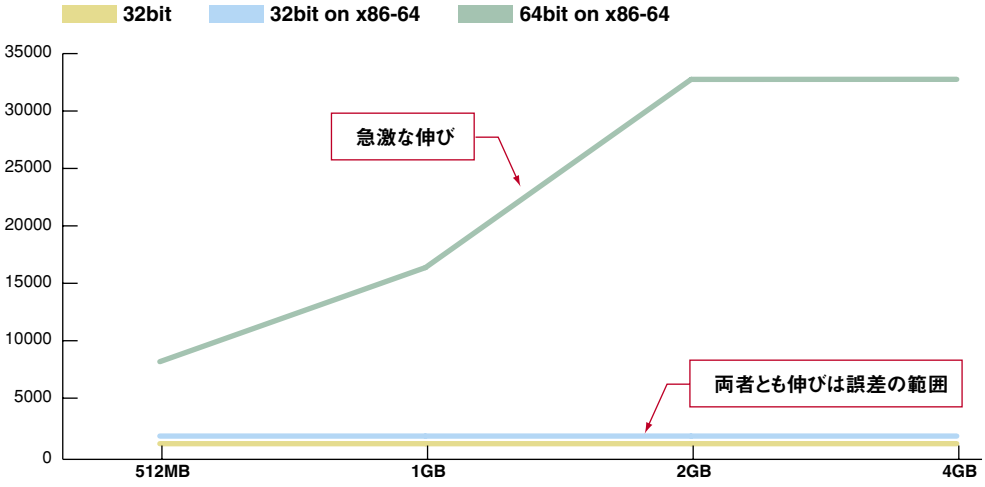


表 6　1 プロセスあたりの最大スレッド数

次に示すのは、プロセスが確保できるメモリを搭載メモリサイズごとにグラフ化したものである（表 7）。32bit 環境では、仮に 4096MB 以上の搭載メモリがハード内にあったとしても、扱うことができていないことが見て取れる。

また、x86-64 環境で 32bit アプリケーションを動作させた場合には、32bit アドレス空間の制約により、4GB までしかメモリを確保できないことも分かる。

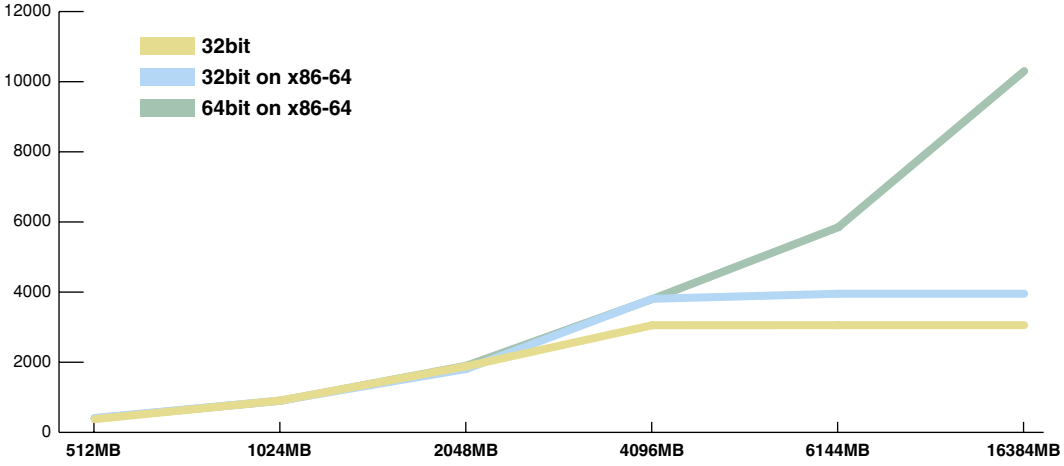


表 7　プロセスの最大確保可能メモリサイズ

V3.0 for x86-64
V3.0 for x86

差分 6

Disk I/O およびワークロード実行性能を確保

既存 IT 資産の有効活用	パフォーマンスの向上	
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

ディスク性能を測定する「bonnie++」で Disk I/O についてのテストを実行した。グラフに表れているように、どの項目も多少の差はあるものの、ほとんど性能に差がないことが分かる（表 8）。

またプロセス数を増やしながら、ワークロードの実行性能を比較できる「OSDL AIM ベンチマーク」でも、同様に、基本的な差はないことが見て取れる（表 9）。同一のハードウェアを利用した場合、32bit 動作と 64bit 動作では演算能力やハードウェア性能に準じた結果が素直に表れており、64bit 環境に変更することによるパフォーマンスの低下は見られないことが分かる。

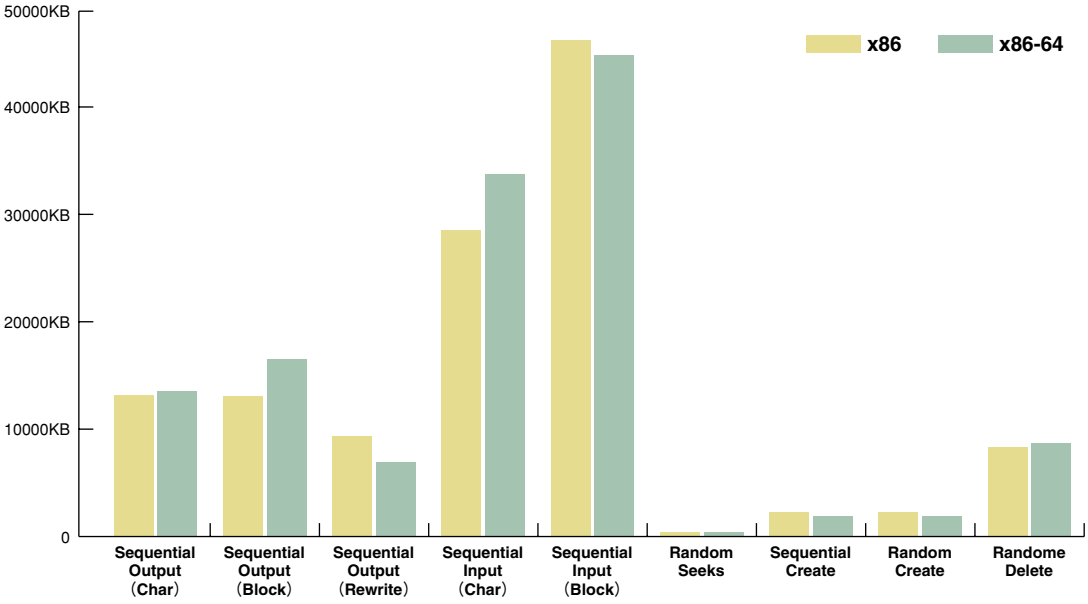


表 8　bonnie++（Disk I/O）

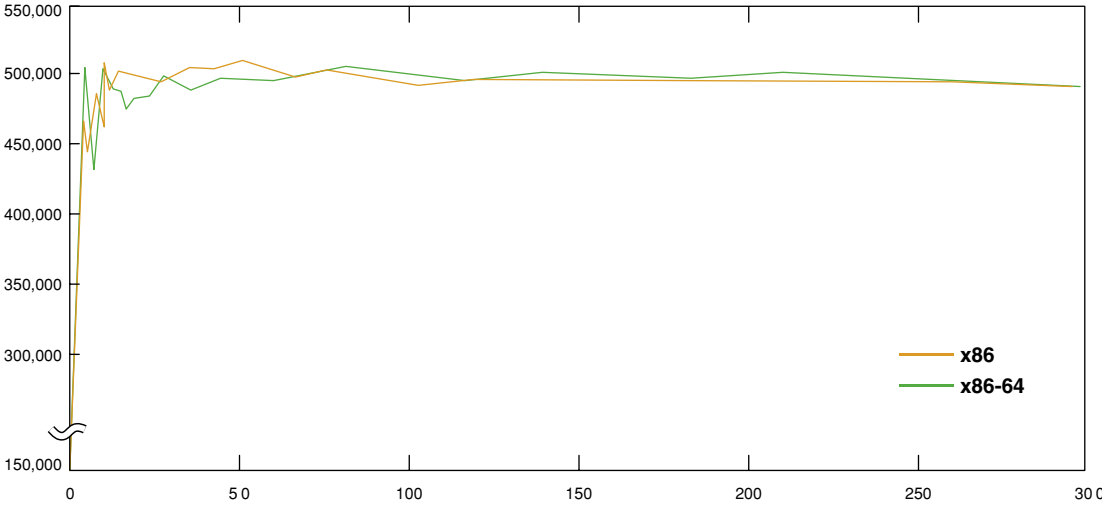


表 9　OSDL AIM ベンチマーク　（http://sourceforge.net/projects/re-aim-7/）

差分7

Java 上での性能が向上

既存 IT 資産の有効活用	パフォーマンスの向上	
32bit プログラムとの混在運用が可能	メモリ領域の拡大	64bit ネイティブ対応
差分 1	差分 3	差分 5
差分 2	差分 4	差分 6
		差分 7

次に、32bit および 64bit それぞれの Java 環境内で多数のスレッドを作成した結果をグラフにまとめてみる。V3.0 for x86-64 では、メモリの増加と共に飛躍的にスレッド数が伸びる結果となっている（表10）。

また、Java 演算ベンチマーク「scimark」ではどの項目も結果に差はない（表 11）。

様々な角度からのベンチマークにより、フル 64bit 環境で V3.0 for x86-64 を運用した場合に劇的なパフォーマンス向上が得られることが明らかとなった。また、32bit アプリケーションを実行した場合でも、エミュレータのように性能が低下するようなことはなかった。CPU 演算性能に頼る作業についても大きな変化はなく、ハードウェアのパフォーマンスどおりの結果となり、機能の低下は見られない。

V3.0 for x86-64 に関しては、導入によるデメリットはほぼ見られない。それよりも 32bit アドレッシングの制約が解消されることによるスケーラビリティの向上が大きなアドバンテージであると考えられる。

ちなみに、さまざまな要因で現時点では x86 版を導入せざるをえない場合もあるだろう。その場合には MIRACLE LINUX の有償サポート契約を結んでおくことで、OS の更新権を利用して x86 版から x86-64 版へ OS を変更することも可能である。ニーズによっては、将来の移行に備えつつ、x86 版を導入しておくことができるということになる。

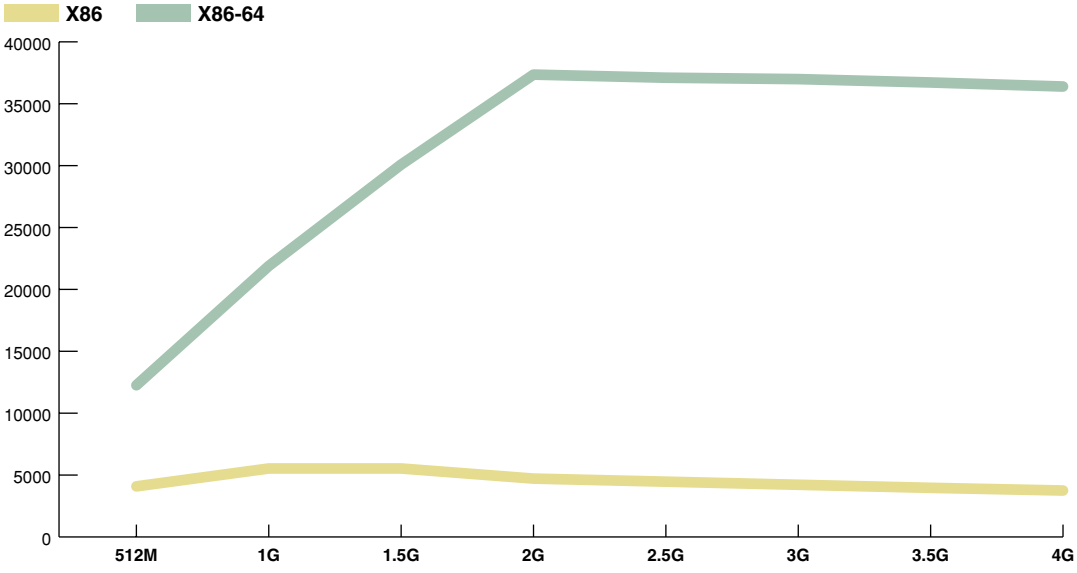


表 10　Java VM での最大スレッド数

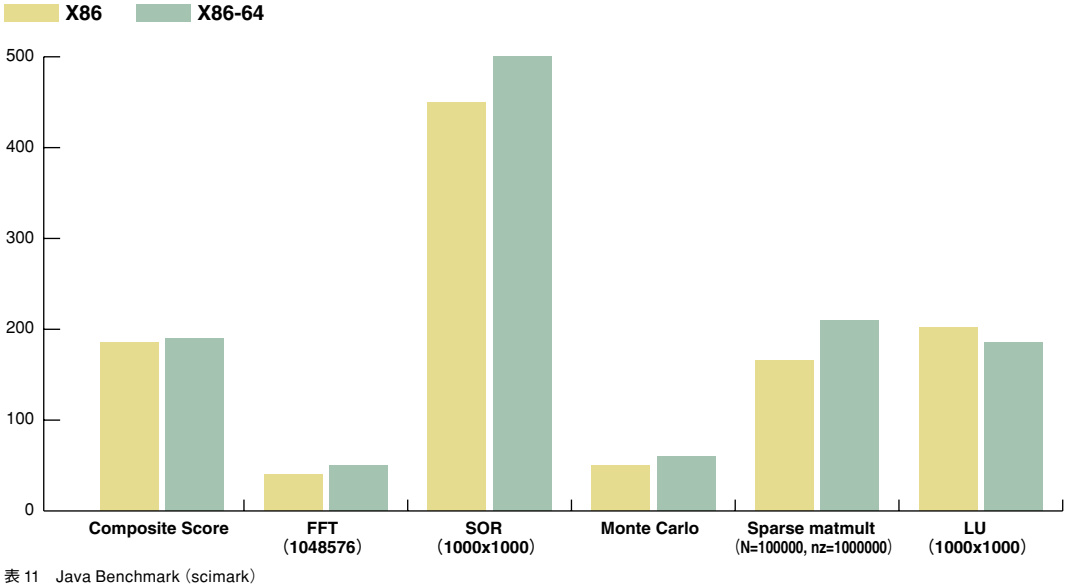


表 11　Java Benchmark（scimark）

［お問い合わせ先］

ミラクル・リナックス株式会社

〒105ー0021 東京都港区東新橋二丁目4番1号 サンマリーノ汐留 5階
TEL：03-5404-5050
FAX：03-5404-5051
URL：http://www.miraclelinux.com/