Solaris ZFS 解説書

2009 年 11 月 30 日 初版
2010 年 01 月 06 日 第 1 版
2021 年 05 月 31 日 第 2 版

はじめに

ZFS は今までのファイルシステムの概念を覆す画期的なファイルシステムです。OS/370(IBM MVS の後継 OS。現在は z/OS)、UNIX System V をはじめとする S5、UFS や VxFS、Linux の EXT2、EXT3、EXT4、Windows NTFS など、数多くのファイルシステムを使用してきましたが、ZFS ほどわくわくさせるファイルシステムは未だかつて触れることはありませんでした。

本解説書は、ファイルシステムそのものから ZFS の構造まで網羅的に解説するものです。ファイルシステムの概念から、ZFS の姿をできるだけ分かり易く解説します。ZFS ファイルシステムの概要、使用方法、構造などを解説することで、少しでも読者のお役に立てることができれば幸いです。

なお、本書に記述の ZFS は OpenSolaris 2009.06 をベースにしています。Solaris 10 の ZFS は必要に応じて付記しました。

内容

本文　第 1 章 ファイルシステムとは
基本的なファイルシステムの概念を解説します。

第 2 章 ZFS 概要
ZFS の概要を解説します。

第 3 章 ディスク管理と ZFS 操作
ZFS の使用方法、スナップショットやクローン、バックアップ方法を解説します。

第 4 章 ZFS インターナル
ZFS 構造、ディスク内部形式、チューニングなどを解説します。

第 5 章 性能
ZFS と UFS、SVM、ハードウェア RAID の性能比較を解説します。

第 6 章 TIPS
ZFS の評価で得た情報をご紹介します。

関連用語 ABC 順と五十音順の用語です。

付録　付録 A ZFS CLI コマンドリファレンス
ZFS で使用される zpool、zfs およびブート環境関連コマンドのリファレンスです。

付録 B ZFS を仮想マシンで使用してみる
物理的なハードウェアが無い場合に手軽に ZFS を体験する方法を解説します。

付録 C RAID 概要
RAID 0、1～RAID Z2 の概要を解説します。

付録 D 関連文献と WEB
本書作成にあたって参考にした文献、および関連 WEB の URL を記述します。

文中の略号について

N/A Not Applicable：情報が無いため、記述しないことを示します。

T.B.D To Be Described：今後の改訂で記述される可能性があることを示します。

目次

図目次

表目次

第 1 章 ファイルシステムとは

1-1. ファイルシステム概要

ファイルの割り当てや開放の管理、ファイル内のデータの読み書き管理、ファイルが入れられている領域の空きやしきい値などを管理するオペレーティングシステムの機能を総称してファイルシステムと呼んでいます。

ファイルシステムは、キャビネットやバインダ(ファイル)が置かれている「資料室」のようなものです…

資料室に行くと受付に目次(インデックス)があり、そこには、目的とする資料の在り処がバインダ(ファイル)の名前とともに書かれており、資料の作成日付や作成した人の名前、その場所に置かれた日付や、他の人が参照した日付などが記録されています。

新しい資料を保管するには、他の目次と同様に、必要な情報をそこに書き込みます。保管場所「A-1-2」キャビネットが空いていたので、目次に書き込んで割り当てます。目次をもとに「A-1-2」と書かれた場所に行き、その場所に資料を置きます。ファイルの割り当てです。

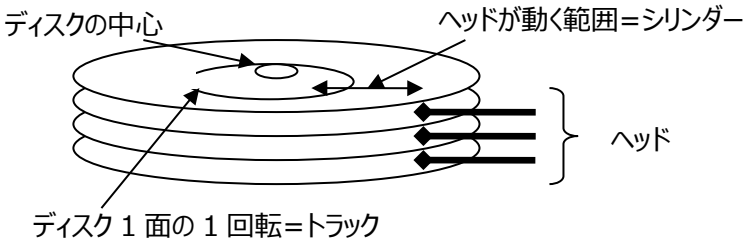
資料を参照するには、目的のバインダ(ファイル)を見つけます。「B-1-1」と書かれたキャビネットに行き、資料を参照します。ファイルの読み取りです。受付にある目次に参照した日付を記入します。

必要であれば内容を更新します。ファイルの更新です。受付にある目次に、更新した日付を記入します。

保管期限の過ぎた資料は廃棄します。資料を破棄して、受付の目次から抹消、または破棄日付を記入します。

コンピュータシステムでは「資料室」を外部補助記憶装置であるディスク装置(磁気ディスク)に置き換えます。

ディスク装置は磁性体をもった音楽レコードのような円盤に、レコードの針に相当するリードライトヘッドを介して情報を記録し、読み出します【備考】。



【解説】

●ディスク装置の構成

シリンダー　ヘッドが動く範囲。複数のヘッドが同時に位置づけ(シーク)されます。
ヘッド　　リードライトヘッド。
トラック　　あるディスク一面の 1 回転分。
セクター　　1 トラック中にあるレコード。単位は 512 バイトです。

●ディスク容量は次のように計算します。

一般に、シリンダー数×ヘッド数×1 トラックあたりのセクター数　で計算します。

【73 ギガバイトディスクの例】

8921 シリンダー×255 ヘッド×63 セクター×512 バイト=73,377,722,880 バイト≒73 ギガバイト

図 1-1-1. ディスク装置の概念

【備考】メモリ素子を磁気ディスクの代わりに使用する高速の SSD(Solid State Drive)があります。SSD においても、論理的にシリンダー、ヘッド、セクターとしてフォーマットしてアクセスしますので、磁気ディスクと同じと考えて良いです。

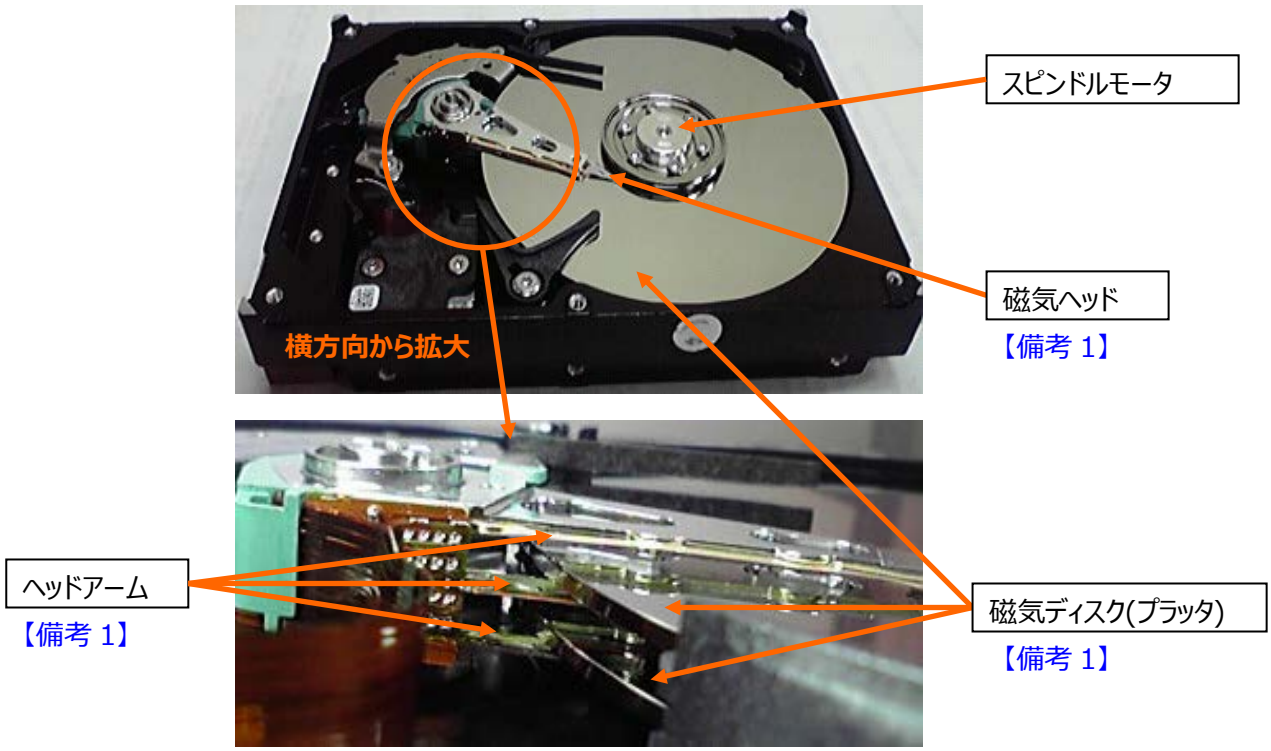


図 1-1-2. ディスク装置の実際

ファイルシステムは、このディスク装置全体、またはディスク装置を論理的に分割したスライス、またはパーティションと呼ばれる領域に作成します。

「図 1-1-2. ディスク装置の実際」とイメージが違いますが、ディスク内部の論理的なパーティション構成を示すため、シリンダー0(外側)から最後のシリンダー(内側)を示す構造を図で表してみました【備考 2】。

図 1-1-1 の 73GB ディスクの例を用います。なお、パーティションの名前はディスクをフォーマットした時点で名前が付けられます。パーティション番号もインストーラのフォーマットプログラムによって設定されます。

シリンダー**0** 名前：**boot** パーティション番号 **8** → スライス 8
シリンダー**1～8920** 名前：**root** パーティション番号 **0** → スライス 0

CYL	**CYL**
0	**1** **8920**
boot	**root** (“ルートファイルシステム”と呼びます)

図 1-1-3. ディスク装置とスライスの例

一方、ディスク装置全体や、スライスを直接使用する方法があります。これをロウ(生)モードと呼んでいます。ロウモードを使用する場合は、使用するプログラム自身が、ディスク装置そのものやスライスを、責任をもって操作する必要があります(一般にロウモードはファイルシステムと呼びません)。

【備考 1】初期のディスク装置は、プラッタ数とヘッド数が対応していました。例えば、「プラッタが 10 面の場合、プラッタ表裏にヘッドがあり、一つがタイミングヘッドだとすると 19 ヘッド」のような計算です。最近のディスク装置は、物理的なヘッド数とは別に論理的ヘッド数を持っています。このため、必ずしもプラッタ数とヘッド数は例の計算のようにいきません。

【備考 2】ハードディスクの場合、CD-R のように内側から外側ではなく、外側が 0 で内側が最後のシリンダーです(レコード盤と同じイメージです)。このため、ディスク装置のデータシートでは、外周(シリンダー0)の転送速度が速く、内周(最後のシリンダー)の転送速度が遅いです。【例】Seagate Savvio 10K.2 の例：外周～内周=89～55MB/秒の転送速度となっています。

図 1-1-3 のディスク装置とスライスの例は、OpenSolaris の OS をインストールすることによって、73GB のシステムディスクがフォーマットされ、パーティションに区切られた状態を示しています。システムディスク装置のフォーマットは OS のインストールプログラムが行います。

次に、具体的な **format** と呼ばれるシステムコマンドのパーティション(**partition**)機能による印書(**print**)リストを示します(リストの一部分を抜粋しています)。

```
# format ⏎ ←フォーマットプログラムはこのように起動します。
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c8t0d0 <DEFAULT cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@0,0
Specify disk (enter its number): 0 ⏎←
selecting c8t0d0
[disk formatted]
/dev/dsk/c8t0d0s0 is part of active ZFS pool rpool. Please see zpool(1M).
      :
format> partition ⏎
      :
partition> print ⏎
Current partition table (original):
Total disk cylinders available: 8921 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders         Size            Blocks
  0       root   wm       1 - 8920        68.33GB    (8920/0/0) 143299800
  1 unassigned   wm       0                   0      (0/0/0)           0
  2     backup   wu       0 - 8920        68.34GB    (8921/0/0) 143315865【備考】
  3 unassigned   wm       0                   0      (0/0/0)           0
  4 unassigned   wm       0                   0      (0/0/0)           0
  5 unassigned   wm       0                   0      (0/0/0)           0
  6 unassigned   wm       0                   0      (0/0/0)           0
  7 unassigned   wm       0                   0      (0/0/0)           0
  8       boot   wu       0 -     0        7.84MB    (1/0/0)       16065
  9 unassigned   wm       0                   0      (0/0/0)           0
      :
format> quit ⏎
#
```

ディスク装置名

構成されているディスクの一覧表が表示されます。

選択に使用する番号

システムディスクは通常、"0"番です。

図 1-1-3 の右側の領域

図 1-1-3 の左側の領域

図 1-1-4. format コマンドのパーティション表示例

【図 1-1-4 partition 出力情報の解説】

Part パーティションの番号を示しています。図 1-1-3 の図上部に示す情報です。
Tag パーティションに付けられた名前を示しています。図 1-1-3 の図内の名前です。
Cylinders シリンダーの範囲を示しています。図 1-1-3 の図内に示すシリンダー値です。
Size そのパーティションの大きさを示しています。図 1-1-1 の計算例をご覧ください。
Blocks そのパーティションのブロック(セクター)数を示しています。数字の左側にある括弧内の値は(シリンダー/ヘッド/セクター)を示しています。図 1-1-1 の計算例をご覧ください。

【備考】パーティション番号 2 の"backup"とは、ディスク全体を示すものです。シリンダーは"0-8920"となっています。一つの連続した領域で区切られていませんが、パーティション番号が振られています。「バックアップパーティション」と呼ばれます。

1-2. ファイルシステムの比較

外部補助記憶装置をボリューム(Volume)として管理する OS/370 では、VTOC(Volume Table Of Contents)と呼ばれるテーブルを持ち、その配下にファイルを置いて管理しています。一つのディスク装置の先頭ブロックには VTOC があり、ファイル名とディスク上の位置を持つラベルが続いて配置されます。形式はスタンダードラベルと呼ばれ、JIS 規格でも定められていることで知られています【備考 1】。

UNIX System V Release 4(SVR4)では当初 S5 と呼ばれるファイルシステムがありました。フォーマット時に決定されるファイルシステムの先頭 10～40%にファイルの名前、各種属性、ファイル位置を示すインデックスノード(i ノード【備考 2】)と呼ばれる「目次」の領域が置かれ、残りがデータブロックとして用いられました。ファイルのアクセスは、まず i ノードを参照し、i ノード内にあるデータ位置を元にデータブロックを参照していました。この方式の欠点は、フォーマット時に i ノードとデータブロックの割合を決定する必要があり、後に i ノードが不足したり、データブロック領域が不足したりするとファイルシステム全体を再構築する必要がある点、もう一つは、ディスク装置の後方にあるデータブロックは、シリンダー0 付近の i ノードを先にアクセス、次にディスク装置の終わりに近いブロックをアクセスすることになり、シーク距離が長くなる点でした。必然的に、ディスク装置後方のデータアクセスは時間がかかる結果になりました【備考 3】。

この S5 ファイルシステムの欠点を解消したのが UFS(Unix File System)です。パーティション内部はシリンダーグループと呼ばれる 16 シリンダー単位で区切られた範囲に、i ノードとデータブロックを収めることでシーク時間の短縮を図りました。ディスクのシークは平準化され、シーク距離を短縮することに成功しました。しかし、i ノードとデータブロックのバランスをフォーマット時に決定しなくてはならない問題は解消できませんでした。

VxFS や Windows NTFS【備考 4】は、エクステントと呼ばれる単位でパーティション内部を分割して管理することで、性能と保守性を向上させました。VxFS はアロケーションユニットと呼ばれるエクステント単位で i ノードとデータブロックをまとめて管理します。UFS ファイルシステムでは、システムがクラッシュした後のファイルシステムチェック(fsck)によるファイル回復に長い時間がかかっていましたが、VxFS ではインテントログと呼ばれる独自のログ機構で、システムクラッシュ後のファイルシステム検査を高速に処理することができる機能を持っていました。

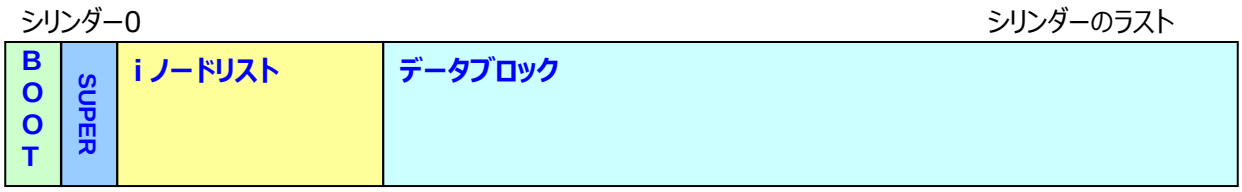
EXT2、3、および 4 は Linux のファイルシステムです。EXT3、4 ではジャーナリング機能があります【備考 5】。

これらのファイルシステム全てに共通して、扱うことのできるデータ量の制限は、ハードウェアアーキテクチャのビット幅そのものでした。また、ディスク装置単体では物理的に限られたデータ量のため、例えビット幅がたくさんあったとしても制限がありました。

【備考 1】現在 OS/370、MVS は OS/390 を経て z /OS と呼ばれており、メインフレームから UNIX まで動作させることのできる統 合 化された OS になっています。ここでは比較として取り上げています。
【備考 2】i ノードを含む、他の種類のディスク制御ブロックは、各種属性やブロック情報を持っていることから、「メタデータ」と呼ばれています。
【備考 3】S5 ファイルシステムは UNIX System V Release 4 の初期で使用されていたファイルシステムです。ファイルシステムの i ノードとデータブロックのスタイルとして、基本的な概念の説明のため記述しています。
【備考 4】NTFS の詳細構造は参考文献にありました。NTFS は VxFS のコピー版のように見えます。
【備考 5】Linux で一般的なファイルシステムです。ここでは名称のみ取り上げます。

1-2-1. S5(UNIX System V Release 4 File System)

レガシーなファイルシステムです。S5 ファイルシステムは次の構造を持っています。



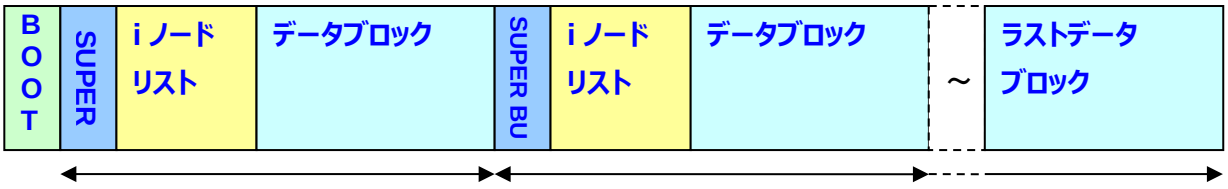
BOOT	ブートブロック(boot blodk)
SUPER	スーパーブロック(super block)
i ノードリスト	i ノードリスト(inode list)。データブロックをインデックスします。直接ブロックは 10 ブ ロック、1 次間接ブロックは 1 階層、2 次間接ブロックは 2 階層、3 次間接ブロックは 3 階層のインデックスを持ちます。
データブロック	データブロック(data block)

図 1-2-1-1. S5 ファイルシステムの構造

この構造から、データブロックが後方にあると、ヘッドシークに多くの時間を消費することがわかります。

1-2-2. UFS(Unix File System)

UFS は、S5 ファイルシステムの欠点をカバーするため、ファイルシステムをシリンダーグループに分けてフォーマットして管理するものです。UFS は次の構造を持っています。

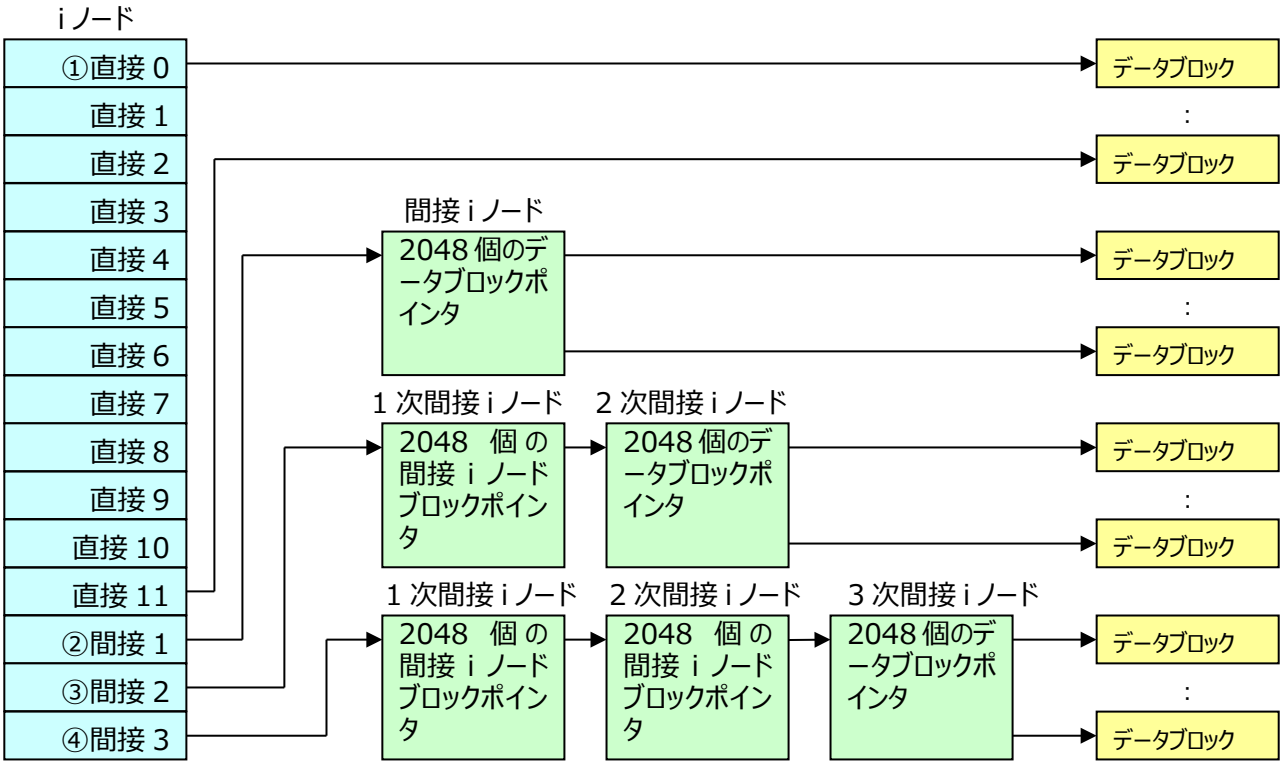


BOOT	ブートブロック(boot block)
SUPER	スーパーブロック(super block)
i ノードリスト	i ノードリスト(inode list)。データブロックをインデックスします。直接ブロックは 12 ブロック、1 次間接ブロックは 1 階層、2 次間接ブロックは 2 階層、3 次間接ブロックは 3 階層のインデックスを持ちます。
データブロック	データブロック(data block)
シリンダーグループ	16 シリンダー単位で区切られたグループです。i ノードとデータブロックは出来るだけ隣接するシリンダーグループに置かれます。
SUPER BU	スーパーブロックバックアップ(super block backup)。各シリンダーグループの先頭にあります。SUPER ブロックが破損した場合のバックアップになります。

図 1-2-2-1. UFS の構造

S5 ファイルシステムの欠点が解消され、ヘッドシークを平準化することができます。

次に UFS i ノードの持つブロックポインタがデータブロックをどのように参照しているのかを見てみます。



【解説】

① i ノードには直接データブロックを参照するディスクアドレスエントリを 12 個持っています。ブロックサイズを 8,192 バイトとすると、12×8,192=98,304 バイトのデータが収まります。

② 98,304 バイトを超えるサイズのファイルでは、間接 1 の示すブロックに間接 i ノードが置かれます。間接 i ノードに収まるデータブロックアドレスのエントリは 8,192÷4=2,048 個を持つことができます。データブロックアドレスの数が 2,048 個のため 2,048×8,192=16,777,216 バイトのデータが収まります。

③ ①+②を超えるサイズのファイルでは間接 2 の示すブロックに 1 次間接 i ノード、その次に 2 次間接 i ノードが作られます。各々に 2,048 個のエントリが作成できます。データブロックアドレスの数は、2,048×2,048 個、計 2,048×2,048×8,192=34,359,738,368 バイトのデータが収まります。

④ ①+②+③を超えるサイズのファイルでは、間接 3 の示すブロックに 1 次間接 i ノード、次に 2 次間接 i ノード、その次に 3 次間接 i ノードが作られます。各々に 2,048 個のエントリが作成できます。データブロックアドレスの数は、2,048×2,048×2,048 個、計 2,048×2,048×2,048×8,192=70,368,744,177,700 バイトのデータが収まります。

⑤ 結果、一つのファイルの最大サイズは、①+②+③+④で求めます。合計で約 70TB になります。

図 1-2-2-2. UFS のデータブロック参照の仕組み

ディスクブロックの位置を示すブロックポインタは 4 バイト(32 ビット)です。32 ビット幅では上図のように一個のファイルサイズが制限されることがわかります。

ファイルシステムそのものも、同様の制限を受けることになります。ブロックの位置付けはディスクセクターサイズの単位である 512 バイトがベースになっていることが理由です。

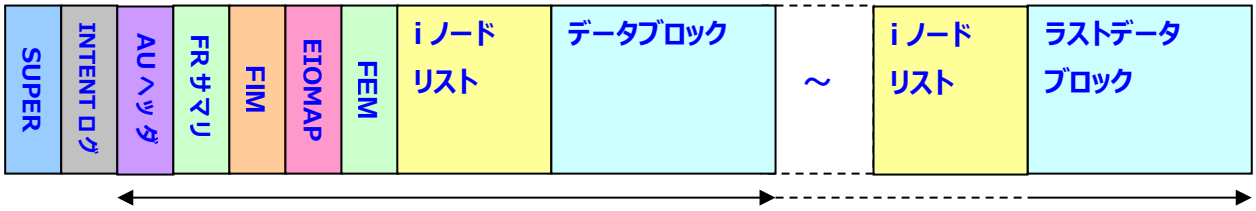
実際のポインタ操作は、値を 1 ビット左にシフト(2 倍)して操作しています。

このように、プロセッサアーキテクチャのビット幅は、ファイルシステムに大きく影響します。

【参考】後述の ZFS は 128 ビットのファイルシステムです。ZFS におけるディスクのブロック位置計算は、63 ビットの値=オフセットを 9 ビット左にシフトし、0x400000(4MB)をプラスしたところを物理ブロックの位置として使用しています。

1-2-3. VxFS(VERITAS File System)

VxFS はアロケーションユニット単位で領域が割り当てられます。データブロックはエクステント内の連続したエリアに確保されるため、データが分散することなく、冗長なヘッドシークが抑えられます。次に VxFS の構造を示します(BOOT ブロックは記述していません)。

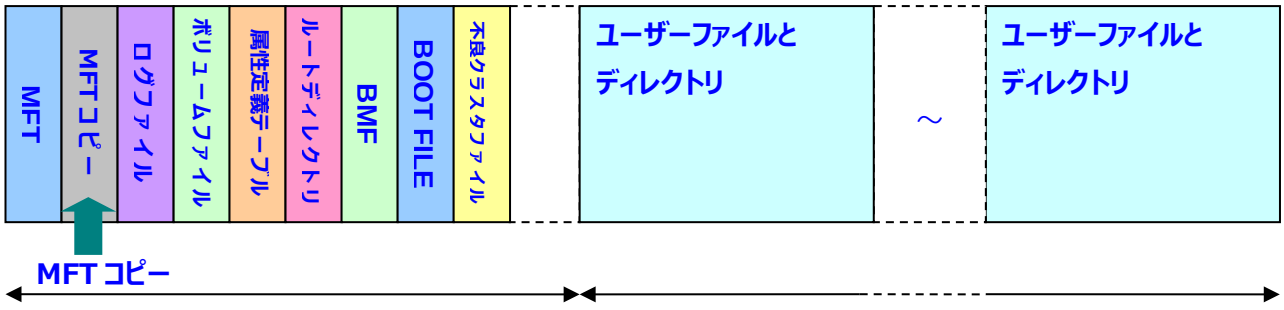


SUPER	スーパーブロック(super block)
INTENT ログ	インテントログ(intent log)
AU ヘッダ	アロケーションユニットヘッダ(allocation unit header)
FR サマリ	フリーリソースサマリ(free resource summary)
FIM	フリーi ノードマップ(free inode map)
EIOMAP	拡張 i ノード操作マップ(extended inode operations map)
FEM	フリーエクステントマップ(free extent map)
i ノードリスト	i ノードリスト(inode list)
データブロック	データブロック(data block)
アロケーションユニット	Allocation unit

図 1-2-3-1. VxFS の構造

1-2-4. NTFS(NT File System)

NTFS は MS-DOS の FAT や HPFS(OS/2)の次期ファイルシステムとして開発されました。大規模構成と信頼性の向上を目的として、復旧機能やセキュリティ機能が追加されています。次に NTFS の構造を示します。



MFT	マスターファイルテーブル(master file table)
MFT コピー	MFT のコピー(master file table copy:ディスクの中央付近に割り当てられる)
ログファイル	Redo ログファイル(Redo log file)
ボリュームファイル	Volume file
属性定義テーブル	Attribute definition table
ルートディレクトリ	ディレクトリの先頭(root directory)
BMF	ビットマップファイル(bit map file)
BOOT FILE	ブートファイル
不良クラスタファイル	Bad cluster file
ユーザーファイルとディレクトリ	クラスタ単位で割り当てられるユーザーファイルとディレクトリ

図 1-2-4-1. NTFS の構造

1-2-5. ファイルシステムを比較する

次に各ファイルシステムの比較を示します。表中のジャーナルとは、ファイルの回復機能です。

表 1-2-5-1. ファイルシステムの比較

No	名前	ファイル名長	パス名長	最大ファイルサイズ	最大ボリュームサイズ	ジャーナル
1	UFS	255 バイト	制限無し	4GB～256TB【備考 1】	256TB	有り
2	VxFS	255 バイト	制限無し	16EB	16EB	有り
3	NTFS	255 文字	32,767 文字	16TB【備考 2】	256TB	有り
4	EXT3	255 バイト	制限無し	16GB～2TB	2TB～32TB	有り

【備考 1】図 1-2-2-2 では約 70TB と計算されます。本表の値は参考 WEB より引用しています。

【備考 2】NTFS はファイルの断片化を最適化するデフラグ機能が最大の長所だと思います。UNIX のファイルシステムにもこのデフラグ機能が欲しいところです…

【参考】本解説書で使用のサイズ表記記号は次表に基づきます。

表 1-2-5-2. サイズ表記

SI 名称	SI 接頭辞	漢字の表記と読み	2 の乗数	10 進数	16 進数 【備考 1】
キロ kilo	K	千	10	1,024	00000400
メガ mega	M	百万	20	1,048,576	00100000
ギガ giga	G	十億	30	1,073,741,824	40000000
	4G	四十億	32	4,294,967,296	00000001 00000000
テラ tera	T	一兆	40	1,099,511,627,776	00000100 00000000
ペタ peta	P	千兆	50	1,125,899,906,842,624	00040000 00000000
エクサ exa	E	百京 けい	60	1,152,921,504,606,846,976	10000000 00000000
	16E	千八百京	64	18,446,744,073,709,551,616	00000001 00000000 00000000
ゼタ zetta	Z	十垓 がい	70	1,180,591,620,717,411,303,424	00000040 00000000 00000000
ヨタ yotta	Y	一杼 じょ	80	1,208,925,819,614,629,174,706,176	00010000 00000000 00000000
		千杼	90	1,237,940,039,285,380,274,899,124,224	04000000 00000000 00000000
		百穣 じょう	100	1,267,650,600,228,229,401,496,703,205,376	00000010 00000000 00000000 00000000
【備考 2】		十溝 こう	110	1,298,074,214,633,706,907,132,624,082,305,024	00004000 00000000 00000000 00000000
		一澗 かん	120	1,329,227,995,784,915,872,903,807,060,280,344,576	01000000 00000000 00000000 00000000
		百澗	128	340,282,366,841,710,300,949,110,269,842,519,228,416	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 【備考 3】

【備考 1】Big Endian 形式です。
【備考 2】128 ビットまで記述した理由は、後述の ZFS のためです。SI 名称と接頭辞は現在のところ JIS Z 8203 に定義はありません。
【備考 3】厳密には“0x1 00000000 00000000 00000000 00000000”ですが、この値では 128 ビットを超えるため、マイナス 1 します。これより、“0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF”(拡張倍精度)を用いました。

これらのファイルシステムは主にスライス(パーティション)内部に作成して使用します。ファイルシステムは一般に次の手順で使用することになります。

① 将来のデータ増加を予測してストレージ構成を設計
② アプリケーション単位でデータのバックアップ計画を立てる
③ ディスク機器を導入して設置
④ ディスクを各アプリケーションに割り当てる
⑤ ファイルシステムに応じて設定しフォーマットを実行
⑥ アプリケーションで使用
⑦ 必要に応じてデータをバックアップ
⑧ 障害があった場合、ファイルシステムのバックアップから戻す(機器を入れ替えて④⑤⑥⑦繰り返し)
⑨ ディスクがもし満杯になると機器を増設(①②③④⑤⑥⑦繰り返し)

前述のいずれのファイルシステムでも、同様の作業ステップが必要になります。

さて、ディスク装置を見た場合、近年、1 個で 20TB もの容量のディスク装置が出現しており、通常の使用では十分なサイズになっています。一方、ディスク装置の容量のみならず、対障害性の向上や、性能向上のため、さらに大きなファイルシステム「ボリューム」を構成する場合、ストレージの仮想化技術が必要になってきます。

ストレージ仮想化技術は、ボリュームマネージャと RAID 技術の出現で大きく進化しました。RAID 技術は大量のディスク装置を安価で安全に使用することができるストレージの仮想化技術で、ファイルシステム管理の方法を大きく変えることになりました。

表 1-2-5-3. OpenSolaris のファイルシステム

No	名称	使用目的	内容
1	autofs	自動マウント	自動マウントのパラメタと値を持つファイルシステム。
2	cachefs	メモリキャッシュ	主記憶上に作成されるキャッシュファイルシステム。
3	ctfs	コンタクト	ブート時に/system/contact にマウントされます。
4	dcfs	圧縮ファイルシステム	ブートアーカイブに用いられるファイルシステム。
5	devfs	デバイスファイルシステム	OS 環境下の全デバイスの名前空間を管理します。
6	fifofs	FIFO ファイルシステム	FIFO(パイプ)ファイルシステムです。
7	hsfs	High Sierra ファイルシステム	ISO 9660 CD-ROM ファイルシステムです。
8	lofs	ループバック仮想ファイルシステム	ループバックの仮想ファイルシステムです。
9	mntfs	マウントファイルシステム	マウントされるファイルシステム情報を持つファイルです。
10	namefs	名前空間ファイルシステム	名前空間のファイルシステムです。
11	nfs	NFS	ネットワークファイルシステムです。
12	objfs	カーネルオブジェクトファイルシステム	ブート時に/system/object にマウントされます。
13	procfs	プロセルファイルシステム	/proc ディレクトリ下のプロセス情報ファイルシステムです。
14	sharefs	カーネル sharetab ファイルシステム	/etc/dfs/sharetab で定義された共有ファイルシステムです。
15	smbfs	CIFS ファイルシステム	CIFS(Samba)ファイルシステムです。
16	sockfs	ソケットファイルシステム	Socket で使用するファイルシステム。
17	specfs	特殊ファイルシステム	特殊ファイル(SPEC)のファイルシステムです。
18	tmpfs	メモリベースファイルシステム	VM やページキャッシュの使用するメモリファイルシステム。
19	udfs	UDF 形式ファイルシステム	UDF(Universal Disk Format)ディスクのファイルシステム。
20	ufs	UFS ファイルシステム	UNIX File System です。
21	zfs	ZFS ファイルシステム	本書で解説のファイルシステムです。

【備考】ファイルシステムは上記以外に数多くあります。Wikipedia にまとまった記事があります。巻末の参考 WEB をご覧ください。なお、現在 OpenSolaris で扱うことが可能なファイルシステムは表 1-2-5-3 の通りです。なお、グレー表記は一般の使用者が意識して使用することは少ないです。

1-3. ストレージの仮想化

「1-2. ファイルシステムの比較」では一つのディスク装置、またはスライスにファイルシステムがどのように作られるのかを解説しました。ストレージの仮想化は、単体ディスク装置の容量を超えたファイルシステムを構成する場合や、システムの安定稼働を目的として使用されます。

論理ボリュームマネージャ(Logical Volume Manager)は、次のような機能を提供します。

- 複数の物理ディスク(PV : Physical Volume)をまとめることで大きなサイズのボリュームグループ(VG : Volume Group)を作成する。
- VG を論理的なボリュームに区切り(スライス)論理ボリューム(LV : Logical Volume)として使用する。
- 論理的に区切った LV のサイズを拡張(UFS では growfs コマンド)や縮小する。
- LV に PV を追加、削除する。
- RAID を構成する。

これらの機能によって単一ディスク装置の容量、機能を超えてストレージを提供します。

ボリュームマネージャには VERITAS Volume Manager や Solaris Volume Manager、Linux LVM (Logical Volume Manager)などがあります。いずれもシステムディスクへの適用に制限があります。また、ディスク装置の交換時、システム本体や部分停止を伴う操作を行わなくてはならないことがあります。

他に、ストレージや OS によって各種のボリュームマネージャが存在しますが、本解説書では割愛させていただきます。

特筆すべきは、いずれのボリュームマネージャも、使用の際、ソフトウェアの操作、特性を熟知する必要がある点です。

1-3-1. 大容量ボリューム構成

単体ディスクをまとめて大容量化することで、大規模データベースや巨大なファイル構成に使用することができます。下記の構成例では VG を一つのファイルシステムで使用すると 365GB の容量になります。

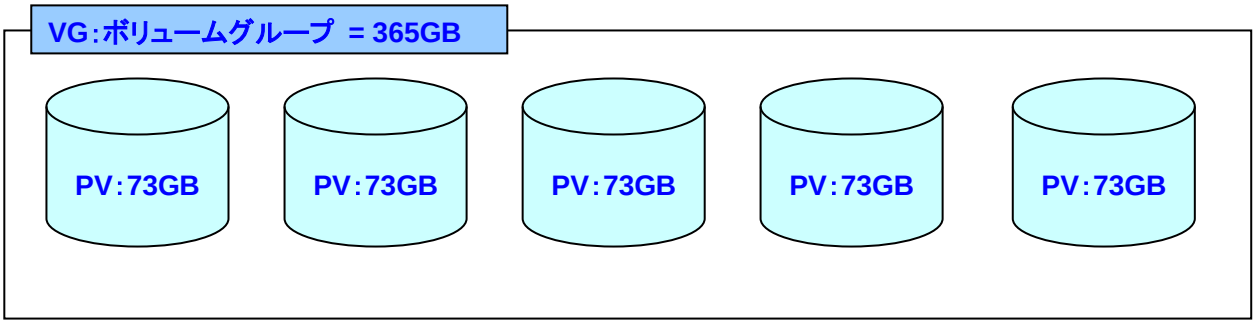


図 1-3-1-1. 大容量ボリューム構成例

1-3-2. LV の区分化構成

大容量化したボリュームを論理的に区切って使用します。各スライスにアプリケーション、データベースなどを配置して使用します。ファイルシステム A、B、C は LV 名のみで、もはや物理ディスクを知ることはありません。アプリケーションは各々の LV を別ファイルシステムとして使用することができます。

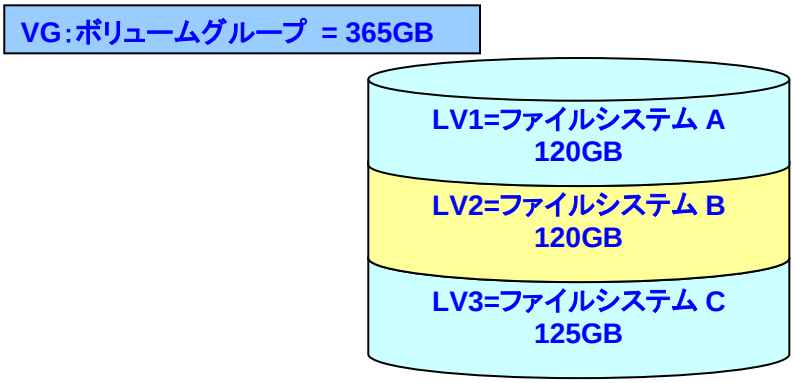


図 1-3-2-1. ボリュームの区分化構成例

1-3-3. LV の拡張

Solaris Volume Manager では構成した LV に対して growfs コマンドでサイズを拡張することができます。このように表現するとあたかも簡単に拡張できるように感じますが、設定手順は複雑で、ボリューム名やデバイス名を完全に把握して、間違いのないように操作する必要があります。また、操作中、当該のファイルシステムは停止する必要があります。

1-3-4. LV への PV 追加と削除

Solaris Volume Manager では構成した LV へ PV を追加や、削除することができます。LV の拡張と同様に設定手順は複雑で、ボリューム名やデバイス名を完全に把握して、間違いのないように操作する必要があります。また、操作中、当該のファイルシステムは停止する必要があります。

1-3-5. RAID 構成

Solaris Volume Manager、VERITAS Volume Manager で構成できる RAID 方式を示します。

表 1-3-5-1. SVM と VERITAS Volume Manager の RAID 方式

SW/RAID	RAID 0	RAID 1	RAID 1+0	RAID 0+1	RAID 5
SVM	○	○	○	○	○
VERITAS	○	○	○	○	○

これらはソフトウェア RAID です。ストレージのハードウェア RAID もありますが、本解説書では割愛させていただきます。なお、付録 C に上記以外の RAID 方式も含めて解説します。ご覧下さい。

1-4. ZFS ファイルシステムの登場

「1-2. ファイルシステムの比較」と、「1-3. ストレージの仮想化」で解説した技術は、それぞれ長所・短所があります。ファイルシステムを使用する立場から考えると、これらの長所と短所、ストレージ設計をはじめとして、全ての技術に習熟しなくてはならないというプレッシャーがつきまとっていました。

しかしながら、ZFS は、技術を習熟しなくてはならないというプレッシャーから見事に解き放ってくれました。

1-4-1. ストレージプールという概念

ストレージの集まりを示す名称、「ボリューム」という概念は無くなり、「ストレージプール」と呼ばれる領域が用いられます。使用者はその「プール」から必要な領域を割り当て、使用していくことになります。

1-4-2. 群を抜くスケーラビリティ

表 1-2-5-1 のように、既存のファイルシステムでも 256TB ものサイズでファイルステムを構成することができます(70TB、または 16TB?)。しかし、先に物理サイズが限界になるとはいえ、ZFS のスケーラビリティは群を抜いています。将来に向けて安心のスケーラビリティと言えます。前述、表 1-2-5-2 も合わせて参考にして下さい。

表 1-4-2-1. ZFS の制限値

No	項目	2 の乗数 (ビット数)	値	単位
1	プールサイズ【備考】	78	302,231,454,903,657,293,676,544	バイト
2	プール数	64	18,446,744,073,709,551,616	個
3	ファイルシステムサイズ	64	18,446,744,073,709,551,616	バイト
4	ファイルサイズ	64	18,446,744,073,709,551,616	バイト
5	ディレクトリ下のファイル数	48	281,474,976,710,656	個

1-4-3. 極めて簡単で直感的なコマンド

ZFS を通常運用で使用する主な管理コマンドは、ストレージプールを操作する zpool コマンドと、ファイルシステムを操作する zfs コマンドの 2 個です。直感的なファンクションで非常に使いやすいのが特徴です。

1-4-4. RAID マネージャ不要

簡単な操作で RAID 機能を持つ ZFS ストレージプールを作成することができます。RAID 方式を明示的に指定しないでストレージプールを作成した場合、デフォルト RAID 0 で構成されます。ミラーリング(RAID 1)、RAID Z(RAID 5 相当)、また RAID Z2(RAID 6 相当)のストレージプールを簡単に構成できます。ホットスペアの設定も可能で、故障ディスク装置の入れ替えも簡単な操作で行うことができます。

1-4-5. 安心のデータ保全

COW(Copy On Write)トランザクションと呼ばれる書き込み方式で、コミットが完全に実行されます。エラーになっても元のデータが残されます。ディスクブロックはチェックサムを持っており、不良データは自動的に検出し復旧します。ディスクスクラブはチェックサムの検証とデータ復旧、再同期処理をバックグラウンドで実行します。データセキュリティは、厳密な ACL(アクセス制御リスト：Access Control List)とデジタル署名で万全です。

【備考】参考 WEB のプールサイズ値、は 2 の 77 乗である“151,115,727,451,828,646,838,272”となっていました。2 の 78 乗を計算すると表 1-4-2-1 の値欄で示す数字になります(Microsoft Excel でも近似の値になります)。

第 2 章 ZFS 概要

ZFS の特徴を概説します。

ストレージプール

ZFS は Oracle 社に吸収された Sun Microsystems が開発した究極といっても良い画期的なファイルシステムです。既存のファイルシステムは、ディスク装置単体、または複数のディスク装置を設置し、ボリュームマネージャでボリュームを構成し、個々のファイルシステムに応じた手順でストレージを構築します。アプリケーションは目的のボリュームに構成されたファイルシステムの容量や特性を意識して使用する必要があります。

ZFS ではボリュームの概念を取り払い、ストレージプールと呼ばれる領域をディスク装置単体、またはディスク装置の集まりとして構成します。アプリケーションは必要に応じて領域を確保(ストレージプールから切り出し)して使用する形になります。ZFS を紹介する各種文献では、この様子を「メインメモリのようだ」と例えています。

スケーラビリティ

既存のファイルシステムは、プロセッサアーキテクチャの 32 ビットや 64 ビットなどのビット幅によって容量に制限があります。32 ビットでは 4GB(2 の 32 乗=4,294,967,296 バイト)、64 ビットでは 18EB(2 の 64 乗 =18,446,744,073,709,551,616 バイト)です。

ZFS はネーミングにあるように、zettabytes=2 の 70 乗=十垓(がい)= 1,180,591,620,717,411,303,424 バイトのアドレス空間をストレージ上で管理することができます。

ZFS は 128 ビットのファイルシステムですが、基本的にプロセッサのアーキテクチャに依存しません。128 ビットの内、64 ビット(内部では 63 ビット)はデバイス仮想アドレス(DVA：Device Virtual Address)、70 ビットを超えるビットは、仮想デバイス(VDEV：Virtual Device)のポインタとして使用し、ストレージプール全体を管理しています。このため、事実上、無制限ともいえるデータ空間を管理することができます。

簡単な管理コマンド

既存のファイルシステムを構成する時、ディスク装置や複数ディスク装置を設置してストレージを構成します。目的に応じて、ボリュームマネージャや、各種の RAID マネージャを用いてファイルシステムを作成します。ファイルシステムは種類に応じて各種のコマンドを使い分け、加えて、マウントのための情報を/etc/fstab に記述することでブート時に自動マウントされ、アプリケーションで使用することができるようになります。システム運用を開始すると、バックアップや回復手順を実行しますが、この際、構成に使用した各種ソフトウェアや RAID マネージャはもとより、コピーや復元のためのコマンドを駆使して運用しなくてはなりません。

ZFS では、ストレージプールの作成・削除などを管理する zpool コマンドと、ファイルシステムの作成・削除などを管理する zfs コマンドの 2 個のコマンドを使用します。マウントは基本的に自動マウントで、/etc/fstab に記述する必要はありません。バックアップや復元は自システム内だけではなく、UNIX のパイプや基本的なコマンドを組み合わせることで簡単に遠隔バックアップや復元ができます。

瞬時にバックアップを作成できるスナップショット機能も魅力的です。読み書きの出来るバックアップを作成するクローン機能や、ストレージプールそのものをエクスポート/インポートする機能もあります。

RAID 機能

既存のファイルシステムでは、ボリュームマネージャや RAID マネージャを使用して RAID レベルや、ホットスワップを構成します。いくつかのソフトウェアが必要になります。エラーの発生したディスク装置の切り離しや接続は、場合によって、システム停止やアプリケーションの停止を伴うことがあります。

ZFS は RAID 構成だけではなく、特定のディスク装置をホットスワップにして耐故障性を高める機能を持っています。また、ボリュームマネージャや RAID マネージャのように、特定のディスク装置を削除や接続することが可能です。エラーの発生したディスク装置の切り離しや接続時、システム停止やアプリケーションを停止することなく作業することができます。

ストレージプールを ZFS として構成すると RAID 0 のストライピングが仮定されます。ミラーリングの RAID 1、RAID 5 相当の RAID Z、RAID 6 相当の RAID Z2 を、ストレージプールを構成する zpool コマンドの引数に指定することで簡単に RAID を構成することができます。

ZFS はファイルシステム機能の他に、ボリュームマネージャや RAID マネージャの機能を併せ持つ高機能の優れたファイルシステムです。

データ保全

既存のファイルシステムでは、アプリケーションから実行したデータの書き込みから、実際にディスク装置上に書き込まれるまで、メモリ上のキャッシュを経由します。アプリケーションには書き込み完了が報告されても、ディスク装置にデータが書き込まれる時点でエラーになるとデータは失われます。一般の UNIX ではキャッシュデータの書き込みがエラーになるとシステムパニックになります。

ハードウェア障害が発生したとしてもエラー報告されない場合(このようなことは滅多にありませんが)、ディスク上のデータは破損したまま、アプリケーションが次にそのデータを使用して論理的に検査しない限り、データ破損は判明しないことになります。

ZFS では、アプリケーションから実行された書き込みは一つのトランザクションとして扱われます。データベースシステムのコミットのようなもので、ディスク上の整合性を保つと同時に、エラーが発生した場合で元のデータが壊れるのを防いでいます。この方式を COW(Copy On Write)と呼んでいます。

ZFS では書き込み時のデータ破損が発生しても、チェックサムを計算して不良であることが判明すると、バックアップから正しいデータに修復する機能をもっています。ZFS の自己修復機能です。また、データセキュリティは ACL(Access Control List)によって厳密に管理することができます。

これらの特徴を解説していきます。

2-1. ZFS の特徴
2-1-1. ストレージプール
ストレージプールとファイルシステムは次の関係になります。

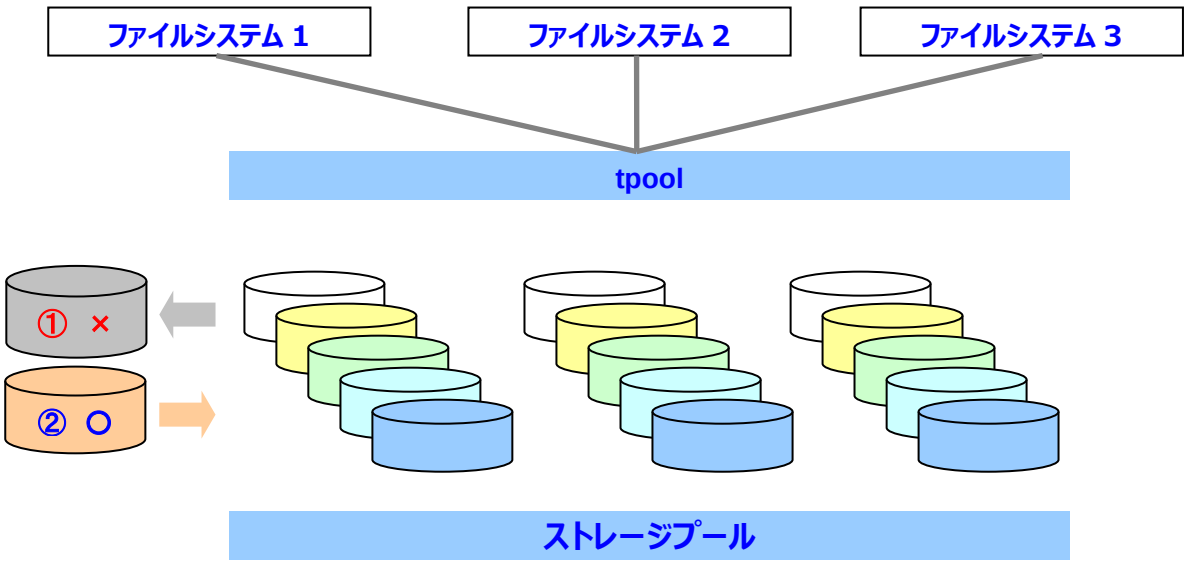


図 2-1-1-1. ストレージプール

ホットスペアをあらかじめ構成しておくと、不良ディスク装置を取り外しできます①。ストレージプールが一杯になった場合、新たなディスク装置を追加することもできます②。

図の例では tpool という名前のストレージプールが 15 個のディスク装置を持っています。ユーザーはその配下にファイルシステムを割り当て、使用することができます。

同一のストレージプールに属するファイルシステム全体で、そのプールの領域を共有します。サイズは要求されただけ、最大のファイルシステムサイズまで使用することができます。使用量を予約して確保する、あるいは、容量を制限することも可能です。

既存のファイルシステムでは、ファイルシステム作成時に i ノードなどのメタデータ領域を割り当てます。ZFS では、ストレージプール内に動的に確保されます。

ZFS ファイルシステムは、トップディレクトリから見て、UNIX のファイルツリーのように作成や削除することができます。ユーザーは、どの物理的なディスク装置にファイルシステムが割り当てられているかを意識することはありません。

次に、OpenSolaris を標準にインストールした場合に作成される、ストレージプールとファイルシステムのツリーを示します。ユーザー“taro”はインストール時に作成するユーザーアカウントの例です。

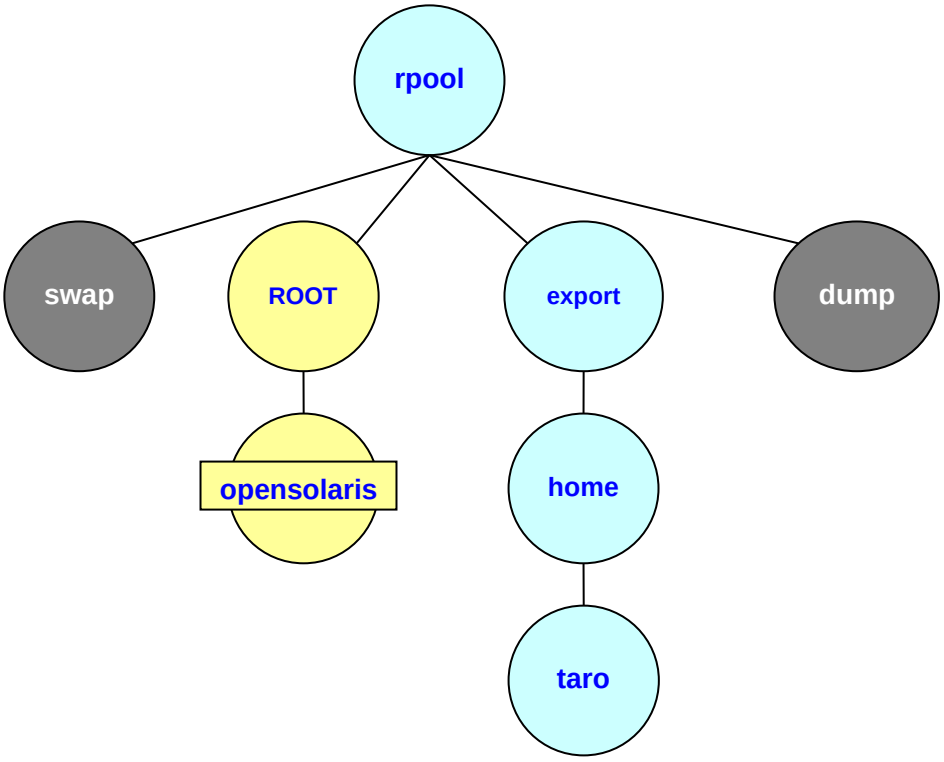


図 2-1-1-2. ストレージプール内のファイルシステム

上記を zfs list コマンドで表示すると次のように表示されます。

```
root@act131:~# zfs list ⏎
NAME                    USED   AVAIL  REFER MOUNTPOINT
rpool                   4.05G  68.8G  77.5K  /rpool
rpool/ROOT              3.04G  68.8G  19K    legacy
rpool/ROOT/opensolaris  3.04G  68.8G  2.90G  /
rpool/dump               511M  68.8G  511M   -
rpool/export            15.7M  68.8G  21K    /export
rpool/export/home       15.7M  68.8G  21K    /export/home
rpool/export/home/taro  15.6M  68.8G  15.6M  /export/home/taro
rpool/swap              512M   69.2G  137M   -
root@act131:~#
```

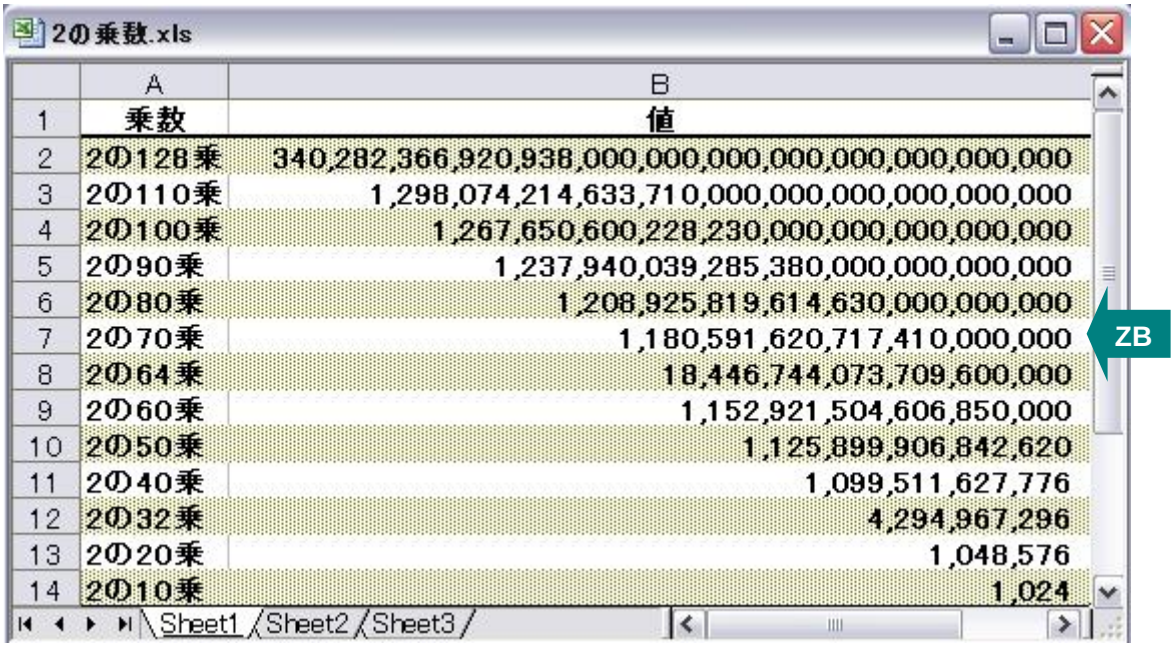
図 2-1-1-3. ストレージプールのファイルシステム

NAME の rpool ファイルシステムは/rpool、rpool/export ファイルシステムは/export、rpool/export/home ファイルシステムは/export/home、rpool/export/home/taro ファイルシステムは/export/home/taro ファイルシステムにマウントされます。これらのマウントされる位置がマウントポイント MOUNTPOINT です。既存のファイルシステムでは/etc/vfstab と呼ばれるマウント情報を持つファイルに、デバイス名と対で指定することで、マウント関係を表していました。rpool/ROOT はレガシー(legacy)のマウント型で、rpool/ROOT/opensolaris のファイルシステム“/”(ルート)にマウントポイントが設定されています。rpool/swap と rpool/dump にマウントポイントはなく、特殊なファイルシステムとして扱われます。

例として、当該ストレージプールを構成したディスク装置は 73GB です。図 2-1-1-3 の AVAIL に表示されている値に注目して下さい。swap 以外のどのファイルシステムも 68.8GB のプールを使用可能であることがわかります。

2-1-2. スケーラビリティ

下図は Microsoft Excel で 2 の乗数を表したものです。セルに"=2^nn"(nn は乗数)と入力して数値プロパティにすると表示できます。矢印で示す ZB がゼタバイト(2 の 70 乗)です。2 の 60 乗を超えたあたりから下の桁に"0"がうまって誤差が大きくなります【備考】。



	A	B
1	乗数	値
2	2の128乗	340,282,366,920,938,000,000,000,000,000,000,000
3	2の110乗	1,298,074,214,633,710,000,000,000,000,000,000
4	2の100乗	1,267,650,600,228,230,000,000,000,000,000
5	2の90乗	1,237,940,039,285,380,000,000,000,000
6	2の80乗	1,208,925,819,614,630,000,000,000
7	2の70乗	1,180,591,620,717,410,000,000
8	2の64乗	18,446,744,073,709,600,000
9	2の60乗	1,152,921,504,606,850,000
10	2の50乗	1,125,899,906,842,620
11	2の40乗	1,099,511,627,776
12	2の32乗	4,294,967,296
13	2の20乗	1,048,576
14	2の10乗	1,024

Sheet1 / Sheet2 / Sheet3

図 2-1-2-1. 2 の乗数と値

ZFS は 128 ビットのファイルシステムと言われていますが、ディスク装置内のポインタは次のような計算で物理ブロックアドレスを求めています。

物理ブロックアドレス(physical block address)=(オフセット offset << 9) + 0x400000(4MB)

オフセットは 64 ビットアドレスです(実際は 63 ビット)。これを 9 ビット左にシフトして、4MB を加えた位置が物理ブロックアドレスになります。4MB を加えるのは、ディスク装置の先頭にある制御ブロックをスキップするためです。

これに加えて、32 ビットの論理デバイス番号を持っています。その論理デバイス番号内の、上記計算で求めた物理ブロックアドレスに目的のデータがセットされます。

ZFS は最初小さい構成でスタートし、順次拡張していくような使い方が有効と考えられます。

【備考】表 1-2-5-2 も参考にしてください。表 1-2-5-2 は拡張倍精度のプログラムで計算した値です。

【参考】2008 年 3 月、EMC 社は、「世界のデジタル情報量は毎年 1.6 倍増加、2011 年には 1 兆 8 千億 GB になる」とのニュー スを発表しました。1 兆 8 千億 GB は 1.8ZB(ゼタバイト)です。2020 年には約 40ZB(ゼタバイト)に達すると予想されています。2011 年、全世界のストレージはたった 2 個の ZFS で収まるっ!? ZFS がいかに大きなスケーラビリティかがわかります(ただし、そのような巨大なハードウェアが存在した場合のお話ですが…)。 ちなみに、2007 年における世界のデジタル情報量は合計 281EB(一人あたり 45GB)で当初の予想を 10%上回ったそうです(News ITpro より引用。下線部分は筆者が加筆しました)。

2-1-3. 管理コマンド

ストレージプールの管理は zpool、プール中のファイルシステム管理は zfs コマンドを使用します。作成、削除、バックアップ、スナップショット、クローンの作成、ディスク装置の置き換えなどのコマンドは、引数が直感的でわかり易く、自由自在に操作できます。

zpool コマンドと zfs コマンドは、引数で与えるファンクション「機能」がたくさんあります。また、ブート環境の管理コマンドが別にあります。とはいえ、とてもわかり易い管理コマンドであるのは間違いありません。

警告や、補完メッセージ(次に何をしたら良いのか)が親切です。また、面倒なファイルシステムフォーマット(ディスク装置のハードウェアフォーマットは除く【備考 1】)や、マウント操作は基本的に不要です。

zpool コマンド：ストレージプールを操作するコマンドです。次の 20 個のファンクション(引数)があります。

add	detach	iostat	replace
attach	export	list	scrub
clear	get	offline	set
create	history	online	status
destroy	import	remove	upgrade

zfs コマンド ：ファイルシステムを操作するコマンドです。次の 20 個のファンクション(引数)があります。

allow	inherit	rename	snapshot
clone	list	rollback	unallow
create	mount	send	unmount
destroy	promote	set	unshare
get	receive	share	upgrade

beadm コマンド ：ブート環境を操作するコマンドです。次の 7 個のファンクション(引数)があります。

activate	destroy	mount	unmount
create	list	rename	

2-1-4. RAID 機能

「2-1-3. 管理コマンド」の zpool コマンドでストレージプールを作成する時、“zpool create”コマンドの引数に **mirror** や **raidz**、**raidz2** を指定すると目的の RAID レベルを選択することができます。引数を指定しない場合は RAID 0 のストライピングになります。

ZFS(RAID 0)の特徴

構成したディスク装置全部をまたがって、まんべんなくストライプされ、ブロックが書き込まれます。ストライプ幅(512 バイト～128KB)は可変長です。create の引数の RAID レベルは無指定でデフォルト RAID 0 相当に構成されます。

ミラー(RAID 1)の特徴

mirror(RAID 1)は通常のミラーと同じです。create の引数に指定し、セットになるデバイスを指定します。

RAID Z(RAID 5 相当)/RAID Z2(RAID 6 相当)の特徴

ZFS(RAID 0)と同様に、動的なストライプ幅(512 バイト～128KB)で書き込まれます。従来の RAID 5/6 では書き込みホール問題がありましたが、これを解消しています【備考 2】。読み込み-変更-書き込みのサイクルは書き込みのみに短縮されており、従来の RAID 5/6 より高速に処理しています。

RAID Z は **raidz**、RAID Z2 は **raidz2** を zpool create コマンドの引数に指定します。

【備考 1】最近は、購入して構成すると論理フォーマットのみで使用することができます。物理フォーマットは滅多に行いません。
【備考 2】全てのストライプとパリティが書き込み終わる前にエラーが発生した場合、パリティとデータの不整合が発生する問題。

2-1-5. データ保全

ZFS はトランザクションタイプのデータコミットを行っています。COW(Copy On Write)はブロックのツリーをコピーして元のブロックを残し、全ての書き込みが終了してからuberblock(ウーバブロック：“oo-ba-block”と発音するようです。ソースコードに記述がありました)と呼ばれる制御ブロックを更新することでコミットを終了します。途中でエラーが発生すると元データが残されているのでデータの破損を防ぐことができます。

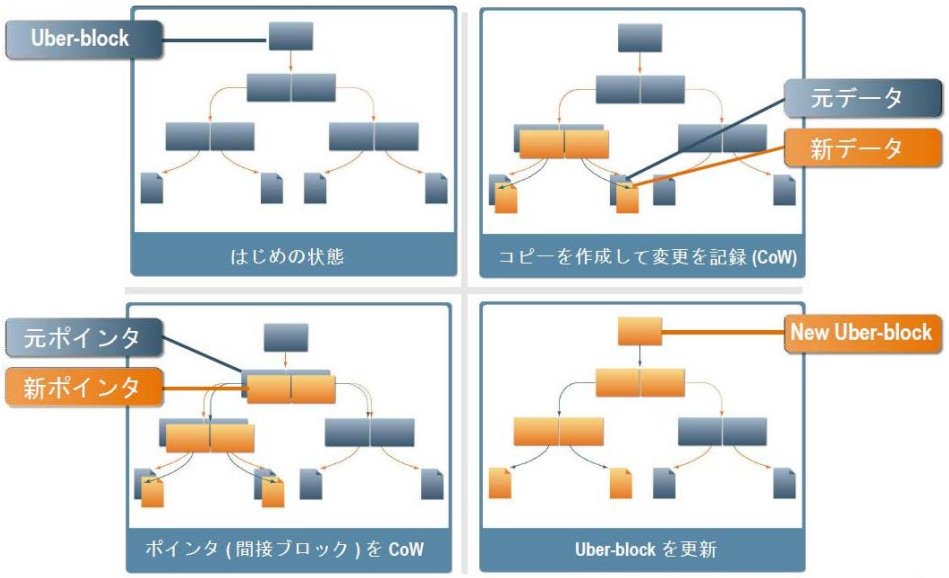


図 2-1-5-1. COW-トランザクションベースの書き込み時コピー

出典：今注目の Solaris ZFS!!(以降 ZFS)っていうかファイルシステムって何?(OSNS-Solaris_ZFS-20090306.pdf)

ZFS は同時に、End-to-End のチェックサムを採取し、uberblock 配下の全ブロックから上程されたチェックサムを最も上位の親ブロックに反映させています。これにより、一貫したデータ保障が可能になっています。

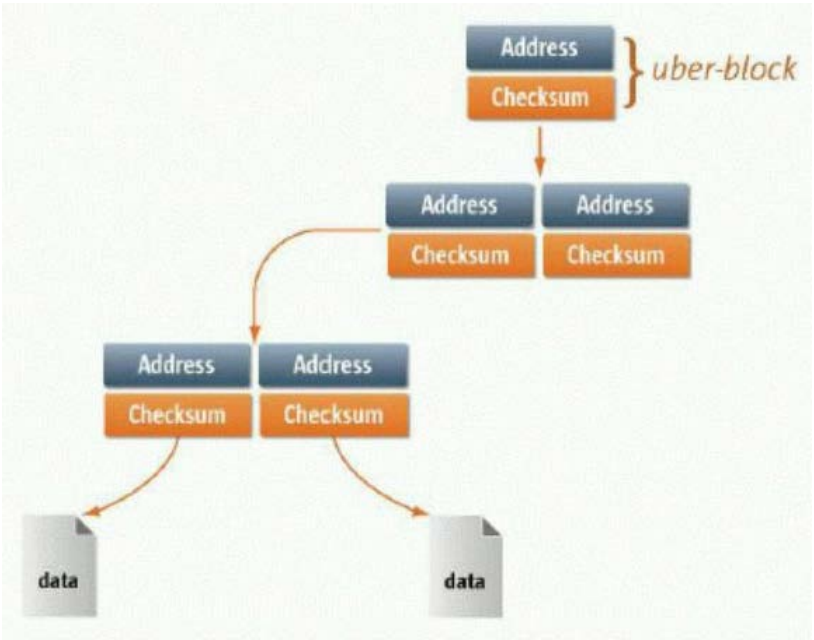


図 2-1-5-2. ZFS のチェックサム

出典：Solaris ZFS の基本的な仕組みを知る(2/3)

ZFS は Ditto Block と呼ばれるデータ複製を持っています。ブロックが壊れていた場合、データ複製を使用して自動的に修復します。

ディスクスクラビングは従来の同期化と似た動作をします。障害の起きたブロックを検出するとともに、Ditto Block からの修復、ミラーからのデータ修復を行います。

スクラビングはバックグラウンドで動作し、停止や開始を手動で行うことも可能です。

ディスク装置単位で、L0、L1、L2、および L3 と呼ばれるメタデータの複製を持っています。どのラベルが破損してもリカバリできるようです。実際に色々なブロックを試しに壊してみたのですが、何事もなかったように動作していました…本当にぃ?!

2-2. ZFS のプロパティ

ZFS ファイルシステムのプロパティを示します。下線(緑色)の強調文字はデフォルトです。rpool/swap や rpool/dump のプロパティには違いがあります。なお、プロパティの並びはアルファベット順にソートしています。

```
# zfs get ⏎
missing property argument
usage:
        get [-rHp] [-o field[,...]] [-s source[,...]]
            <"all" | property[,...]> [filesystem|volume|snapshot]
            ←"all"は詳細リスト出力。
            プロパティ指定できます。

PROPERTY       EDIT    INHERIT    VALUES

aclinherit     YES     YES        discard|noallow|
                                  restricted|passthrough|
                                  passthrough-x
aclmode        YES     YES        discard | groupmask | passthrough
atime          YES     YES        on | off
available      NO      NO         <size>
canmount       YES     NO         on | off | noauto
casesensitivity NO     YES        sensitive | insensitive | mixed
checksum       YES     YES        on | off | fletcher2 | fletcher4 | sha256
compression    YES     YES        on | off | lzjb | gzip | gzip-[1-9]
compressratio  NO      NO         <1.00x or higher if compressed>
copies         YES     YES        1 | 2 | 3
creation       NO      NO         <date>
devices        YES     YES        on | off
exec           YES     YES        on | off
mounted        NO      NO         yes | no
mountpoint     YES     YES        <path> | legacy | none
nbmand         YES     YES        on | off
normalization  NO      YES        none | formC | formD | formKC | formKD
origin         NO      NO         <snapshot>
primarycache   YES     YES        all | none | metadata
quota          YES     NO         <size> | none
readonly       YES     YES        on | off
recordsize     YES     YES        512 to 128k, power of 2
referenced     NO      NO         <size>
refquota       YES     NO         <size> | none
refreservation YES     NO         <size> | none
reservation    YES     NO         <size> | none
secondarycache YES     YES        all | none | metadata
setuid         YES     YES        on | off
shareiscsi     YES     YES        on | off | type=<type>
sharenfs       YES     YES        on | off | share(1M) options
sharesmb       YES     YES        on | off | sharemgr(1M) options
snapdir        YES     YES        hidden | visible
type           NO      NO         filesystem | volume | snapshot
used           NO      NO         <size>
utf8only       NO      YES        on | off
version        YES     NO         1 | 2 | 3 | current
volblocksize   NO      YES        512 to 128k, power of 2
volsize        YES     NO         <size>
vscan          YES     YES        on | off
xattr          YES     YES        on | off
zoned          YES     YES        on | off

Sizes are specified in bytes with standard units such as K, M, G, etc.

User-defined properties can be specified by using a name containing a colon (:).
#
```

2-3. ZFS の ACL

NFSv4 によってファイルシステムを共有する場合、UNIX クライアントと UNIX 以外のクライアント間で相互運用性をサポートする必要があります。このため、ZFS では新しい ACL モデルが適用されました。新しい ACL モデルは次の点で POSIX の ACL と違いがあります。

- NFSv4 仕様に基づきます。NT 方式の ACL に似ています。
- アクセス特権をより詳細に設定できます。
- setfacl や getfacl コマンドは用いません。かわりに chmod と ls コマンドで操作します。
- ディレクトリのアクセス特権をどのようにサブディレクトリに適用するかは ACL の継承に基づきます。

次に ACL を表示する ls コマンドの例を示します。"**-v**"は小文字と大文字の違いがあります【備考】。

```
taro@opensolaris:~/zfsinfo/testdir$ /usr/bin/ls -l ⏎
total 1
-rwxr-xr-x    1 taro     staff          8212 11 月  25 09:01 md6
taro@opensolaris:~/zfsinfo/testdir$ /usr/bin/ls -v ⏎
total 1
-rwxr-xr-x    1 taro     staff          8212 11 月  25 09:01 md64
     0:owner@::deny
     1:owner@:read_data/write_data/append_data/write_xattr/execute
         /write_attributes/write_acl/write_owner:allow
     2:group@:write_data/append_data:deny
     3:group@:read_data/execute:allow
     4:everyone@:write_data/append_data/write_xattr/write_attributes
         /write_acl/write_owner:deny
     5:everyone@:read_data/read_xattr/execute/read_attributes/read_acl
         /synchronize:allow
taro@opensolaris:~/zfsinfo/testdir$ /usr/bin/ls -V ⏎
total 1
-rwxr-xr-x    1 taro     staff          8212 11 月  25 09:01 md64
            owner@:----------------------:-------:deny
            owner@:rwxp---A-W-Co---:-------:allow
            group@:-w-p----------------:-------:deny
            group@:r-x-----------------:-------:allow
         everyone@:-w-p----A-W-Co--:-------:deny
         everyone@:r-x-----a--R--c--s-:-------:allow
taro@opensolaris:~/zfsinfo/testdir$
```

1 行が **ACE** です。（AC エントリ）上から順に評価されます。

A[index]の index 番号

ACL(AC リスト) この表示形式を「非簡易」と呼びます。

ACL(AC リスト) この表示形式を「簡易」と呼びます。

従来のパーミッションとの違いは次のとおりです。

```
r(Read) r  list_directory
   a read_attributes
   R read_xattr
   c read_acl
w(Write) w add_file
   p add_subdirecttory
   A write_attributes
   W  write_xattr
   C write_acl
   o write_owner
x(eXecute) x  eXecute
```

【備考】PATH 環境変数に注意してください。デフォルトの$PATH は、/usr/gnu/bin:/usr/bin:/usr/X11/bin:/usr/sbin:/sbin になっていますので、ls コマンドは GNU のものが使用されます。この場合、"-v"オプションは無効です。"-V"オプションはエラーになります。これらのオプションを使用する場合はフルパスで**/usr/bin/ls** を実行します。

ACL を設定する chmod コマンドの構文は次の通りです【備考】。

非簡易 ACL を設定する構文

【形式 1】
chmod [options] A[index]{+|=}user|group:name:access-permissions
/…[:inheritance-flags]:deny | allow file ⏎
【例】
/usr/bin/chmod A2+user:taro:read_data/append_data/write_data/execute:allow md64 ⏎
【形式 2】
chmod [options] A-user|group:name:access-permissions
/…[:inheritance-flags]:deny | allow file ⏎
【形式 3】
chmod [options] A[index]- file ⏎
【例】
/usr/bin/chmod 755 md64 ⏎

簡易 ACL を設定する構文

【形式 1】
chmod [options] A[index]{+|=}owner@|group@|everyone@:
access-permissions/…[:inheritance-flags]:deny | allow file ⏎
【例】
/usr/bin/chmod A1=owner@:read_data:allow md64 ⏎ ←ACL エントリを入れ替え。
/usr/bin/chmod A1+owner@:read_data:allow md64 ⏎ ←ACL エントリを追加。
【形式 2】
chmod [options] A-owner@, group@, everyone@:
access-permissions/…[:inheritance-flags]:deny | allow file ⏎
【例】
/usr/bin/chmod A=owner@:rwxp---A-W-Co-:----:allow md64 ⏎ ←全 ACL がクリアされこのエントリに入れ替わります。
【形式 3】
chmod [options] A[index]- file ⏎
【例】
/usr/bin/chmod A- md64 ⏎ ←ACL をリセットします。

ACL のエントリタイプを次に示します。

表 2-3-1. ACL のエントリタイプ

No	タイプ	説明
1	owner@	オブジェクトの所有者に許可するアクセス権を指定します。
2	group@	オブジェクトを所有するグループに許可するアクセス権を指定します。
3	everyone@	他のどの AC エントリに一致しない全ユーザー、またはグループに許可するアクセス権を指定します。
4	user	ユーザー名を使ってオブジェクトに追加するユーザーに許可するアクセス権を指定します。ACL-entry-ID を含める必要があります。username、または userID を指定します。値が有効な数値 UID、または username でない場合、その ACL エントリタイプは無効です。
5	group	グループ名を使ってオブジェクトに追加するユーザーに許可するアクセス権を指定します。ACL-entry-ID を含める必要があります。groupname、または groupID を指定します。値が有効な数値 GID、または groupname でない場合、その ACL エントリタイプは無効です。

【備考】PATH 環境変数に注意してください。デフォルトの$PATH は、/usr/gnu/bin:/usr/bin:/usr/X11/bin:/usr/sbin:/sbin になっていますので、chmod コマンドは GNU のものが使用されます。この場合、"A"オプションはエラーになります。これらのオプションを使用する場合は、フルパスで**/usr/bin/chmod** を実行します。

アクセス特権は短縮文字で示されています。次に簡易(コンパクト)表記を示します。

表 2-3-2. ACL アクセス特権の簡易(コンパクト)表記

No	アクセス特権	表記	説明
1	add_file	w	ディレクトリに新しいファイルを追加するためのアクセス権。
2	add_subdirectory	p	ディレクトリ上でサブディレクトリを作成するためのアクセス権。
3	append_data	p	プレースホルダー。現在未実装です。
4	delete	d	ファイルを削除するためのアクセス権。
5	delete_child	D	ディレクトリ内のファイル、またはディレクトリを削除するためのアクセス権。
6	execute	x	ファイルを実行するためのアクセス権。またはディレクトリの内容を検索するためのアクセス権。
7	list_directory	r	ディレクトリの内容を表示するためのアクセス権。
8	read_acl	c	ACL(ls)を読み取るためのアクセス権。
9	read_attributes	a	ACL 以外のファイル基本属性を読み取るためのアクセス権。基本属性は stat レベルの属性です。このアクセスマスクビットを許可したエンティティは、ls コマンド、および stat システムコールを実行できます。
10	read_data	r	ファイルの内容を読み取るためのアクセス権。
11	read_xattr	R	ファイルの拡張属性を読み取るためのアクセス権。または、ファイルの拡張属性ディレクトリの検索を次項するためのアクセス権。
12	synchronize	s	プレースホルダー。現在未実装です。
13	write_xattr	W	拡張属性を作成するためのアクセス権。または拡張属性ディレクトリに書き込むためのアクセス権。このアクセス権を許可したユーザーは、ファイルの拡張属性ディレクトリを作成できます。属性ファイルのアクセス権を使って、その属性にユーザーがアクセスできるかどうかを制御します。
14	write_data	w	ファイルの内容を変更、または書き換えるためのアクセス権。
15	write_attributes	A	ファイル、またはディレクトリに関連付けられた時刻を任意の値に変更するためのアクセス権。
16	write_acl	C	ACL を書き込むためのアクセス権。 chmod コマンドを使用して ACL を変更することができます。
17	write_owner	o	ファイルの所有者、またはグループを変更するためのアクセス権。ファイルに対して chmod、または chgrp コマンドを実行することができます。ファイルの所有者を取得するためのアクセス権。または、ファイルのグループ所有権をユーザーが所属するグループに変更するためのアクセス権。ファイルまたはグループの所有権を任意のユーザー、またはグループに変更する場合は、PREIV FILE CHOWN 権限が必要です。

ACL 継承を使用する目的は、親ディレクトリの既存のアクセス権ビットを考慮して、意図した ACL を新たに作成したファイル、またはディレクトリに引き継ぐためのものです。

デフォルトでは ACL は継承されません。ACL を継承する場合は、ファイル、またはディレクトリにその旨をセットする必要があります。次にオプションの継承フラグを示します。

表 2-3-3. ACL 継承フラグ

No	継承フラグ	表記	説明
1	file inherit	f	親ディレクトリの ACL をそのディレクトリのファイルにのみ継承します。
2	dir inherit	d	親ディレクトリの ACL をそのディレクトリのサブディレクトリにのみ継承します。
3	inherit only	i	親ディレクトリから ACL を継承しますが、新しく作成したファイル、またはディレクトリにのみ適用され、そのディレクトリ自体には適用されません。このフラグを使用する場合は、何を継承するかを指定するために、file inherit フラグ、あるいは両方を指定する必要があります。
4	no propagate	n	親ディレクトリの ACL をそのディレクトリの第 1 レベルの内容にのみ継承します。第 2 レベル以降には継承しません。このフラグを使用する場合は、何を継承するかを指定するために、file inherit フラグ、または dir inherit フラグ、またはその両方を指定する必要があります。
5	-	-	アクセス権は付与されていません。

ZFS ファイルシステムでは ACL に関するプロパティが 2 つあります。

- **aclinherit** このプロパティを使って、ACL 継承の動作を決定します。次の値を使用できます。
  - **discard** 新しいオブジェクトの場合に、ファイル、またはディレクトリを作成する時に ACL エントリは継承されません。ファイル、またはディレクトリの ACL は、そのファイル、またはディレクトリのアクセス権モードと同じです。
  - **noallow** 新しいオブジェクトの場合に、継承可能な ACL エントリのうち、アクセスタイプが deny のエントリだけが継承されます。
  - **restricted** 新しいオブジェクトの場合に、ACL エントリが継承されるとき、write_owner と write_acl アクセス権が取り除かれます。
  - **passthrough** プロパティの値が passthrough に設定されている場合、作成されるファイルのモードは継承可能な ACE によって決定されます。モードに影響を与える継承可能な ACE が無い場合、モードはアプリケーションから要求されたモードに従って設定されます。
  - **passthrough-x** 動きは次の点を除き passthrough と同じです。passthrough-x を有効にした場合、ファイル作成モード、およびモードに影響を与える継承可能な ACE で実行アクセス権が設定されている場合に限り、ファイルが実行(x)アクセス権付きで作成されます。

  ※aclinherit のデフォルトは **restricted** です。
- **aclmode** このプロパティを指定すると、ファイルが最初に作成されるとき、または、chmod コマンドを使ってファイル、またはディレクトリのモードが変更されるときに、ACL の動作が変更されます。次の値を使用できます。
  - **discard** ファイル、またはディレクトリのモードを定義するために必要なエントリを除いて、全ての ACL が取り除かれます。
  - **groupmask** ユーザー、またはグループの ACL アクセス権の強度が、グループのアクセス権ビットより弱くなるように変更されます。ただし、そのファイル、またはディレクトリの所有者と同じ UID を持つユーザーエントリは変更されません。さらに、その ACL アクセス権の強度が、所有者のアクセス権ビットより弱くなるように変更されます。
  - **passthrough** owner@、group@、または everyone@以外の ACE は、chmod 操作時に変更されることはありません。owner@、group@、または everyone@の ACE は無効になり、chmod コマンドで操作された要求モードが設定されます。

  ※aclmode のデフォルトモードは、**groupmask** です。

2-4. ZFS 委任管理

root ユーザー以外の一般ユーザーに対して、細かいアクセス権を割り当てることができます。

大きく 2 つの委任アクセス権があります。

- 作成、削除、マウント、スナップショットのような個別のアクセス権を指定できます。
- たくさんのアクセス権をまとめて、「アクセス権セット」と呼ばれるアクセス権の集まりを定義することができます。アクセス権セットはあとで更新することができ、そのセットの使用者は自動的に変更内容を取得します。アクセス権セットは 64 文字以下の長さの@文字で始まるセット名を指定します。

ZFS 委任管理では RBAC セキュリティモデルに似た機能が提供されます。ZFS 委任モデルを使用すると、ZFS ストレージプール、およびファイルシステム管理に次のような利点が得られます。

- ZFS ストレージプールの移行時、アクセス権も移行されます。
- 動的継承により、ファイルシステム間でアクセス権をどのように伝達するかを制御できます。
- ファイルシステムの作成者だけがそのファイルシステムを削除できるように設定することができます。
- アクセス権を特定のファイルシステムに割り当てることができます。新しく作成されるファイルシステムは、アクセス権を自動的に取得できます。
- ZFS 委任モデルによって、単純な NFS 管理が提供されます。たとえば、明示的なアクセス権を持っているユーザーは、NFS 経由でスナップショットを作成し、適切な./zfs/snapshot ディレクトリに保存できます。

ZFS 委任アクセス権を無効にする

```
# zpool get delegation tpool ⏎
NAME   PROPERTY     VALUE        SOURCE
tpool  delegation   on           default
# zpool set delegation=off tpool ⏎
# zpool get delegation tpool ⏎
NAME   PROPERTY     VALUE        SOURCE
tpool  delegation   off          default
#
```

ZFS 委任アクセス権を委任する

zfs allow コマンドを使用してアクセス権を委任します。

```
# zfs allow taro create,destroy,mount,snapshot tpool/test ⏎
# zfs allow tpool/test ⏎
-------------------------------------------------------------
Local+Descendent permissions on (tpool/test)
        user taro create,destroy,mount,snapshot
-------------------------------------------------------------
#
```

2-5. 高速化のための ZFS 技術

ZFS では高速化のために様々な技術が取り入れられています。

(1) パイプライン接続エンジン
相互依存性と優先順位に基づいて入出力をソートし、24 段パイプラインを経由してアクセスします。

(2) 先読みアルゴリズム
ARC や L2ARC に、アクセスパターン分析による推論を行い、隣接ブロックを読み込みます。

(3) 並行処理
並列にディレクトリやファイルの読み書きを行います。

(4) トランザクションの最適化
ディスク動作を最適化します。

(5) 組込データ圧縮
組み込みデータを圧縮することで、入出力操作を削減します。

(6) ログ機構のキャッシュ
書き込みログを SSD や NVRAM に置くことで高速化することが可能です。

なお、第 5 章に実測の性能情報を解説します。ご覧ください。

2-6. 他 OS のリリース状況

Oracle(Sun Microsystems)から OpenSolaris プロジェクトへソースコードが公開され、それを機に他の OS への移植が進んでいます。CDDL(Common Development and Distribution License)のもと、オープンソースで開発されています。

次に他 OS のリリース状況を示します。

Linux	ZFS on Linux のプロジェクトのもと、RedHat Enterprise Linux、Debian、CentOS、Ubuntu、Fedora などのディストリビューションでリリースされています。CDDL は GPL に抵触するという問題があり、Linux カーネルには含まれません。
MacOS	10.5 Leopard より実装されています。その後、アップデートは進められていませんが、2020 年に OpenZFS on OS X 2.0 がリリースされました。
FreeBSD	x64 9.0-RELEASE で v28 がサポートされています。
NetBSD	2007 年の Google Summer of Code で開発が始められましたが、2016 年時点でメンテナンスされていません。

第 3 章 ディスク管理と ZFS 操作

3-1. ZFS の構成と機能

ZFS の構成や代表的な機能を紹介します。

3-1-1. ストレージプールを操作する zpool コマンド

(1) ストレージプールの状態を表示する zpool コマンド

```
# zpool list ⏎                  プールリストを表示
# zpool status ⏎                プールの状態を表示
# zpool get all ⏎               全部のプールプロパティを表示
# zpool get all tpool ⏎         特定のプールプロパティを表示
# zpool get ⏎                   提供されているプロパティリストを表示
```

(2) ストレージプールを操作する zpool コマンド

ストレージプールの作成

```
# zpool create tpool c1t0d0 c1t2d0 ⏎                  ZFS ストレージプール(RAID 0 相当)作成
# zpool create tpool mirror c1t0d0 c1t2d0 ⏎           RAID 1(ミラー)のストレージプール作成
# zpool create tpool raidz c1t0d0 c1t2d0 ⏎            RAID Z(RAID 5 相当)を作成
# zpool create tpool raidz2 c1t0d0 c1t2d0 ⏎           RAID Z2(RAID 6 相当)を作成
# zpool create -o version=10 tpool c1t0d0 c1t2d0 ⏎    古いバージョンのプールを作成
```

引数に指定する“c1t0d0”のようなデバイス装置名は format コマンドで表示されたデバイスをそのまま記述します。スライス番号でも可能ですが、参考文献ではデバイス装置名を割り当てることが推奨されています。

ホットスペアは“**spare**”をデバイス装置名の前に指定します。装置名の表示は、次のように format の出力を調べて使用します。USB デバイスは“format -e”のように、“**-e**”オプションを付けると表示することができます。

```
root@opensolaris:~# format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c8t0d0 <DEFAULT cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@0,0
       1. c8t1d0 <DEFAULT cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@1,0
       2. c8t2d0 <DEFAULT cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@2,0
       3. c8t3d0 <SUN72G cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@3,0
       4. c8t4d0 <SUN72G cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@4,0
       5. c8t5d0 <SUN72G cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@5,0
Specify disk (enter its number): Ctrl-C
root@opensolaris:~#
```

図 3-1-1. format コマンドの出力例

ストレージプールの削除

```
# zpool destroy tpool ⏎
```

デバイスの追加

```
# zpool add tpool c1t3do ⏎
```

ストレージプールのエクスポート

```
# zpool export tpool ⏎
```

ストレージプールのインポート

```
# zpool import tpool ⏎
```

他に各種のファンクションがあります。付録 A-1 をご覧ください。

3-1-2. ファイルシステムを操作する zfs コマンド

(1) ファイルシステムを表示する zfs コマンド

```
# zfs list ⏎                  ファイルシステムを表示
# zfs get all ⏎               全ファイルシステムのプロパティ値を表示
# zfs get all rpool ⏎         指定ファイルシステムプロパティ値を表示
# zfs list -t snapshot ⏎      スナップショットの表示
```

(2) ファイルシステムを操作する zfs コマンド

ファイルシステムの作成

```
# zfs create tpool/home ⏎
```

ファイルシステムの削除

```
# zfs destroy tpool/home ⏎
```

スナップショットの作成

```
# zfs snapshot tpool/home/taro@s2 ⏎
```

スナップショットのロールバック

```
# zfs rollback tpool/home/taro@s1 ⏎
```

クローンの作成

```
# zfs clone tpool/export@orig tpool/export/taro/current ⏎
```

ファイルシステムの遠隔バックアップ

```
# zfs send tpool/export/home | ssh taro@hostA dd of=/backup/20091124 ⏎
# zfs send tpool/export/home | ssh taro@hostA zfs receive -d
/upool/export/home ⏎
```

“>”と、dd コマンドを組み合わせて自システムへバックアップすることも可能です。

ファイルシステムの遠隔からのリストア

```
# ssh taro@hostA dd if=/backup/20091124 | zfs receive -d
/tpool/export/home ⏎
# ssh taro@hostB zfs send /upool/export/home | zfs receive -d
/tpool/export/home ⏎
```

dd や cat コマンドと“|”を組み合わせて自システムのバックアップからリストアすることも可能です。

ファイルシステムのプロパティをセットする

```
# zfs set sharenfs=rw tpool/nfs ⏎
# zfs set quoata=10g tpool/export/home ⏎
# zfs set reservation=20g tpool/export/home ⏎
```

他に各種のファンクションがあります。付録 A-2 をご覧ください。

3-1-3. ブート環境を操作する beadm コマンド

(1) ブート環境を操作する beadm コマンド

BE を作成する

```
# beadm create BE20091125 ⏎
```

BE を削除する

```
# beadm destroy BE20091125 ⏎
```

他に各種のファンクションがあります。付録 A-3 をご覧ください。

3-2. ZFS プールと ZFS ファイルシステム

3-2-1. ストレージプールの操作

はじめにストレージ要件を確認して、ストレージプールを作成します。プールによって、ストレージの物理的な特性が決まります。どのようなファイルシステムを作成する場合でも、はじめにストレージプールを作成する必要があります。

3-2-1-1. ミラー化されたストレージプールの作成

ZFS のミラー構成は、複数のドライブを一つにまとめた仮想ドライブを作成し、これを使用してストレージプールを作成します。ミラーリングに使用するドライブは、同じ容量、かつ同じモデルでなければいけません。以下の例では、<デバイス 1>と<デバイス 2>のドライブをミラー化したものを使用して、ストレージプールを作成しています。使用可能容量は、小さい方の容量になります。

構文 1：ミラー化されたストレージプールの作成
zpool create <ストレージプール名> mirror <デバイス 1> <デバイス 2>

実行例 1：ドライブ 2 台をミラーリングして、1 つのストレージプールを作成します。
root@opensolaris:~# **zpool create tpool mirror c9t0d0 c9t1d0** ⏎

3-2-1-2. RAID Z 構成のストレージプールの作成

RAID Z は、パリティを付加することによって 1 台のドライブが故障しても、データを復元することができます。OpenSolaris 2008.11 以降から使用できるようになった RAID Z2 では、演算式の異なる 2 つのパリティをそれぞれ 2 つのドライブに記録することにより、1 つまたは 2 つのドライブが故障したとしても、データを復元することができます。

RAID Z を構成するには、RAID Z で 2 台以上、RAID Z2 で 3 台以上のドライブを一つにまとめた仮想ドライブを作成し、これを使用してストレージプールを作成します。RAID Z は、データの保存とそのデータに対応するパリティを保存するドライブが別になるように動的に割り当てられています。そのため、2 台のドライブからパリティチェックを行うことができます。

使用可能容量は、すべてのドライブの総容量から、パリティに必要な容量を差し引いた容量になります。

構文 1：RAID Z の作成
zpool create <ストレージプール名> raidz <デバイス 1> <デバイス 2> ・・・・・・・・・・ <デバイス n>
構文 2：RAID Z2 の作成
zpool create <ストレージプール名> raidz2 <デバイス 1> <デバイス 2> ・・・・・・・・・・ <デバイス n>

実行例 1：ドライブ 2 台を RAID Z 構成して、1 つのストレージプールを作成します。
root@opensolaris:~# **zpool create tpool raidz c9t0d0 c9t1d0** ⏎
実行例 2：ドライブ 4 台を RAID Z2 構成して、1 つのストレージプールを作成します。
root@opensolaris:~# **zpool create tpool raidz2 c9t0d0 c9t1d0 c9t2d0 c9t3d0** ⏎

3-2-1-3. ストレージプールへのドライブの追加

ZFS では稼働中のファイルシステムに影響を及ぼすことなくディスクドライブをストレージプールに追加することができます。ドライブをストレージプールに追加するには、作成時と同じような構文になり、コマンドは“zpool add”を使用します。

なお、OS のブートに使用しているストレージプール(rpool)にドライブを追加することはできません。

“-f”オプションは、使用中と表示されていた、または、競合する複製レベルが指定されていたとしても、デバイスの使用を強制します。この方法で、すべてのデバイスを上書きできるわけではありません。“-n”オプションは、デバイスを実際に追加することなく、使用される設定を表示します。

構文 1：ストレージプールへのドライブの追加
zpool add [-fn] <ストレージプール名> <デバイス 1> <デバイス 2> ・・・・・・・・・・ <デバイス n>

実行例 1：ストレージプールにドライブを 1 台追加します。
root@opensolaris:~# **zpool add tpool c9t5d0** ⏎

3-2-1-4. ホットスワップ用のドライブの追加

ドライブの故障時に利用するホットスペアのドライブを追加することができます。ストレージプール内のドライブが故障した場合に、ホットスペアのドライブが代行ドライブとして機能し、故障したドライブは一時停止状態として表示されます。ホットスペアを追加するには、“zpool add”コマンドに“spare”オプションを使用します。また、ホットスペアのドライブは、複数のストレージプールで共有することができます。ディスクドライブに異常があった場合、ストレージプールの状態が“Degraded”になります。

以前の状態に戻すには、異常のあったドライブを交換して“Online”に戻したあと、スペアをプールから“zpool detach”コマンドで切り離します。また、スペアドライブをミラー本体として動作させたい場合は、異常のあったドライブをプールから“zpool detach”コマンドで切り離します。

構文 1：ホットスワップ用のドライブの追加
zpool add <ストレージプール名> spare <デバイス>
構文 2：ドライブをオンラインにする
zpool online <ストレージプール名> <デバイス>
構文 3：ドライブの切り離し
zpool detach <ストレージプール名> <デバイス>

実行例 1：ストレージプールにスペアディスクを追加します。
root@opensolaris:~# **zpool add tpool spare c9t6d0** ⏎
実行例 2：ドライブをオンラインにします。
root@opensolaris:~# **zpool online tpool c9t3d0** ⏎
実行例 3：ドライブを切り離します。
root@opensolaris:~# **zpool detach tpool c9t0d0** ⏎

3-2-1-5. ストレージプールの空き容量の確認

ZFS では、ストレージプールの容量をその中のファイルシステム間で分け合うため、ストレージプールの情報を使用して、空き容量を確認します。ZFS ストレージプールとその使用容量を表示させるには、“zfs list”コマンドを使用します。

構文 1：ストレージプールの空き容量の確認
zpool list
構文 2：ストレージプールとその使用容量の確認
zfs list

実行例 1：ストレージプールの空き容量を確認します。
root@opensolaris:~# **zpool list** ⏎
```
NAME      SIZE     USED    AVAIL     CAP   HEALTH    ALTROOT
rpool     8.94G    3.54G   5.40G     39%   ONLINE    -
tpool     11.9G    2.04M   11.9G      0%   ONLINE    -
```
実行例 2：ストレージプールとその使用容量を確認します。
root@opensolaris:~# **zfs list** ⏎
```
NAME                    USED   AVAIL   REFER   MOUNTPOINT
rpool                   3.90G   4.90G   77.5K   /rpool
rpool/ROOT              3.01G   4.90G   19K     legacy
rpool/ROOT/opensolaris  3.01G   4.90G    2.87G  /
rpool/dump              274M   4.90G    274M   -
rpool/export            125M   4.90G    21K    /export
rpool/export/home       125M   4.90G    21K    /export/home
rpool/export/home/taro  125M   4.90G    125M   /export/home/taro
rpool/swap              512M   5.30G    101M   -
tpool                   152K   7.79G    38.0K  /tpool
```

3-2-1-6. 動作の統計情報を表示する

“zpool iostat”コマンドを使用して、ストレージプールの入出力トランザクション数(operations)と帯域幅(bandwidth)を表示します。

構文 1：ストレージプールの動作の統計情報を表示
zpool iostat [<ストレージプール名>] [<間隔> [<回数>]]

実行例 1：ストレージプールの稼動統計情報の表示
root@opensolaris:~# **zpool iostat** ⏎
```
               capacity     operations    bandwidth
pool        used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
rpool       4.26G  4.68G     17      4   368K  39.0K
tpool       77.5K  3.97G      0      2   118   2.19K
----------  -----  -----  -----  -----  -----  -----
```
実行例 2：特定のストレージプールの稼動統計情報を 3 秒ごとに 5 回表示させます。
root@opensolaris:~# **zpool iostat tpool 3 5** ⏎
```
               capacity     operations    bandwidth
pool        used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
tpool       77.5K  3.97G      0      0     48    911
tpool       77.5K  3.97G      0      0      0      0
tpool       77.5K  3.97G      0      0      0      0
tpool       77.5K  3.97G      0      0      0      0
tpool       77.5K  3.97G      0      0      0      0
```

3-2-1-7. エラーの有無やストレージプールの構成の確認

“zpool status”コマンドを使用して、ストレージプールの動作状況や、構成状況を確認します。“-x”オプションを指定すると、異常のあるストレージプールに関する情報のみが表示され、“-v”オプションを指定すると、詳細なエラー情報を表示します。

構文 1：ストレージプールの状況確認
zpool status [-vx] [<ストレージプール名>]

実行例 1：エラー情報を確認します。
```
root@opensolaris:~# zpool status tpool ⏎
  pool : tpool
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
   the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see: http://www.sun.com/msg/ZFS-8000-2Q
 scrub : none requested
config:

        NAME  STATE             READ WRITE CKSUM
        tpool  DEGRADED         0       0       0
          mirror ONLINE         0       0       0
            c9t0d0  ONLINE      0       0       0
            c9t1d0  ONLINE      0       0       0
          mirror DEGRADED       0       0       0
            c9t2d0  UNAVAIL     0       0       0   cannot open
            c9t3d0  ONLINE      0       0       0

errors: No known data errors
```

3-2-1-8. 強制的にデータをチェックする

ZFS はすべてのデータにチェックサムが付けられ、エラーデータを検出できるようになっていますが、エラーの検出はデータを読み出したタイミングになります。“zpool scrub”コマンドでは、強制的にすべてのチェックサムを確認して、エラーの有無を確認することができます。ミラーや RAID Z が構成されている場合に、ドライブでエラーが検出されると、自動的に再同期が行われます。“scrub”は時間がかかる処理であるため、同時に 1 つしか実行されません。“scrub”の進行状況は“status”コマンドで確認することができます。“-s”オプションで、進行中の“scrub”処理を中断することができます。

構文 1：データとチェックサムの整合性を強制的にチェックする
zpool scrub [-s] <ストレージプール名>

実行例 1：scrub を実行して、そのステータスを表示します。
```
root@opensolaris:~# zpool scrub rpool ⏎
root@opensolaris:~# zpool status rpool ⏎
  pool : rpool
 state: ONLINE
 scrub : scrub in progress for 0h0m, 1.00% done, 0h26m to go
config:

        NAME  STATE                 READ WRITE CKSUM
```

```
        rpool  ONLINE                0       0       0
          mirror ONLINE             0       0       0
            c7d0s0   ONLINE         0       0       0
            c7d1s0   ONLINE         0       0       0

errors: No known data errors
```

3-2-1-9. 異常が発生したドライブの交換

ZFS ではエラーが発生しても自己修復しますが、ドライブにエラーが頻発する場合、故障の兆しであることが多いため、新しいドライブに交換しておくことが大切です。

ドライブを交換するには“zpool replace”コマンドを使用します。

構文 1：ストレージプールに属しているドライブの交換

```
# zpool replace [-f] <ストレージプール名> <デバイス> <新デバイス>
```

実行例 1：ストレージプールに属しているドライブを交換します。

```
root@opensolaris:~# zpool replace tpool c9t2d0 c9t4d0 ⏎
```

3-2-1-10. 別システムにストレージを接続する

ZFS はデバイスを固有のデバイス ID で管理しているため、ドライブを別のシステムにつなぎ替えたり、CPU を交換する場合や、ドライブが接続されているデバイス ID を変更する場合には、あらかじめ“zfs export”コマンドを使用して、エクスポートする必要があります。エクスポートしたストレージプールは使用不可能になります。“zfs export”コマンドは、“-f”オプションを指定することによって、強制的にインポートできます。

新しいシステムでは、“zfs import”コマンドを使用してドライブをスキャンし、エクスポートされている使用可能なストレージプールを見つけてシステムに登録すると、そのストレージプールは使用可能になります。引数のない“zfs import”コマンドを実行すると、“/dev/dsk”ディレクトリをスキャンし、インポート可能なストレージプールを表示します。“-d”オプションを指定すると、別のディレクトリからドライブをスキャンします。また、“-o”オプションを指定して、マウントオプションを同時に指定することもできます。

構文 1：ストレージプールをエクスポート

```
# zpool export [-f] <ストレージプール名>
```

構文 2：インポート可能なストレージプールの一覧を表示

```
# zpool import [-d <ディレクトリ名>] [-D]
```

構文 3：ストレージプールをインポート

```
# zpool import
```

実行例 1：ストレージプール“tpool”をエクスポートします。

```
root@opensolaris:~# zpool export tpool ⏎
```

実行例 2：インポート可能なストレージプールを“/dev/dsk”からスキャンします。

```
root@opensolaris:~# zpool import ⏎
   pool : tpool
     id: 17415058951999357010
  state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        tpool  ONLINE
```

```
        raidz2  ONLINE
          c9t0d0  ONLINE
          c9t1d0  ONLINE
          c9t2d0  ONLINE
          c9t3d0  ONLINE
```

実行例 3：ID を指定して、別の名前を付けてストレージプールをインポートします。

root@opensolaris:~# **zpool import 17415058951999357010 tpool2** ⏎

3-2-2. ファイルシステムの操作

3-2-2-1. ファイルシステムの作成

ZFS では、複数のファイルシステムを木構造にまとめて 1 つのファイルシステムに構成することが可能です。また、上位のファイルシステムのプロパティを子孫ファイルシステムに継承させることができます。ファイルシステムは事前に作成しておいたストレージプールから作成します。デフォルトでは、ファイルシステムの作成時に自動的にマウントされます。“zfs create”コマンドを使用して、ファイルシステムを作成します。削除する場合は“zfs destroy”コマンドを使用します。

ZFS では、階層構造を持ったファイルシステムを 1 つのコマンドでまとめて作成できます。たとえば、ファイルシステム“tpool/home/test/data”を作成することによって、“tpool/home”と“tpool/home/test”も一度にまとめて作成することができます。このため、多数のファイルシステムからなる複雑な階層構造のファイルシステムの構成・運用も容易になっています。

通常、“tpool/home/test/data”というファイルシステムをマウントポイントの指定なしで作成したとき、自動的にファイルシステム名の先頭に“/”(スラッシュ)の付いた、“/tpool/home/test/data”というマウントポイントになります。また、ファイルシステムを作成する時点で、別のマウントポイントを指定することも可能です。

構文 1：ファイルシステムの作成
zfs create <ファイルシステム名>
構文 2：階層構造のファイルシステムをまとめて作成
zfs create -p <ファイルシステム名>
構文 3：階層構造のファイルシステムの表示
zfs list -r <ファイルシステム名>
構文 4：マウントポイントを指定したファイルシステムの作成
zfs create -o mountpoint=<マウントポイント> <ファイルシステム名>
構文 5：ファイルシステムの破棄
zfs destroy <ファイルシステム名>

実行例 1：ストレージプール“tpool”からファイルシステムを作成します。
root@opensolaris:~# **zfs create tpool/home** ⏎
実行例 2：階層構造のファイルシステムをまとめて作成して、確認します。
root@opensolaris:~# **zfs create -p tpool/home/test/data** ⏎
root@opensolaris:~# **zfs list -r tpool** ⏎

NAME	USED	AVAIL	REFER	MOUNTPOINT
tpool	170K	3.91G	21K	/tpool
tpool/home	**59K**	**3.91G**	**21K**	**/tpool/home**
tpool/home/test	**38K**	**3.91G**	**19K**	**/tpool/home/test**
tpool/home/test/data	19K	3.91G	19K	/tpool/home/test/data

実行例 3：マウントポイントを“/test”に指定したファイルシステムを作成して、確認します。
root@opensolaris:~# **zfs create -o mountpoint=/test tpool/home** ⏎
root@opensolaris:~# **zfs list -r tpool** ⏎

NAME	USED	AVAIL	REFER	MOUNTPOINT
tpool	130K	3.91G	19K	/tpool
tpool/home	**19K**	**3.91G**	**19K**	**/test**

3-2-2-2. マウントポイントの変更

ZFS のファイルシステム管理は、SMF サービス svc:/system/filesystem/local で管理されているため、“vfstab(4)”への記入なしでもマウントが可能です。

SMF サービスでの管理方法ではなく、従来のファイルシステムと同様の管理方法で管理したい場合、“mountpoint”プロパティを“legacy”に設定します。“vfstab(4)”への設定によってマウントポイントを指定できます。

構文 1：マウントポイントの変更
zfs set mountpoint=<変更先マウントポイント> <ファイルシステム名>
構文 2-1：従来のファイルシステムと同じ方法での設定
zfs set mountpoint=legacy <ファイルシステム名>
mount -F <ファイルシステム> <ファイルシステム名> <マウントポイント>
構文 2-2：/etc/vfstab の設定
#device device mount FS fsckmount mount
#to mount to fsck point type pass at bootoptions
#
/devices - /devices devfs - no -
[ファイルシステム名] - [マウントポイント] ufs - [yes|no] -

実行例 1：ファイルシステムのマウントポイントを変更します。
root@opensolaris:~# **zfs set mountpoint=/tpoolx/home/test/data tpool/home/test/data** ⏎
実行例 2-1：従来のファイルシステムと同じように設定します。
root@opensolaris:~# **zfs set mountpoint=legacy tpool/home/test/data** ⏎
root@opensolaris:~# **mount -F ufs tpool/home/test/data /data** ⏎
実行例 2-2：vi エディタで、“/etc/vfstab”を設定します。
#device device mount FS fsckmount mount
#to mount to fsck pointtype pass at bootoptions
#
/devices -/devices devfs -no -
tpool/home/test/data -/dataufs -yes -

3-2-2-3. プロパティの設定

ZFS は、ファイルシステムそれぞれにプロパティを設定して、ファイルシステムの動作を制御できます。“zfs get <プロパティ>”コマンドに“all”を指定することで、一覧を表示することができます。

プロパティの値はファイルシステム作成時に設定することができます。また、作成後“zfs set”で変更することもできます。

親ファイルシステムのプロパティを変更することによって、子ファイルシステムのプロパティも自動的に継承されます。自動的に継承された子ファイルシステムのプロパティは、子ファイルシステムのプロパティを明示的にプロパティセットすることによって変更できます。また、子ファイルシステムで変更を加えたプロパティは、“zfs inherit”コマンドを使用することによって親ファイルシステムのプロパティを継承することができます。“-r”オプションを指定すると、子孫のファイルシステムに対して同じようにプロパティを継承させることができます。

構文 1：プロパティ値の取得
zfs get <プロパティ> <ファイルシステム名>
構文 2：プロパティ値の変更
zfs set <プロパティ>=<値> <ファイルシステム名>
構文 3：親ファイルシステムのプロパティを継承する
zfs inherit [-r] <プロパティ> <ファイルシステム名>

実行例 1：ファイルシステム“tpool/home”のプロパティ“compression(圧縮)”を確認します。
root@opensolaris:~# **zfs get compression tpool/home** ⏎
NAME PROPERTY VALUE SOURCE
tpool/home/test/data compression off default
実行例 2：ファイルシステム“tpool/home”のプロパティ“compression(圧縮)”を“on”にします。
root@opensolaris:~# **zfs set compression=on tpool/home** ⏎
実行例 3：“tpool/home”のプロパティを、子ファイルシステムである“tpool/home/test”に継承します。
root@opensolaris:~# **zfs inherit compression tpool/home/test** ⏎

3-2-2-4. ファイルシステムのマウント

ZFS でファイルシステムを作成すると自動的にマウントされますが、システムの管理状況によってアンマウントして使用できないようにする場合に、“zfs unmount”コマンドを使用します。また、アンマウントしたファイルシステムをマウントするには、“zfs mount”コマンドを使用します。

ZFS ファイルシステムがマウントされているかどうかは、プロパティ“mounted”の値で確認することができます。また、“mount”コマンドで確認することもできます。

構文 1-1：プロパティ“mounted”でマウントされているかを確認
zfs get -r mounted <ファイルシステム名>
構文 1-2：“mount”コマンドでマウントされているかを確認
mount -p | grep -i <ファイルシステム名>
構文 2：ZFS ファイルシステムをマウントする
zfs mount <ファイルシステム名>
構文 3：ZFS ファイルシステムをアンマウントする
zfs unmount <ファイルシステム名>

実行例 1-1：ファイルシステム“tpool”以下がマウントされているかを、プロパティ“mounted”で確認します。
root@opensolaris:~# **zfs get -r mounted tpool** ⏎

NAME	PROPERTY	VALUE	SOURCE
tpool	mounted	yes	-
tpool/home	mounted	yes	-
tpool/home/test	mounted	yes	-
tpool/home/test/data	mounted	no	-

実行例 1-2：ファイルシステム“tpool”以下がマウントされているかを、“mount”コマンドで確認します。
root@opensolaris:~# **mount -p | grep -i tpool** ⏎
tpool - /tpool zfs - no rw,devices,setuid,nonbmand,exec,xattr,atime
tpool/home - /tpool/home zfs - no rw,devices,setuid,nonbmand,exec,xattr,atime
tpool/home/test - /tpool/home/test zfs - no rw,devices,setuid,nonbmand,exec,xattr,atime
実行例 2：ファイルシステム“tpool/home/test/data”をマウントします。
root@opensolaris:~# **zfs mount tpool/home/test/data** ⏎
実行例 3：ファイルシステム“tpool/home/test/data”をアンマウントします。
root@opensolaris:~# **zfs unmount tpool/home/test/data** ⏎

【注意】 ZFS は親ファイルシステムをマウントする前に子ファイルシステムをマウントすると、親ファイルシステムのマウント状態を復旧できないことがあります。

3-2-2-5. ZFS の予約領域と使用容量の制限設定

ZFS では、それぞれの子ファイルシステムに予約領域を設定して、一定の容量を確保することができます。予約領域を設定した場合、親ファイルシステムの容量はその予約領域の分だけ利用されたことになります。

ファイルシステムの使用容量に制限を設けたい場合は、“quota”プロパティ値を変更することで、ファイルシステムの使用容量に制限を設けることができます。

構文 1：ZFS 予約領域の設定
zfs set reservation=<容量> <ファイルシステム名>
構文 2：ZFS 予約領域の解除
zfs set reservation=none <ファイルシステム名>
構文 3：使用制限の設定
zfs set quota=<容量> <ファイルシステム名>
構文 4：使用制限の解除
zfs set quota=none <ファイルシステム名>

実行例 1：ファイルシステム“tpool/home/test/data”に 2GB の予約領域を設定して、確認します。
```
root@opensolaris:~# zfs set reservation=2G tpool/home/test/data ⏎
root@opensolaris:~# zfs list -r tpool ⏎
NAME                 USED  AVAIL  REFER  MOUNTPOINT
tpool                2.00G  7.77G    21K  /tpool
tpool/home           2.00G  7.77G    21K  /tpool/home
tpool/home/test      2.00G  7.77G    21K  /tpool/home/test
tpool/home/test/data   19K  9.77G    19K  /tpool/home/test/data
```
実行例 2：ファイルシステム“tpool/home/test/data”の予約領域を解除して、確認します。
```
root@opensolaris:~# zfs set reservation=none tpool/home/test/data ⏎
root@opensolaris:~# zfs list -r tpool ⏎
NAME                 USED  AVAIL  REFER  MOUNTPOINT
tpool                163K  9.77G    21K  /tpool
tpool/home            61K  9.77G    21K  /tpool/home
tpool/home/test       40K  9.77G    21K  /tpool/home/test
tpool/home/test/data  19K  9.77G    19K  /tpool/home/test/data
```
実行例 3：ファイルシステム“tpool/home/test”に 2GB の使用制限を設定して、確認します。
```
root@opensolaris:~# zfs set quota=2GB tpool/home/test ⏎
root@opensolaris:~# zfs list -r tpool ⏎
NAME                 USED  AVAIL  REFER  MOUNTPOINT
tpool                168K  9.77G    21K  /tpool
tpool/home            61K  9.77G    21K  /tpool/home
tpool/home/test       40K  2.00G    21K  /tpool/home/test
tpool/home/test/data  19K  2.00G    19K  /tpool/home/test/data
```
実行例 4：ファイルシステム“tpool/home/test”の使用制限を解除して、確認します。
```
root@opensolaris:~# zfs set quota=none tpool/home/test ⏎
root@opensolaris:~# zfs list -r tpool ⏎
NAME                 USED  AVAIL  REFER  MOUNTPOINT
tpool                168K  9.77G    21K  /tpool
tpool/home            61K  9.77G    21K  /tpool/home
tpool/home/test       40K  9.77G    21K  /tpool/home/test
tpool/home/test/data  19K  9.77G    19K  /tpool/home/test/data
```

3-3. スナップ、クローンとバックアップ

3-3-1. スナップショットの作成

ファイルシステムの状態を、写真を撮るように特定のタイミングで抜き出したものをスナップショットといいます。これは、スナップショット作成時のファイルシステムをいつでも読み出すことができる機能です。日付別にファイルシステムを管理する、特定の日時にファイルシステムの状態を戻すなどの操作ができます。

ZFS では、"-r"オプションを指定することで、子孫のファイルシステムすべてのスナップショットを一度にまとめて取得することができます。

構文 1：スナップショットの作成
zfs snapshot <ファイルシステム名>@<スナップショット名>
構文 2：スナップショットを子孫ファイルシステムもまとめて作成
zfs snapshot [-r] <ファイルシステム名>@<スナップショット名>
構文 3：スナップショットの一覧表示
zfs list -t snapshot

実行例 1：ファイルシステム"tpool/home"のスナップショットを作成して、確認します。

```
root@opensolaris:~# zfs snapshot tpool/home@20091117 ⏎
root@opensolaris:~# zfs list -t snapshot ⏎
NAME                                 USED AVAIL  REFER MOUNTPOINT
rpool/ROOT/opensolaris-1@install     148M -      2.82G  -
tpool/home@20091117                  0    -      21K    -
```

実行例 2：ファイルシステム"tpool"以下のスナップショットをまとめて作成して、確認します。

```
root@opensolaris:~# zfs snapshot -r tpool@20091117 ⏎
root@opensolaris:~# zfs list -t snapshot ⏎
NAME                                 USED    AVAIL  REFER MOUNTPOINT
rpool/ROOT/opensolaris-1@install     148M      -    2.82G  -
tpool@20091117                       0         -    21K    -
tpool/home@20091117                  0         -    21K    -
tpool/home/test@20091117             0         -    21K    -
tpool/home/test/data@20091117        0         -    19K    -
```

3-3-2. スナップショットの参照

スナップショットで取得したファイルシステムはいつでも見ることができます。保存されているスナップショットの中から、参照したいファイルシステムを選択することができます。

構文 1：ディレクトリからスナップショットの内容を確認
ls <ファイルシステムのマウントポイント>/.zfs/snapshot/<スナップショット名>

実行例 1：ファイルシステム"rpool/ROOT/opensolaris-1"のスナップショット"install"の内容を確認します。

```
root@opensolaris:~# ls /.zfs/snapshot/install ⏎
bin      dev      export       lost+found  opt     reconfigure   sbinusr
boot     devices  kernelmedia  platform    root    systemvar
cdrom    etc      lib          mnt         proc    rpooltmp
```

3-3-3. スナップショット作成時の状態にロールバックする

スナップショットを利用して、ファイルシステムを以前の状態に復元することを、ロールバック(rollback)といいます。最新のスナップショットより古いスナップショットを使用したい場合は、それ以後から最新のスナップショットを破棄する必要があります。また、破棄するスナップショットを保存しておく必要がなければ、"-r"オプションでロールバックと同時に破棄することができます。

構文 1：ファイルシステムをロールバック
zfs rollback <ファイルシステム名>@<スナップショット名>

実行例 1：ファイルシステム"tpool/home/test/data"のスナップショット"20091117"より新しいものを破棄してロールバックします。

```
root@opensolaris:~# zfs list -t snapshot ⏎
NAME                                          USED  AVAIL  REFER MOUNTPOINT
rpool/ROOT/opensolaris-1@install              148M    -     2.82G  -
rpool/ROOT/opensolaris-1@2009-10-19-07:32:50  31.4M   -     2.87G  -
tpool/home/test/data@20091117                 15K     -     19K    -
tpool/home/test/data@20091118                 15K     -     19K    -
tpool/home/test/data@20091118-1               0       -     19K    -
tpool/home/test/data@20091118-2               0       -     19K    -
root@opensolaris:~# zfs rollback -r tpool/home/test/data@20091117 ⏎
root@opensolaris:~# zfs list -t snapshot ⏎
NAME                                          USED   AVAILREFER   MOUNTPOINT
rpool/ROOT/opensolaris-1@install              148M   -     2.82G    -
rpool/ROOT/opensolaris-1@2009-10-19-07:32:50  31.4M  -     2.87G    -
tpool/home/test/data@20091117                 0      -     19K      -
```

3-3-4. スナップショットの破棄

ファイルシステムの破棄などと同様に、"zfs destroy"コマンドを使用して、スナップショットを破棄することができます。"-r"オプションを指定すると、同じ時点に作成した子孫ファイルシステムのスナップショットもまとめて破棄できます。

構文 1：スナップショットの破棄
zfs destroy <ファイルシステム名>@<スナップショット名>
構文 2：スナップショットを子孫ファイルシステムもまとめて破棄
zfs destroy [-r] <ファイルシステム名>@<スナップショット名>

実行例 2：ファイルシステム"tpool"以下のスナップショット"20091117"をまとめて破棄して確認します。

```
root@opensolaris:~# zfs list -t snapshot ⏎
NAME                                          USED  AVAIL  REFER MOUNTPOINT
rpool/ROOT/opensolaris-1@install              148M  -      2.82G  -
rpool/ROOT/opensolaris-1@2009-10-19-07:32:50  31.4M -      2.87G  -
tpool@20091117                                0       -     21K    -
tpool/home@20091117                           16K     -     21K    -
tpool/home/test@20091117                      16K     -     21K    -
tpool/home/test/data@20091117                 0       -     19K    -
root@opensolaris:~# zfs destroy -r tpool@20091117 ⏎
root@opensolaris:~# zfs list -t snapshot ⏎
NAME                                          USED   AVAIL  REFER MOUNTPOINT
rpool/ROOT/opensolaris-1@install              148M   -      2.82G  -
rpool/ROOT/opensolaris-1@2009-10-19-07:32:50  31.4M  -      2.87G  -
```

3-3-5. クローンの作成

書き込み可能なスナップショットをクローンといいます。スナップショットの内容を変更したい場合に使用します。クローンは元スナップショットが破棄されると使用できなくなります。クローンは通常のファイルシステムと同様に自動的にマウントされ、読み出しと書き込みが可能になります。クローンとファイルシステムを見分けるためには、プロパティ“origin”の値を参照します。

構文 1：クローンの作成
zfs clone <ファイルシステム名>@<スナップショット名> <保存先クローン名>
構文 2：プロパティ“origin”の値の参照
zfs get [-r] origin <ファイルシステム名>

実行例 1：ファイルシステム“tpool/home/test/data”のスナップショット“20091117”からクローンを作成してプロパティ“origin”の値を確認します。

```
root@opensolaris:~# zfs clone tpool/home/test/data@20091117
tpool/home/test/data_clone ⏎
root@opensolaris:~# zfs list -r tpool/home/test ⏎
NAME                          USED AVAIL   REFER MOUNTPOINT
tpool/home/test               59K  9.77G    21K  /tpool/home/test
tpool/home/test/data          19K  9.77G    19K  /tpool/home/test/data
tpool/home/test/data_clone    0    9.77G    19K  /tpool/home/test/data_clone
root@opensolaris:~# zfs get -r origin tpool/home/test ⏎
NAME                          PROPERTY VALUE                          SOURCE
tpool/home/test               origin   -                              -
tpool/home/test@20091117      origin   -                              -
tpool/home/test/data          origin   -                              -
tpool/home/test/data@20091117 origin   -                               -
tpool/home/test/data_clone    origin   tpool/home/test/data@20091117 -
```

3-3-6. ファイルシステムの内容をクローンで置き換える

ファイルシステムをクローンで置き換えるには、“zfs promote”コマンドを使用します。クローンを作成したスナップショットの元ファイルシステムは、クローンを残して破棄することができません。このコマンドを使用することで、クローンの親子依存関係が逆転し、元ファイルシステムが、指定されたファイルシステムのクローンになり、破棄することができるようになります。

構文 1：元ファイルシステムとクローンの親子依存関係を入れ替える
zfs promote <クローン名>
構文 2：ファイルシステムの名前を変更
zfs rename <ファイルシステム名> <新ファイルシステム名>

実行例 1：ファイルシステム“tpool/home/test/data”と、そのスナップショットから作成されたクローン“tpool/home/test/data_clone”の親子依存関係を入れ替えて破棄し、名前を変更します。

```
root@opensolaris:~# zfs list -r tpool ⏎
NAME                          USED AVAI     LREFER MOUNTPOINT
tpool                         300K 9.77G     23K   /tpool
tpool/home                    98K  9.77G     22K   /tpool/home
tpool/home/test               59K  9.77G     21K   /tpool/home/test
tpool/home/test/data          19K  9.77G     19K   /tpool/home/test/data
tpool/home/test/data_clone    0    9.77G     19K   /tpool/home/test/data_clone
root@opensolaris:~# zfs promote tpool/home/test/data_clone ⏎
```

```
root@opensolaris:~# zfs list -r tpool ⏎
NAME                          USED AVAIL   REFER MOUNTPOINT
tpool                         300K 9.77G   23K   /tpool
tpool/home                    98K  9.77G   22K   /tpool/home
tpool/home/test               59K  9.77G   21K   /tpool/home/test
tpool/home/test/data          0    9.77G   19K   /tpool/home/test/data
tpool/home/test/data_clone    19K  9.77G   19K   /tpool/home/test/data_clone
root@opensolaris:~# zfs destroy tpool/home/test/data ⏎
root@opensolaris:~# zfs list -r tpool ⏎
NAME                          USED AVAIL  REFER MOUNTPOINT
tpool                         263K 9.77G  21K   /tpool
tpool/home                    98K  9.77G  22K   /tpool/home
tpool/home/test               59K  9.77G  21K   /tpool/home/test
tpool/home/test/data_clone    19K  9.77G  19K   /tpool/home/test/data_clone
root@opensolaris:~# zfs rename tpool/home/test/data_clone tpool/home/test/data
⏎
root@opensolaris:~# zfs list -r tpool ⏎
NAME                          USED AVAIL   REFER MOUNTPOINT
tpool                         294K 9.77G   23K   /tpool
tpool/home                    98K  9.77G   2K    /tpool/home
tpool/home/test               59K  9.77G   21K   /tpool/home/test
tpool/home/test/data          19K  9.77G   19K   /tpool/home/test/data
```

3-3-7. スナップショットをストリームにまとめる

“zfs send”コマンドを使用することで、特定のスナップショットをストリーム形式(ZFS ストリーム)にまとめてファイルにリダイレクトしたり、dd コマンドのような別プロセスに渡したりすることができます。

作成した ZFS ストリームは cat や dd コマンドを入力し、パイプで“zfs receive”コマンドに渡して ZFS に適用することで、元のスナップショットを復元することができます。

ssh コマンドなどを利用して ZFS ストリームを送信することで、スナップショットを他のシステムへ転送することもできます。また、“zfs send”コマンドに“-R”オプションを指定することにより、子孫すべてを含めた形でファイルシステムやスナップショットをストリームにまとめることができます。

構文 1：リモートサーバーに ZFS ストリームを書き込む

```
# zfs send <ファイルシステム名>@<スナップショット名> | ssh [<ユーザー名>@]<ホスト名> dd of=
        <保存先ファイル名>
```

構文 2：リモートサーバーにスナップショットの子孫を含む ZFS ストリームを書き込む

```
# zfs send -R <ファイルシステム名>@<スナップショット名> | ssh [<ユーザー名>@]<ホスト> dd of=
        <保存先ファイル名>
```

実行例 1：SSH 経由でスナップショット“tpool@20091117”のストリームを送信し、リモートサーバーにファイル形式として書き込みます。文章中の太字(緑色)の部分は初回起動時のみです。

ここでは、送信先を“192.168.1.4”としています。

```
root@opensolaris:~# zfs send -R tpool@20091117 | ssh taro@192.168.1.4 dd
of=20091117 ⏎
The authenticity of host '192.168.1.4 (192.168.1.4)' can't be established.
RSA key fingerprint is 84:39:bf:9b:10:83:f4:aa:78:34:89:ed:87:11:40:70.
Are you sure you want to continue connecting (yes/no)? yes ⏎
Warning: Permanently added '192.168.1.4' (RSA) to the list of known hosts.
Password: ○○○○○○○○ ⏎
502+9 records in
504+1 records out
```

3-3-8. ZFS ストリームを取り込む

“zfs receive”コマンドで、ストリーム形式のスナップショットを展開して、元の状態に復元することができます。復元の際に、展開先に同じ名前のファイルシステムやスナップショットが存在してはいけませんが、“-F”オプションを指定することで、受信したストリーム形式のデータで同名の既存ファイルシステムを上書きすることができます。また、“-d”オプションで指定したファイルシステムの中に、送信側で使用していた名前でスナップショットを復元することができます。

送信先ユーザーが“zfs receive”コマンドを実行する際に、プロファイルで許可されていない場合、“pfexec”コマンドを指定します。また、パスが通っていない場合、“/usr/sbin/”を付け、絶対パス名で指定します。ここでは、リモート側は『そのどちらも設定されていない』、としています。この作業を行う場合には、あらかじめリモートシステム側の設定を確認してください。

構文 1：ZFS 複製ストリームからスナップショットを復元する

cat <ファイル名> | zfs receive -F -d <ファイルシステム名>

構文 2：“zfs send”コマンドで送信されたストリームからスナップショットを復元する

zfs send -R <ファイルシステム名>@<スナップショット名> | ssh [<ユーザー名>@]
<ホスト名> **pfexec /usr/sbin/zfs** receive -F -d <ファイルシステム名>

実行例 1：あらかじめ作成してあるスナップショット“tpool@20091117”を、“zfs send”コマンドを使用して送信されたストリームから、スナップショットを復元します。

ここでは送信先を“192.168.1.4”としています。

root@opensolaris:~# **zfs send -R tpool@20091117 | ssh taro@192.168.1.4 pfexec /usr/sbin/zfs**
receive -F -d tpool ⏎
Password: ○○○○○○○○ ⏎

3-3-9. スナップショットを差分転送する

スナップショットの増分(差分)だけを転送するには“zfs send”コマンドに“-i”(小文字の i)オプションを指定します。この場合、スナップショットを送受信するシステムの両方に、同等のスナップショットが存在していなければいけません。これは、“-R”オプションを指定して、子孫のファイルシステムすべてを含む複製ストリームを転送する場合も同様です。また、“zfs send”コマンドに“-I”(大文字の i)を指定することで、スナップショットすべての増分を取得して結合し、転送することができます。たとえば、20:40 と 20:43 のスナップショットの差分を転送する場合、“-i”(小文字の i)オプションの場合は 20:40 と 20:43 の差分を取得することになりますが、“-I”(大文字の i)オプションの場合、20:40 と 20:41、20:41 と 20:42、20:42 と 20:43 の差分をまとめて転送することになります。

構文 1：スナップショットを差分転送する

zfs send -i <古いスナップショット名> <新しいスナップショット名> | ssh [<ユーザー名>@] <ホスト名>
zfs receive -F -d <ファイルシステム名>

構文 2：結合されたスナップショットを送信する

zfs send -I <古いスナップショット名> <新しいスナップショット名> | ssh [<ユーザー名>@] <ホスト名>
zfs receive -F -d <ファイルシステム名>

実行例 1：スナップショット“tpool@20091117”と“tpool@20091117_1”の差分を転送します。

ここでは転送先を、“192.168.1.2”としています。

root@opensolaris:~# **zfs send -i tpool@20091117 tpool@20091117_1 | ssh katsumi@192.168.1.2 pfexec /usr/sbin/zfs receive -F -d tpool** ⏎
Password: ○○○○○○○○ ⏎

3-4. iSCSI(shareiscsi)

3-4-1. iSCSI でエクスポートする

ZFS が管理するストレージリソースを、iSCSI ターゲットとして他のコンピュータに提供することができます。iSCSI とは、ディスクなどの周辺デバイスやコンピュータ間でのデータ転送をするのに使用される SCSI プロトコルを、TCP/IP を介してやりとりできるようにした技術です。

SCSI の概念では、同じバスに接続するデバイスやコンピュータは対等なノードです。ノード上のリソースをターゲットと呼び、ターゲットを要求する側をイニシエータと呼びます。iSCSI でも同様に、他のノードにリソースを提供する側をターゲット、要求する側をイニシエータと呼びます。

他の OS も同様ですが、Solaris の iSCSI ターゲット/イニシエータは、サーバー/クライアント型のプログラムとして実装されています。ここでは、サーバープログラムが ZFS のストレージプールから作成したボリュームを、iSCSI イニシエータを介してローカルのボリュームとして使用します。iSCSI イニシエータは多くの OS に用意されており、Solaris システムがエクスポートしたボリュームは、Windows、Linux、MacOS X などからも利用が可能です。

OpenSolaris の iSCSI のイニシエータ/ターゲットソフトウェアは、“pkg.opensolaris.org”からパッケージをダウンロードして、インストールします。このパッケージをインストールした後、再起動してください。再起動することでインストールが完了します。

● iSCSI ターゲット側の設定

設定例 1：iSCSI パッケージのインストール
pkg install SUNWiscsi SUNWiscsitgt ⏎
設定例 2：SMF で iSCSI ターゲットのサービスを有効にする
svcadm enable svc:/system/iscsitgt:default ⏎
設定例 3：ストレージプールからボリュームを作成
zfs create -V <容量> <ボリューム> ⏎
設定例 4：作成したボリュームを iSCSI でエクスポートする
zfs set shareiscsi=on <ボリューム> ⏎
設定例 5：iSCSI ターゲットとして機能していることを確認
iscsitadm list target -v ⏎

● iSCSI イニシエータ側の設定

設定例 1：iSCSI パッケージのインストール
pkg install SUNWiscsi SUNWiscsi ⏎
設定例 2：iSCSI イニシエータを SMF で有効にする
svcadm enable iscsi_initiator ⏎
設定例 3：iSCSI ターゲットの検索対象として IP アドレスを指定する
iscsiadm add discovery-address <IP アドレス> ⏎
設定例 4：iSCSI ターゲットへの接続を有効にする
iscsiadm modify discovery -t enable ⏎
設定例 5：iSCSI ターゲットへの接続を無効にする
iscsiadm modify discovery -t disable ⏎

● iSCSI ターゲット側の実行例

実行例 1：iSCSI パッケージをインストールします。

```
root@opensolaris:~# pkg install SUNWiscsi SUNWiscsitgt ⏎
DOWNLOAD      PKGS   FILES  XFER (MB)
Completed      2/2   18/18  0.79/0.79

PHASE          ACTIONS
Install Phase             78/78
```

実行例 2：SMF で iSCSI ターゲットのサービスを有効にします。

```
root@opensolaris:~# svcadm enable svc:/system/iscsitgt:default ⏎
```

実行例 3：ボリューム“tpool/iscsi”を iSCSI ターゲットとしてエクスポートして確認します。

```
root@opensolaris:~# zfs create -V 1g tpool/iscsi ⏎
root@opensolaris:~# zfs shareiscsi=on tpool/iscsi ⏎
root@opensolaris:~# iscsitadm list target ⏎
Target: tpool/iscsi
    iSCSI Name: iqn.1986-03.com.sun:02:d9423521-877c-cdef-d4be-e6e054315231
    Connections: 0
```

● iSCSI イニシエータ側の実行例

実行例 1：iSCSI パッケージをインストールします。

```
root@act131:~# pkg install SUNWiscsi SUNWiscsiini ⏎
DOWNLOAD      PKGS   FILES  XFER (MB)
Completed      1/1      8/8   0.26/0.26

PHASE          ACTIONS
Install Phase             35/35
```

実行例 2：iSCSI イニシエータを SMF で有効にします。

```
root@act131:~# svcadm enable iscsi_initiator ⏎
```

実行例 3：エクスポートされている iSCSI ターゲットをイニシエータで参照できるように設定して確認します。

```
root@act131:~# iscsiadm add discovery-address 192.168.1.3 ⏎
root@act131:~# iscsiadm modify discovery -t enable ⏎
root@act131:~# iscsiadm list discovery ⏎
Discovery:
        Static: disabled
        Send Targets: enabled
        iSNS: disabled
root@act131:~# format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c0t600144F04B0B473F00080027851D8F00d0 <DEFAULT cyl 1022 alt 2 hd
64 sec 32>
          /scsi_vhci/disk@g600144f04b0b473f00080027851d8f00
       1. c7d0 <DEFAULT cyl 9722 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@1f,1/ide@0/cmdk@0,0
Specify disk (enter its number): ^C
```

● iSCSI ターゲット側の確認例

実行例 1：接続が有効になっているかを確認します。

```
root@opensolaris:~# iscsitadm list target ⏎
Target: tpool/iscsi
    iSCSI Name: iqn.1986-03.com.sun:02:d9423521-877c-cdef-d4be-e6e054315231
    Connections: 1
```

3-5. NFS(sharenfs)

3-5-1. NFS でエクスポートする

ZFS では、“sharenfs”プロパティの値を“on”にするだけで、ファイルシステムを NFS でエクスポートできます。従来のファイルシステムでの、“dfstab(4)”に記述してファイルシステムをエクスポートするという手順に比べて、格段に容易になっています。エクスポートを指定したファイルシステム下に子孫のファイルシステムを作成した場合、プロパティの設定値が継承されます。

“sharenfs”プロパティで共有を指定しているファイルシステムに対しては、“zfs share”で共有、“zfs unshared”コマンドで共有解除できます。また、オプション“-a”が指定されている場合は、すべてのファイルシステムに対しての設定が可能です。

構文 1：ファイルシステムを NFS でエクスポートする
zfs set sharenfs=on <ファイルシステム名>
構文 2：ファイルシステムを NFS のオプションを付与してエクスポートする
zfs set sharenfs=<オプション> <ファイルシステム名>
構文 3：“sharenfs”プロパティがセットされているファイルシステムを共有状態にする
zfs share <[-a] | <ファイルシステム名>>
構文 4：“unshared”プロパティがセットされているファイルシステムを共有解除する
zfs unshared <[-a] | <ファイルシステム名>>

実行例 1：ファイルシステム“tpool/home”に NFS によるエクスポートを設定して、その下に作成した子孫ファイルシステムにもプロパティが継承されていることを確認します。

```
root@opensolaris:~# zfs sharenfs=on tpool/home ⏎
root@opensolaris:~# share ⏎
-@tpool/home    /tpool/home    rw    ""
root@opensolaris:~# zfs create tpool/home/test ⏎
root@opensolaris:~# share ⏎
-@tpool/home      /tpool/home    rw    ""
-@tpool/home    /tpool/home/test    rw    ""
```

実行例 2：オプションが付与されているファイルシステムすべての共有を解除し、再度共有状態にして確認します。

```
root@opensolaris:~# zfs unshare -a ⏎
root@opensolaris:~# share ⏎
root@opensolaris:~# zfs share -a ⏎
root@opensolaris:~# share ⏎
-@tpool/home    /tpool/home    rw    ""
-@tpool/home    /tpool/home/test    rw    ""
```

3-6. CIFS(sharesmb)

3-6-1. CIFS でエクスポートする

Solaris の CIFS は Samba と異なり、Solaris カーネル内に実装されています。このことは、CIFS サービスを安定して動作させるとともに、管理面でも ZFS との密接な連携を可能にしています。

ZFS では、CIFS 用のプロパティ“sharesmb”が用意されており、プロパティを“on”にするだけで ZFS のファイルシステムを CIFS でエクスポートできます。

CIFS プロトコルは複数のユーザーがファイルを共有する時、排他ロックのメカニズムとして“強制ロック(mandatory locking)”を想定していますが、UNIX では紳士協定型の“協調ロック(advisory locking)”を利用しています。強制ロックを使用する場合は、ファイルシステムごとに“nbmand(non-blocking mandatory locking)”プロパティを指定することができます。

OpenSolaris で CIFS を利用するには、あらかじめ“pkg.opensolaris.org”から必要なパッケージをダウンロードして、インストールします。このパッケージをインストールした後、再起動してください。再起動することでインストールが完了します。

設定例 1：CIFS パッケージのインストール

pkg install SUNWsmbskr SUNWsmds ⏎

設定例 2：複数のロッキングメカニズムを併用できる ZFS ファイルシステムの作成

zfs create -o casesensitivity=mixed -o nbmand=on -o sharesmb=on <ファイルシステム名> ⏎

設定例 3：接続時の認証に PAM モジュール(pam_smb_passwd.so.1)を使用する

“/etc/pam.conf”構成ファイルに以下の行を追加

other password required pam_smb_passwd.so.1 nowarn

設定例 4：“pam.conf(4)”の設定後、CIFS サービス用のユーザーにパスワードを設定する

passwd <ユーザー名> ⏎

設定例 5：CIFS としてエクスポートされた ZFS ファイルシステムを確認する

smbutil view //[<ユーザー名>@]<IP アドレス> ⏎

設定例 6：CIFS クライアントサービスを有効にする

svcadm enable smb/client ⏎

設定例 7：CIFS でエクスポートされたファイルシステムをマウントする

mount -F smbfs //[<ユーザー名>@]<IP アドレス>/<CIFS サービス名> <マウントポイント> ⏎

実行例 1：CIFS パッケージをインストールします。

```
root@opensolaris:~# pkg install SUNWsmbskr SUNWsmbs ⏎
DOWNLOAD      PKGS   FILES   XFER (MB)
Completed      2/2   35/35  1.92/1.92

PHASE          ACTIONS
Install Phase            90/90
```

実行例 2：ファイルシステム“tpool/cifs”を CIFS としてエクスポートします。

また、“group”と“everyone”による書き込みができるように設定します。

```
root@opensolaris:~# zfs create -o casesensitivity=mixed -o nbmand=on -o
sharesmb=on tpool/cifs ⏎
root@opensolaris:~# /usr/bin/ls -V -d /tpool/cifs ⏎
drwxr-xr-x 2 rootroot   2 11 月  24 15:04 /tpool/cifs
                owner@:-------------- :-------:deny
                owner@:rwxp---A-W-Co-:-------:allow
                group@:-w-p---------- :-------:deny
```

```
                group@:r-x---------- :-------:allow
              everyone@:-w-p---A-W-Co- :-------:deny
              everyone@:r-x---a-R-c--s :-------:allow
root@opensolaris:~# chmod 777 /tpool/cifs ⏎
root@opensolaris:~# /usr/bin/ls -V -d /tpool/cifs ⏎
drwxrwxrwx 2 rootroot   2 11 月  24 15:04 /tpool/cifs
                owner@:---------- :-------:deny
                owner@:rwxp---A-W-Co-:-------:allow
                group@:----------:-------:deny
                group@:rwxp---------- :-------:allow
              everyone@:-------A-W-Co- :-------:deny
              everyone@:rwxp--a-R-c--s :-------:allow
```

実行例 3：接続時の認証に PAM モジュールを使用するために"/etc/pam.conf"に以下のように追加します。

```
root@opensolaris:~# cd /etc ⏎
root@opensolaris:/etc# vi pam.conf ⏎

#
# CDDL HEADER START
    :
#
other   session required  pam_unix_session.so.1
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
#
other   password required pam_dhkeys.so.1
other   password requisite pam_authtok_get.so.1
other   password requisite pam_authtok_check.so.1
other   password required pam_authtok_store.so.1
other  password requiredpam_smb_passwd.so.1 nowarn
#
# Support for Kerberos V5 authentication and example configurations can
```

実行例 4："pam.conf(4)"の設定後、CIFS サービス用のユーザーにパスワードを設定します。

```
root@opensolaris:~# passwd ⏎
passwd: Changing password for taro
New Password: ○○○○○○○○ ⏎
Re-enter new Password: ○○○○○○○○ ⏎
passwd: password successfully changed for taro
```

実行例 5：CIFS としてエクスポートされた ZFS ファイルシステムを確認します。

```
root@opensolaris:~# smbutil view //taro@192.168.1.4 ⏎
Password: ○○○○○○○○ ⏎
Share  Type  Comment
-------------------------------------------------------
ipc$  IPC   Remote IPC
tpool_cifs disk

2 shares listed from 2 available
```

3-7. BE(Boot 環境：Boot Environment)

3-7-1. BE の管理

OpenSolaris では、“rpool”ストレージプール内のシステム領域を“BE”として管理しています。BE は、複数作成して、起動時に切り替えることができます。OpenSolaris は、BE は ZFS のクローンとして管理されているため、ディスク消費は差分の分だけで効率的です。また、BE の作成も瞬時に行うことができます。

OpenSolaris では、“pkg image-update”コマンドでシステムをアップデートするたびに、自動的に BE が作成されていきます。使用する BE は起動時の GRUB メニューで選択できるため、なんらかの理由でアップデートがうまくいかなかった場合でも、以前の状態に戻すことができます。

3-7-2. BE の作成

BE の作成には、“beadm create”コマンドを使用します。このコマンドは、現在アクティブになっている BE から ZFS のクローンとして新たな BE を作成します。作成した BE は、システム起動時の GRUB メニューから選択できます。作成した BE をブートした場合、“rpool/ROOT/<BE 名>”がルート“/”にマウントされます。

ディスクの設定ファイル“/etc/vfstab”を表示してみると、“/”にマウントされるファイルシステムが作成した BE に自動的に変更されていることが確認できます。

構文 1：BE の作成
beadm create <BE 名>

実行例 1：BE を作成します。
root@opensolaris:~# **beadm create BE1** ⏎

3-7-3. BE の表示

現在の BE の一覧は“beadm list”コマンドで表示できます。
“Mountpoint”の行に“/”が表示されている BE が今現在アクティブになっているブート環境です。

構文 1：BE の一覧を表示
beadm list

実行例 1：BE の一覧を表示します。
root@opensolaris:~# **beadm list** ⏎

```
BE   ActiveMountpoint Space Policy Created
----  -------- -------------- -------- --------  -------
BE1  --            43.0K static 2009-11-25 11:35
opensolaris NR /           3.18G static2009-11-19 14:48
```

3-7-4. beadm と lu コマンドの比較

Solaris の“lu”コマンドは、“Solaris Live Upgrade”関連のコマンドです。このコマンドは、OpenSolaris ではサポートされていません。OpenSolaris では、“beadm”コマンドを使用します。以下に“beadm”コマンドと、“lu”コマンドの比較を示します。

表 3-7-4-1. beadm コマンドと lu コマンドの比較

No	beadm コマンド	lu コマンド	説明
1	beadm	help	ヘルプ情報を表示します。
2	activate	activate	ブート環境をアクティブにします。
3	-	cancel	スケジュールされた Live Upgrade 機能での、 コピー/作成操作を取り消します。
4	-	compare	2 つのブート環境を比較します。
5	-	make	ファイルをコピーすることにより、 指定のブート環境のファイルシステムを作成します。
6	create	create	新しいブート環境を作成します。
7	-	curr	アクティブなブート環境名を表示します。
8	destroy	delete	ブート環境を破棄します。
9	-	exit	lu を終了します。
10	list	fslist	ブート環境に関する情報を表示します。
11	mount	mount	ブート環境をマウントします。
12	rename	rename	ブート環境の名前を変更します。
13	-	status	ブート環境のステータスを表示します。
14	umount or unmount	umount	ブート環境をマウント解除します。
15	-	upgrade or flash	ブート環境上のソフトウェアのインストール、 アップグレード、およびその他の機能を実行します。

3-8. ZONE における ZFS

Solaris Zone は、個々の Zone にノード名、IP アドレス、ユーザー、グループ、ディスク領域、ネットワークポート、ネームサーバーなどを持つことができます。これにより、セキュリティやアプリケーション障害をお互いに影響することなく構成する仮想化技術です。また、ZONE の名称は、インストールした元の OS を“Global Zone(大域ゾーン)”、その OS から作成された OS を“non-global zone(非大域ゾーン)”と呼びます。

個々の Zone は、ネットワークや共有ディスクを介して他のシステムにアクセスする場合を除いては、セキュリティを侵害、あるいは外の環境を見ることはできません。

ZFS の Zone は、同一サーバー内で Zone の複製を可能とし、アプリケーションや設定も丸ごと複製することができます。また、通常よりも高速にインストールすることができます。

従来のサービスの追加方式では、「1. ハードウェアの設置、2. OS、Volume 管理ソフトインストールと設定、3. サービスのインストール、4. サービスの設定」のような手順で行っていましたが、ZFS＋Zone による ZFS クローン機能で瞬時に追加することができるようになりました。また、別のサーバーにゾーン環境を丸ごと移動することが可能です。

ZFS においての Zone とは、セキュリティの更なる強化、ZFS のクローン機能との組み合わせやリモートコピーとの連携による更なる高速化と低消費容量を実現しています。

以下にゾーンプロビジョニングとの連携図(例)を示します。

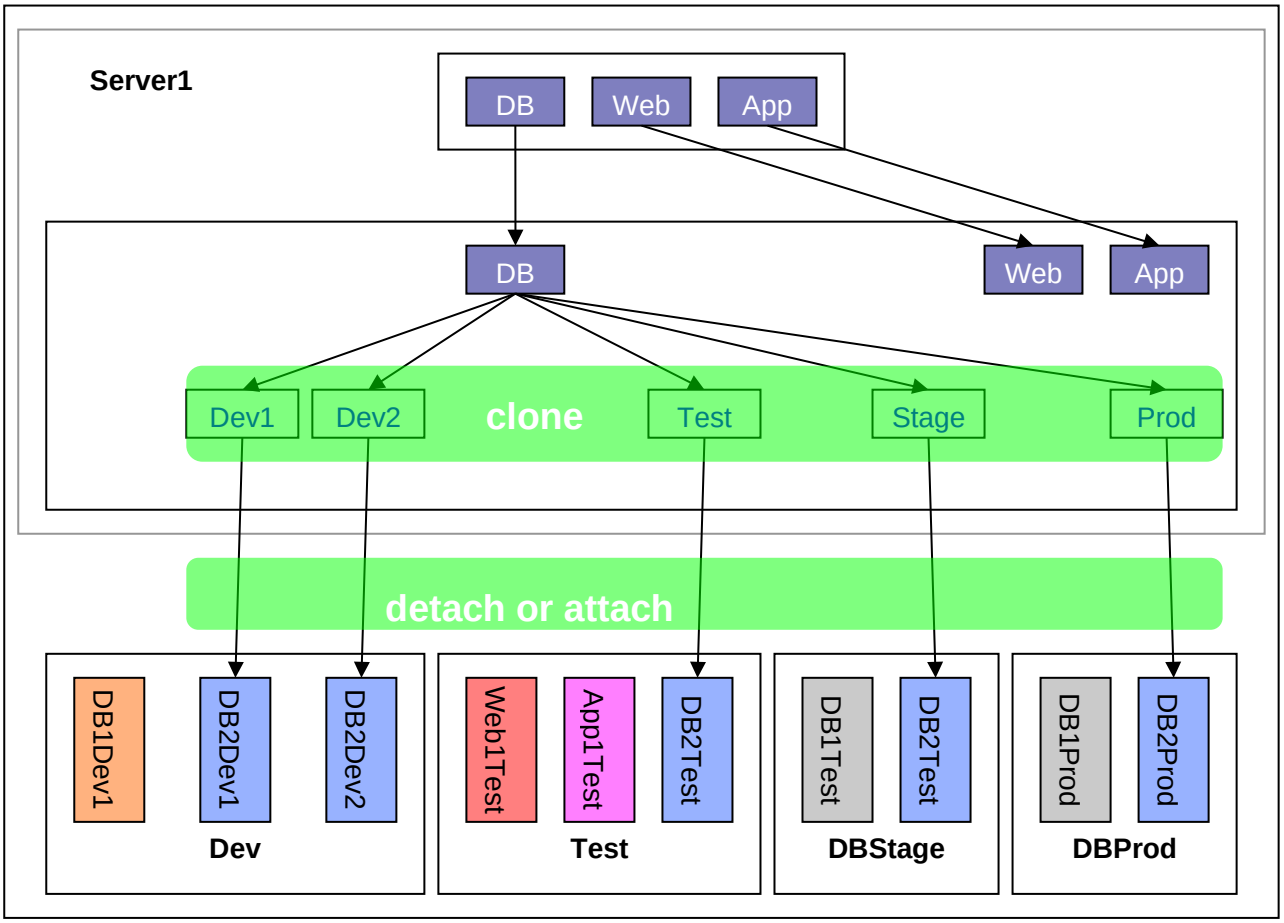


図 3-8-1. ゾーンプロビジョニングとの連携図(例)

出典：Solaris ZFS のメリットとその多角的な活用

ネイティブな非大域ゾーンの作成に必須の要素は、“zonename”および“zonepath”プロパティです。その他の資源およびプロパティは省略可能です。省略可能な資源には、“dedicated-cpu”資源と“capped-cpu”資源のどちらを使用するかを決めるなど、選択肢の中から選ぶ必要があるものもあります。

以下に、ゾーンの作成例を示します。

1. Zone 作成時に一番重要なプロパティは“zonepath”ですが、Solaris 10 のように適当なディレクトリを“mkdir”で作成して“zoonepath”に設定してはいけません。OpenSolaris では、“zonepath”に ZFS のデータセット名を設定する必要があります。例えば、“zonepath”に“/tpool/zone/test”と指定したい場合は、あらかじめ“chmod”コマンドでパーミッションを設定する必要があります。

```
root@opensolaris:~# zfs create tpool/zone ⏎
root@opensolaris:~# zfs create tpool/zone/test ⏎
root@opensolaris:~# chmod 700 /tpool/zone/test ⏎
root@opensolaris:~#
```

2. 次に“zonecfg”コマンドで Zone“test”の作成および設定を行います。

```
root@opensolaris:~# zonecfg -z test ⏎
test: No such zone configured  ←そのような構成済み Zone はありません
Use 'create' to begin configuring a new zone. ←“create”を使用して、
zonecfg:test> create ⏎      新しい Zone の構成を開始してください。
zonecfg:test> set zonepath=/tpool/zone/test ⏎
```

設定値を確認します。

```
zonecfg:test> info ⏎
zonename: test
zonepath: /tpool/zone/test
brand: ipkg
autoboot: false
bootargs:
pool:
limitpriv:
scheduling-class:
ip-type: shared
hostid:
zonecfg:test> ^C
```

“info”の出力結果で注目する箇所は、Solaris 10 と違いブランド名が“native”ではなく“ipkg”形式になっている事です。従って、“inherit-pkg-dir”のプロパティも無くなっています。

“zonecfg”の設定を終了後、Zone の状態を確認します。

```
root@opensolaris:~# zoneadm list -cv ⏎
  ID NAME  STATUS  PATH   BRAND  IP
   0 global  running  /  nativeshared
   - test    configured /tpool/zone/testipkg shared
root@opensolaris:~#
```

“test”のブランドが“native”ではなく“ipkg”になっています。

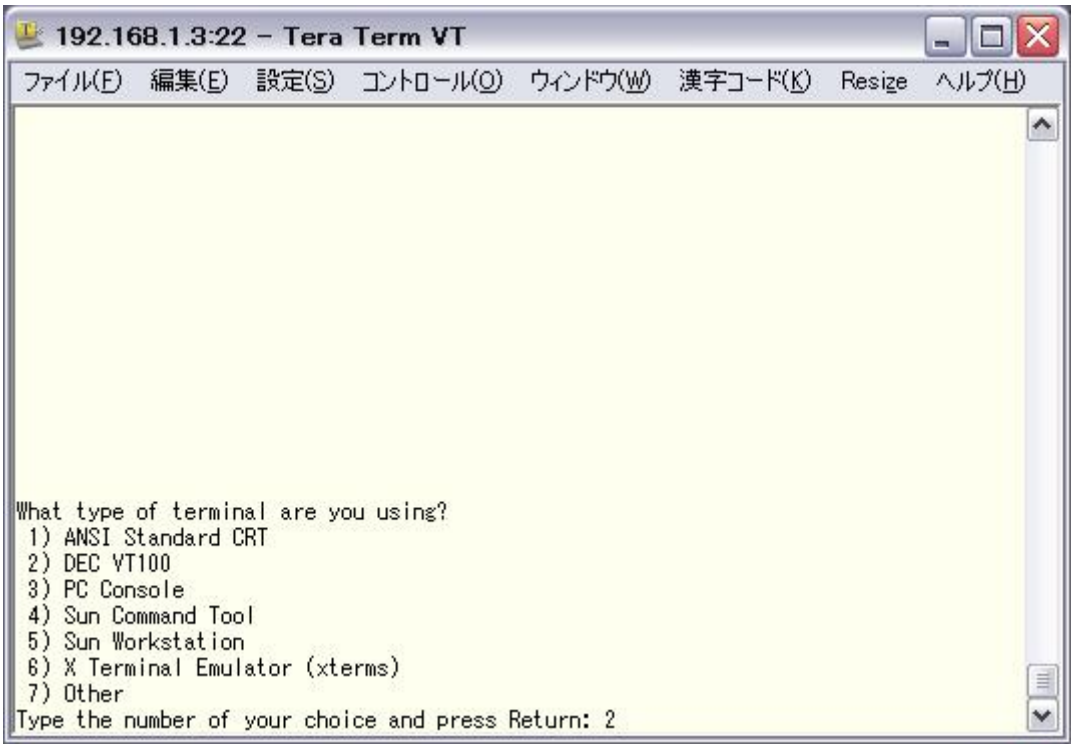
3. Zone の作成が完了しましたので、インストールします。Zone のインストールを実行すると、パッケージのリポジトリからパッケージのダウンロードが開始されます。ログにはインストールにかかった所要時間が表示されます。

```
root@opensolaris:~# zoneadm -z test install ⏎
    Publisher: Using opensolaris.org (http://pkg.opensolaris.org/release/).
        Image: Preparing at /tpool/zone/test/root.
        Cache: Using /var/pkg/download.
Sanity Check: Looking for 'entire' incorporation.
  Installing: Core System (output follows)
DOWNLOAD     PKGS       FILES  XFER (MB)
Completed    20/203021/302142.55/42.55

PHASE        ACTIONS
Install Phase     5747/5747
  Installing: Additional Packages (output follows)
DOWNLOAD     PKGS       FILES  XFER (MB)
Completed    37/375598/559832.52/32.52

PHASE        ACTIONS
Install Phase     7329/7329

          Note: Man pages can be obtained by installing SUNWman
 Postinstall: Copying SMF seed repository ... done.
 Postinstall: Applying workarounds.
        Done: Installation completed in 141.241 seconds.

  Next Steps: Boot the zone, then log into the zone console
              (zlogin -C) to complete the configuration process
root@opensolaris:~#
```

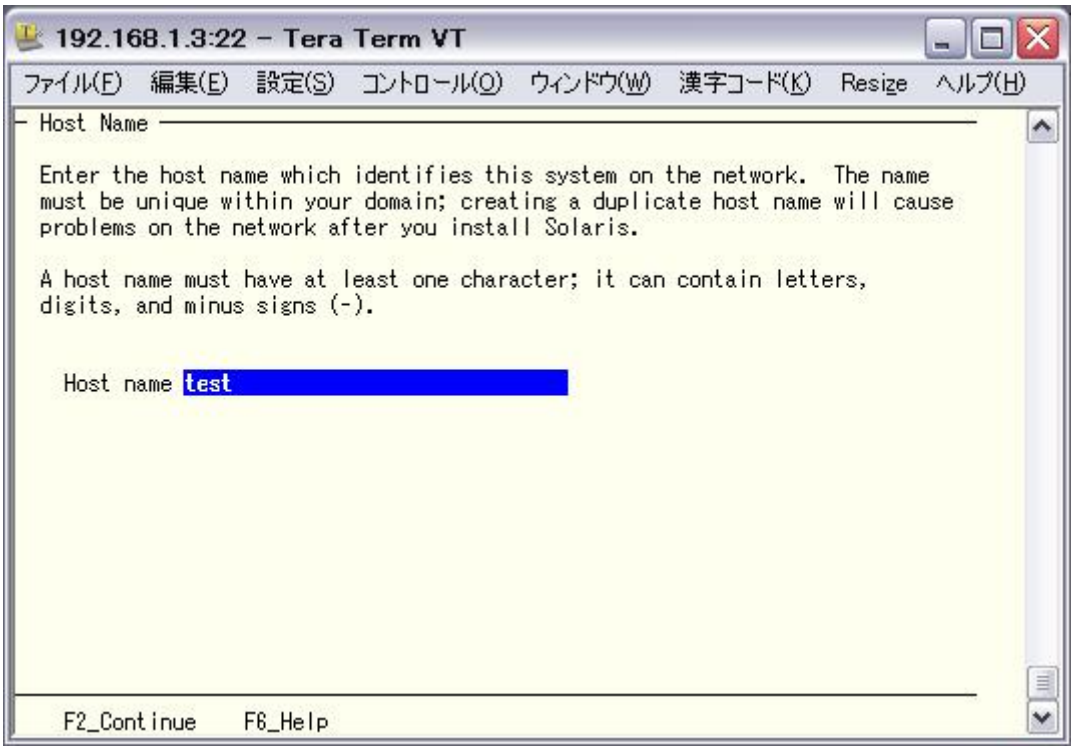
4. インストール完了後、Zone をブートして使用可能になります。
ブート後、“zlogin”コマンドで接続します。以下は、初期起動時のイメージです。

```
root@opensolaris:~# zoneadm -z test boot ⏎
root@opensolaris:~# zlogin -C test ⏎
[Connected to zone 'test' console]
69/69
Reading ZFS config: done.
Mounting ZFS filesystems: (5/5)
```

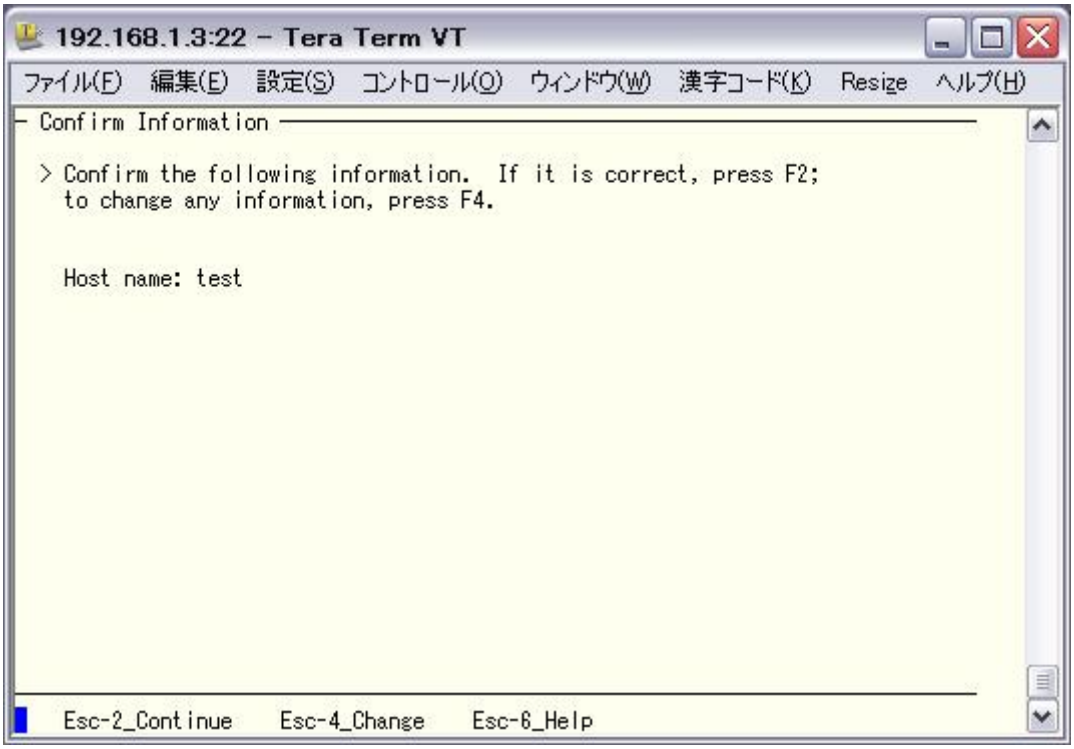
初期設定を行います。まず、端末タイプを選択します。選択後、⏎します。



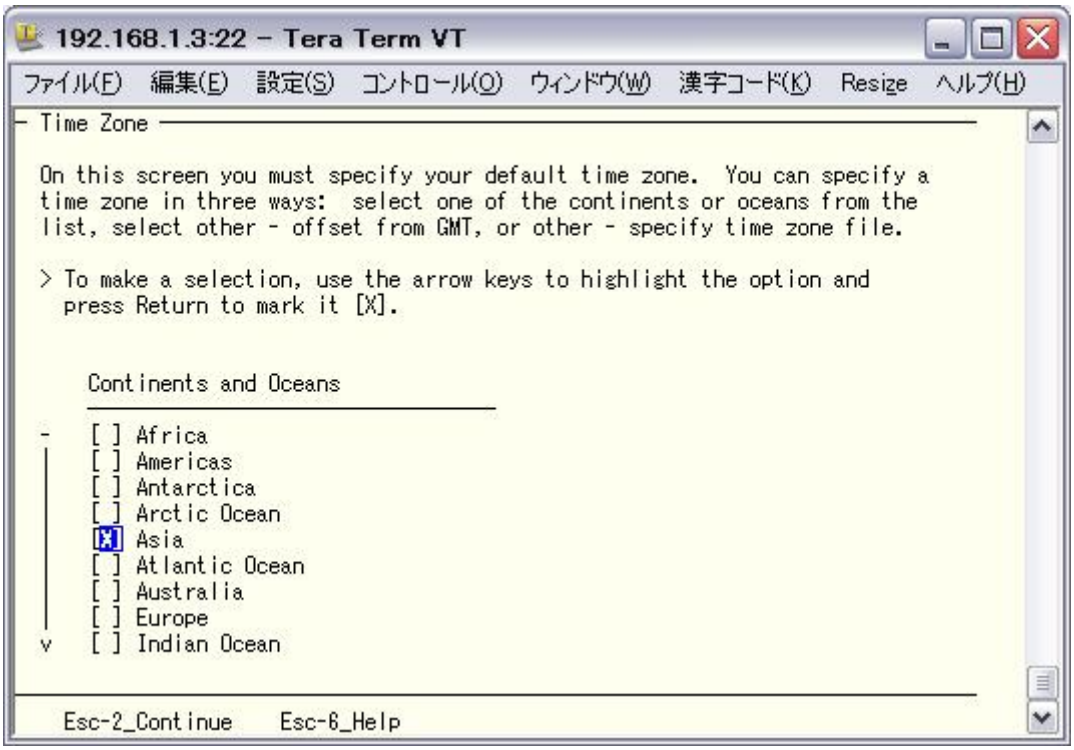
ホスト名を設定します。ホスト名を入力して、**<Esc+2>**を入力します。



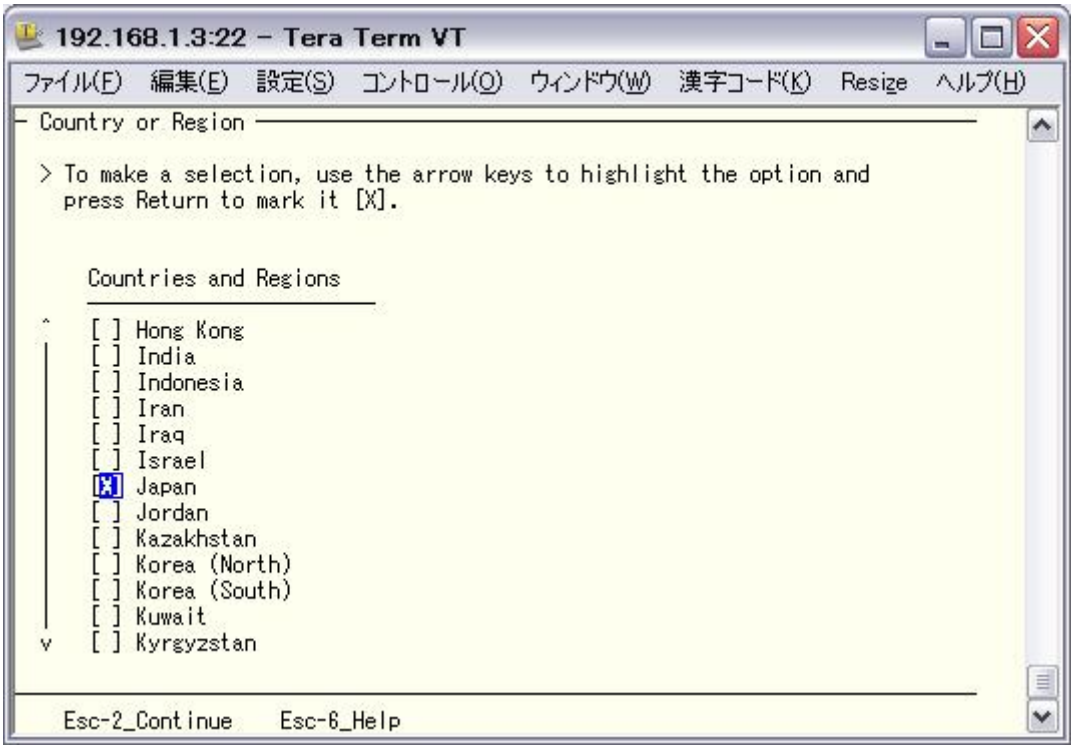
ホスト名設定確認です。よければ**<Esc+2>**を入力します。変更する場合は**<Esc+4>**を入力します。



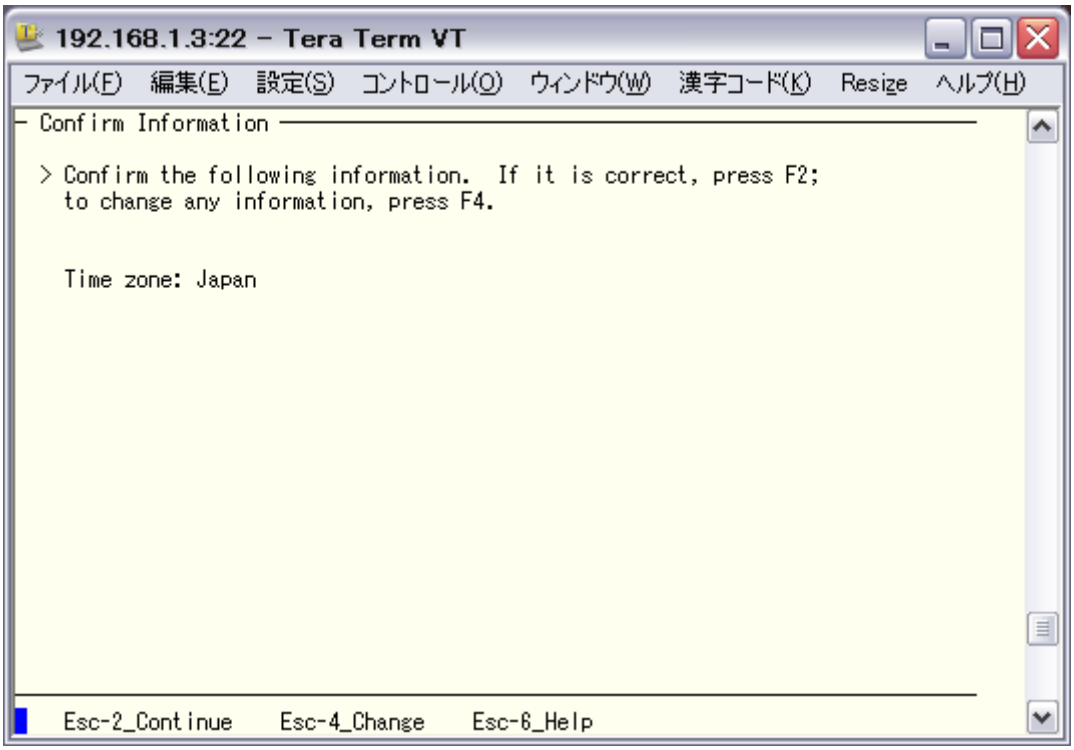
タイムゾーンの設定です。“Asia”にカーソルを移動させ、“**x**”を入力して、**<Esc+2>**を入力します。



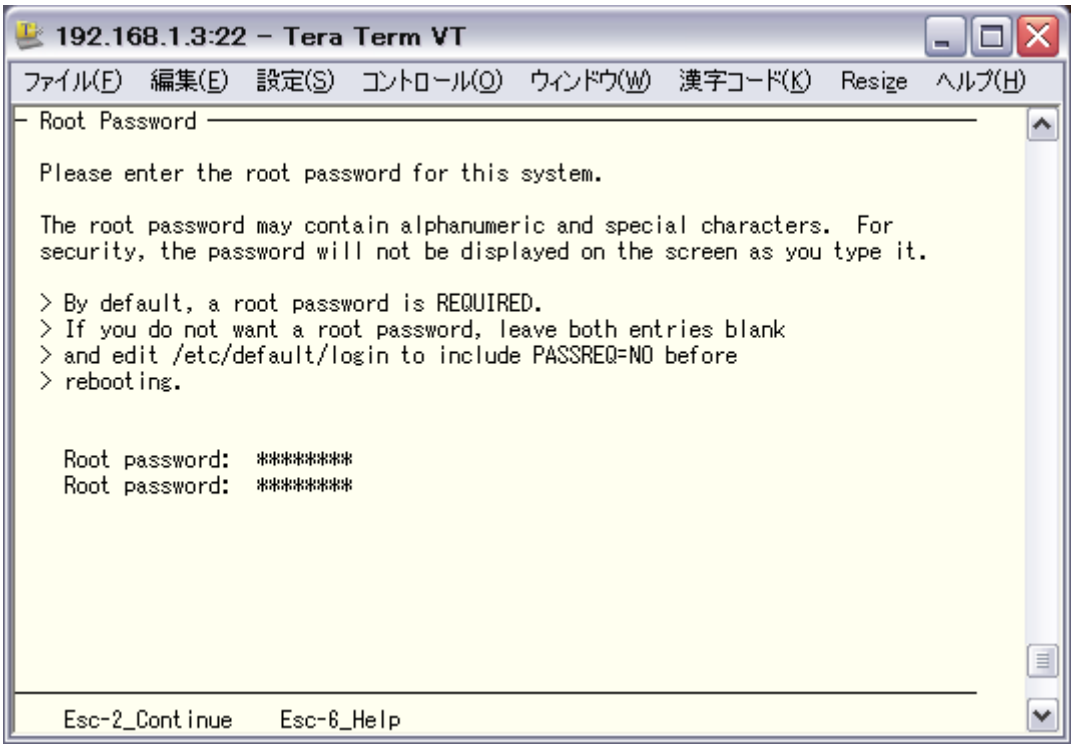
地域設定です。“Japan”にカーソルを移動させ、“**x**”を入力して、**<Esc+2>**を入力します。



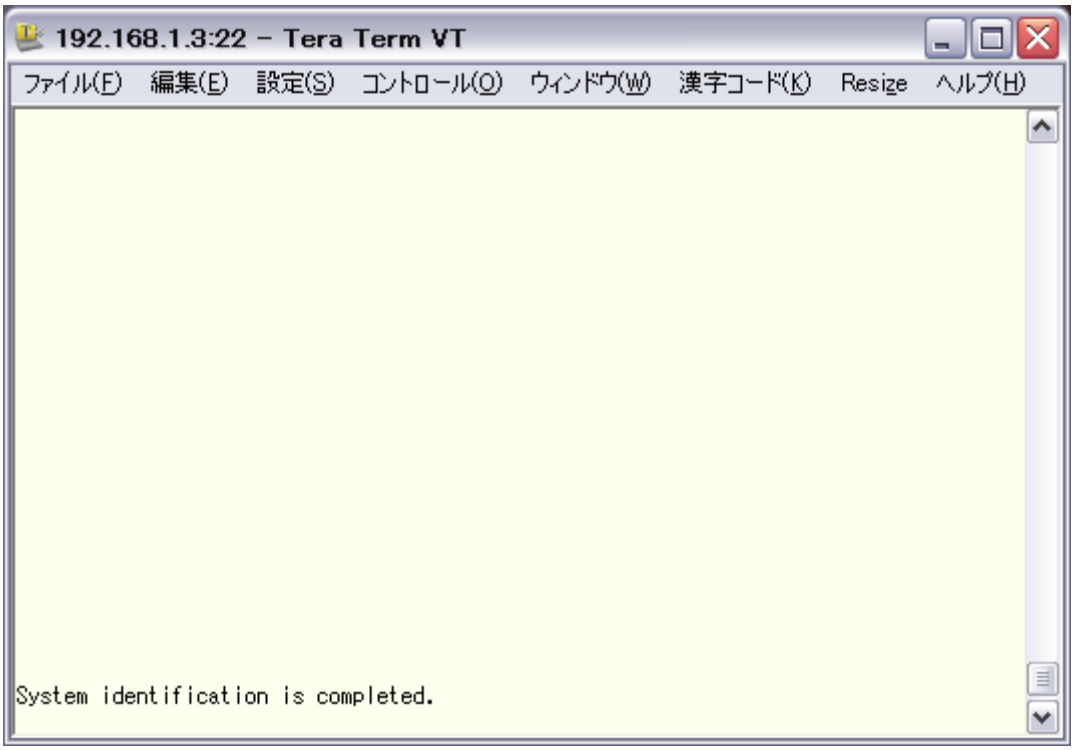
タイムゾーンの確認です。よければ**<Esc+2>**を入力します。変更する場合は**<Esc+4>**を入力します。



ルートパスワードの設定です。確認のために二度入力します。入力後、**<Esc+2>**を入力します。



システム確認完了です。



起動後、“**root**”でログインします。
test console login: **root** ⏎
Password: ○○○○○○○○ ⏎
Nov 6 00:08:04 test login: ROOT LOGIN /dev/console
Sun Microsystems Inc. SunOS 5.11 snv_111b November 2008
root@test:~#

以下に親(大域ゾーン)OS 内での“zfs list”、“zpool status”および“df -k”の見え方について示します。また、子(非大域ゾーン)OS と同じものは太字(緑色)になっています。

親(大域ゾーン)の OS

```
root@opensolaris:~# zfs list ⏎
NAME                                   USED  AVAIL  REFER  MOUNTPOINT
rpool                                  4.66G  4.14G    86K  /rpool
rpool/ROOT                             3.18G  4.14G    19K  legacy
rpool/ROOT/opensolaris                 8.80M  4.14G  2.87G  /
rpool/ROOT/opensolaris-1               3.17G  4.14G  2.99G  /
rpool/ROOT/opensolaris-1@install        148M      -  2.82G  -
rpool/ROOT/opensolaris-1@2009-10-19-    31.4M     -  2.87G  -
07:32:50
rpool/dump                             274M   4.14G   274M  -
rpool/export                           5.51M  4.14G    21K  /export
rpool/export/home                      5.49M  4.14G    22K  /export/home
rpool/export/home/taro                 5.47M  4.14G  5.47M  /export/home/taro
rpool/swap                             512M   4.54G   101M  -
rpool/zones1                           242M   4.14G    21K  /rpool/zones1
rpool/zones1/test-zone                 242M   4.14G    22K
rpool/zones1/test-zone/ROOT            242M   4.14G    19K  legacy
rpool/zones1/test-zone/ROOT/zbe        242M   4.14G   242M  legacy
rpool/zones2                           242M   4.14G    21K  /rpool/zones2
rpool/zones2/snap-zone                 242M   4.14G    22K
rpool/zones2/snap-zone/ROOT            242M   4.14G    19K  legacy
rpool/zones2/snap-zone/ROOT/zbe        242M   4.14G   242M  legacy
rpool/zones3                           242M   4.14G    21K  /rpool/zones3
rpool/zones3/clone-zone                242M   4.14G    22K
rpool/zones3/clone-zone/ROOT           242M   4.14G    19K  legacy
rpool/zones3/clone-zone/ROOT/zbe       242M   4.14G   242M  legacy
rpool/zonesx                           19K    4.14G    19K  /rpool/zonesx
tpool                                  484M   5.37G  23.0K  /tpool
tpool/zone                             242M   5.37G  21.0K  /tpool/zone
tpool/zone/test                        242M   5.37G  22.0K  /tpool/zone/test
tpool/zone/test/ROOT                   242M   5.37G  19.0K  legacy
tpool/zone/test/ROOT/zbe               242M   5.37G   242M  legacy
root@opensolaris:~#

root@opensolaris:~# zpool status ⏎
  pool : rpool
 state : ONLINE
 scrub: none requested
config :

        NAME  STATE           READ WRITE CKSUM
        rpool   ONLINE          0     0     0
          mirror ONLINE         0     0     0
            c7d0s0  ONLINE      0     0     0
            c7d1s0  ONLINE      0     0     0

errors: No known data errors

  pool: tpool
 state: ONLINE
 scrub: none requested
config:
```

```
NAME          STATE     READ      WRITE     CKSUM
tpool         ONLINE    0         0         0
raidz2        ONLINE    0         0         0
  c9t0d0      ONLINE    0         0         0
  c9t1d0      ONLINE    0         0         0
  c9t2d0      ONLINE    0         0         0
mirror        ONLINE    0         0         0
  c9t3d0      ONLINE    0         0         0
  c9t4d0      ONLINE    0         0         0
  c9t5d0      ONLINE    0         0         0
  c9t6d0      ONLINE    0         0         0
  errors: No known data errors
```

```
root@opensolaris:~# df -k ⏎
Filesystem                1K-blocks      Used Available Use% Mounted
rpool/ROOT/opensolaris-1
                          7473789   3137233   4336556  42% /
swap                      672872    304       672568   1%  /etc/svc/volatile
/usr/lib/libc/libc_hwcap2.so.1
                          7473789   3137233   4336556  42% /lib/libc.so.1
swap                      672576    8         672568   1%  /tmp
swap                      672616    48        672568   1%  /var/run
rpool/export              4336577   21        4336556  1%  /export
rpool/export/home         4336578   22        4336556  1%  /export/home
rpool/export/home/taro    4342153   5598      4336556  1%  /export/home/taro
rpool                     4336642   86        4336556  1%  /rpool
rpool/zones1              4336577   21        4336556  1%  /rpool/zones1
rpool/zones1/test-zone    4336578   22        4336556  1%  /rpool/zones1/test-zone
rpool/zones2              4336577   21        4336556  1%  /rpool/zones2
rpool/zones2/snap-        4336578   22        4336556  1%  /rpool/zones2/snap-zone
zone
rpool/zones3              4336577   21        4336556  1%  /rpool/zones3
rpool/zones3/clone-       4336578   22        4336556  1%  /rpool/zones3/clone-zone
zone
rpool/zonesx              4336575   19        4336556  1%  /rpool/zonesx
tpool                     5635698   23        5635676  1%  /tpool
tpool/ponyo               5635696   20        5635676  1%  /tpool/ponyo
tpool/zone                5635696   21        5635676  1%  /tpool/zone
tpool/zonesx              5635697   22        5635676  1%  /tpool/zonesx
tpool/zonesx/ponyo        5635694   19        5635676  1%  /tpool/zonesx/ponyo
tpool/zonesx/x-zone       5635698   22        5635676  1%  /tpool/zonesx/x-zone
tpool/zone/test           5635697   22        5635676  1%  /tpool/zone/test
tpool/zone/test/ROOT/     5883273   247597    5635676  5%  /tpool/zone/test/root
zbe
swap                      672756    188       672568   1%  /tpool/zone/test/root/etc/svc
/volatile
/tpool/zone/test/root/usr/lib/libc/libc_hwcap2.so.1
                          5883273   247597    5635676  5%  /tpool/zone/test/root/lib/libc.so.1
swap                      672568    0         672568   0%  /tpool/zone/test/root/tmp
swap                      672576    8         672568   1%  /tpool/zone/test/root/var/run
root@opensolaris:~#
```

以下に Zone 内での“zfs list”、“zpool status”および“df -k”の見え方を示します。

非大域ゾーンの OS

```
root@test:~# zfs list ⏎
NAME                      USED AVAIL  REFER MOUNTPOINT
tpool                     484M 5.37G  23.0K /tpool
tpool/zone                242M 5.37G  21.0K /tpool/zone
tpool/zone/test           242M 5.37G  22.0K /tpool/zone/test
tpool/zone/test/ROOT      242M 5.37G  19.0K legacy
tpool/zone/test/ROOT/zbe  242M 5.37G  242M  legacy
root@test:~#

root@test:~# zpool status ⏎
  pool : tpool
 state: ONLINE
 scrub : none requested
config:

        NAME  STATE          READ WRITE   CKSUM
        tpool  ONLINE           0     0       0
          raidz2  ONLINE        0     0       0
            c9t0d0  ONLINE      0     0       0
            c9t1d0  ONLINE      0     0       0
            c9t2d0  ONLINE      0     0       0
          mirror   ONLINE       0     0       0
            c9t3d0  ONLINE      0     0       0
            c9t4d0  ONLINE      0     0       0
            c9t5d0  ONLINE      0     0       0
            c9t6d0  ONLINE      0     0       0

errors: No known data errors
root@test:~#

root@test:~# df -k ⏎
Filesystem              Kbytes      used    avail    capacity  Mounted on
/                            0  247597 5635675     5%      /
/dev                         0       0       0     0%      /dev
proc                         0       0       0     0%      /proc
ctfs                         0       0       0     0%      /system/contract
mnttab                       0       0       0     0%      /etc/mnttab
objfs                        0       0       0     0%      /system/object
swap                    672592     188  672404     1%      /etc/svc/volatile
/usr/lib/libc/libc_hwcap2.so.1
                       5883272 247597 5635675     5%              /lib/libc.so.1
fd                           0       0       0     0%      /dev/fd
swap                    672404       0  672404     0%      /tmp
swap                    672412       8  672404     1%      /var/run
root@test:~#
```

3-9. ZFS バージョン

ZFS バージョンは“zpool upgrade -v”コマンドによって表示することができます。次に例を示します。

```
root@opensolaris:~# zpool upgrade -v ⏎
This system is currently running ZFS pool version 14.

The following versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
 7   Separate intent log devices
 8   Delegated administration
 9   refquota and refreservation properties
 10  Cache devices
 11  Improved scrub performance
 12  Snapshot properties
 13  snapused property
 14  passthrough-x aclinherit support
For more information on a particular version, including supported releases, see:

http://www.opensolaris.org/os/community/zfs/version/N

Where 'N' is the version number.
root@opensolaris:~#
```

図 3-9-1. OpenSolaris 2009.06 の ZFS バージョン

```
root@act011 # zpool upgrade -v ⏎
このシステムでは現在、バージョン 15 の ZFS プールが動作しています。

次のバージョンがサポートされています:

VER  説明
---  --------------------------------------------------------
 1   初期バージョンの ZFS
 2   Ditto ブロック(複製されたメタデータ)
 3   ホットスペアおよびダブルパリティー RAID-Z
 4   zpool history
 5   gzip アルゴリズムを使用した圧縮
 6   bootfs プールプロパティー
 7   別のインテントログデバイス
 8   委任管理
 9   refquota および refreservation プロパティー
 10  キャッシュデバイス
 11  Improved scrub performance
 12  Snapshot properties
 13  snapused property
 14  passthrough-x aclinherit
 15  user/group space accounting
サポートされるリリースなど、特定のバージョンの詳細については、以下を参照してください:

http://www.opensolaris.org/os/community/zfs/version/N

'N' はバージョン番号です。
root@act011 #
```

図 3-9-2. Solaris 10 10/09 の ZFS バージョン

第 4 章 ZFS インターナル

4-1. ZFS 構造

次に ZFS を構成するコンポーネントを解説します。図は BigAdmin System Administration Portal ZFS ソースツアーから引用しています(一部加筆・変更しています)。

インタフェースレイヤー(Interface Layer)の ZPL(ZFS POSIX Layer)、トランザクションオブジェクトレイヤー(Transaction Object Layer)のデータ管理ユニット(DMU：Data Management Unit)、およびプールストレージレイヤー(Pooled Storage Layer)のストレージプールアロケータ(SPA：Storage Pool Allocator)が主な ZFS の機能を実行します。

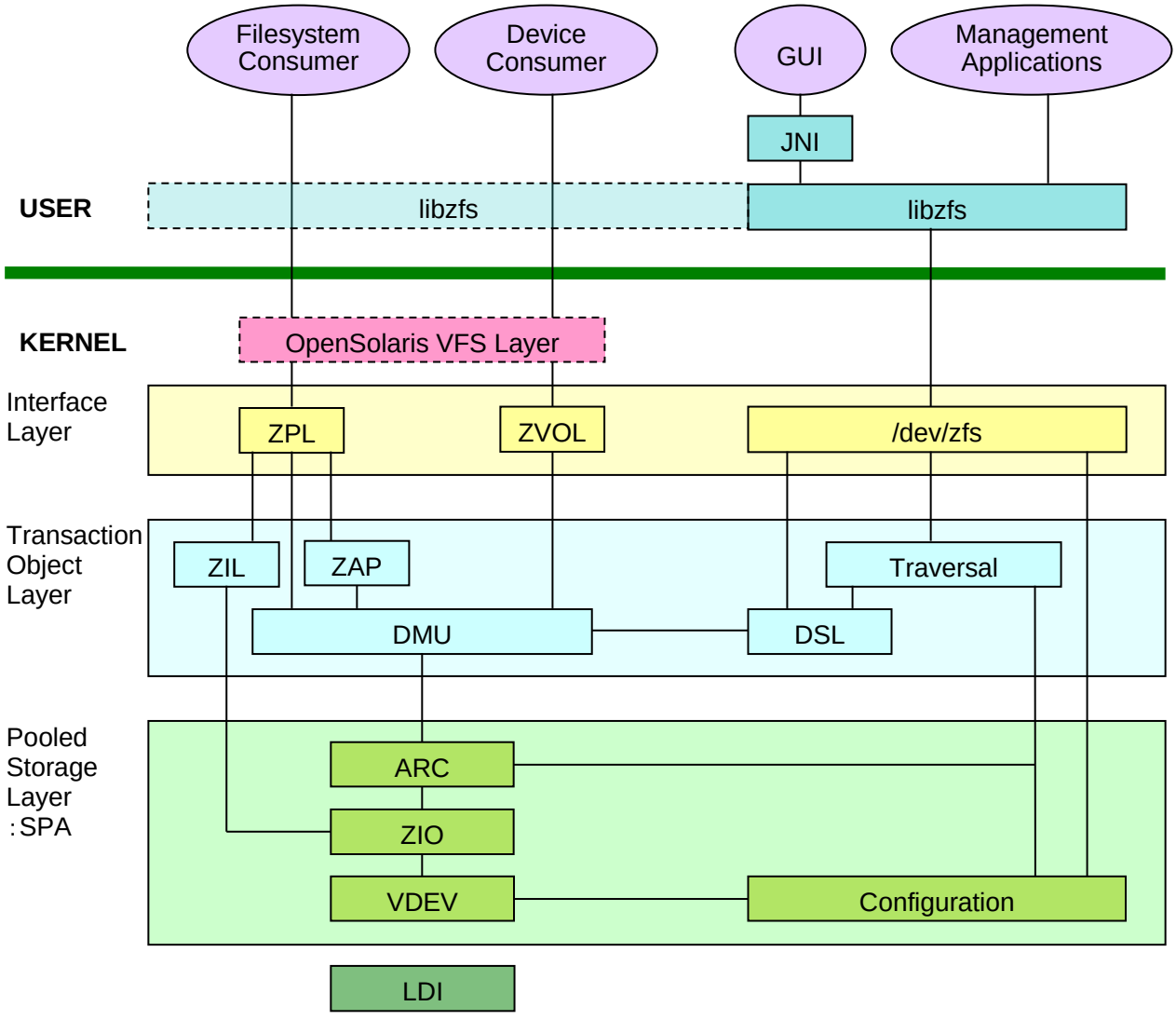


図 4-1-1. ZFS コンポーネントの関連図

出典：BigAdmin Syste Administration Portal ZFS ソースツアー

4-1-1. ユーザープログラム

次にユーザー空間にあるプログラムを概説します。

Filesystem Consumer ファイルシステムコンシューマ ZFS と会話するアプリケーション全てを示します。カーネルとの間に ZFS ライブラリを経由します。

Device Consumer デバイスコンシューマ VDEV(仮想デバイス)を管理します。

GUI グラフィックユーザーインタフェースの管理ルーチンです。Solaris 10 では WEB 経由の Java モジュールが、Java コンソールの元で実行されます。OpenSolaris では未リリースです。

Management Application 管理アプリケーション 代表的なコマンドは zpool や zfs コマンドです。zdb、beadm なども管理アプリケーションです。

libzfs プログラムから ZFS にアクセスするには、ライブラリを経由しています。

4-1-2. カーネルコンポーネント

カーネル空間にあるコンポーネントを概説します。

4-1-2-1. インタフェースレイヤ

ZPL ZFS POSIX Layer write や read システムコールや open、close などほぼ全部のインタフェースは POSIX で標準化されています。これらは、OpenSolaris VFS Layer を経由して操作されます。

ZVOL ZFS Volume Emulation Interface ストレージプール内のデバイスを直接ハンドリングするもので、生(ロウ)デバイスとしてアクセスするためのコンポーネントです。

/dev/zfs /dev/zfs 特殊デバイスです。デバイスは他のデバイスドライバと同様に、特殊ファイルを経由してアクセスします。デバイスは次のようになっています。

```
root@opensolaris:/devices/pseudo# ls -l zfs* ⏎
brw------- 1 root sys 182, 1 2009-11-29 22:48 zfs@0:1c
crw------- 1 root sys 182, 1 2009-11-29 22:48 zfs@0:1c,raw
brw------- 1 root sys 182, 2 2009-11-29 22:48 zfs@0:2c
crw------- 1 root sys 182, 2 2009-11-29 22:48 zfs@0:2c,raw
crw-rw-rw- 1 root sys 182, 0 2009-11-29 14:58 zfs@0:zfs
zfs@0:
total 0
root@opensolaris:/devices/pseudo#
```

4-1-2-2. トランザクショナルオブジェクトレイヤ

ZIL ZFS Intent Log ZFS インテントログ機能のコンポーネントです。
トランザクションコミットされるまでの間、情報を記録しておき、ロールバック可能にしておきます。コミットが正常に終了するとインテントログはクリアされます。

ZAP ZFS Attribute Processor ZFS 内のさまざまなオブジェクトのプロパティを管理するコンポーネントです。

Traversal ストレージプール内のデータ検査を行うコンポーネントです。同期化や、破損ブロックを検出して修復します。

DMU Data Management Unit データ管理ユニット トランザクションオブジェクトモデルを実装したデータ管理ユニットです。ユーザー(コンシューマ)はオブジェクトセット、オブジェクト、およびトランザクションを使用してデータを扱いますが、DMU はそれらをグループにまとめ、ディスクにコミットします。

DSL Dataset and Snapshot Layer データセット&スナップショットレイヤ データセットのスナップショット、およびクローンの管理を行います。DMU データセットを継承したプロパティ、割り当て管理、予約情報などに基づいて階層別にわかれた空間に統合します。

4-1-2-3. プールドストレージレイヤ

このレイヤは、**SPA**(Storage Pool Allocator)とも呼ばれます。

ARC Adaptive Replacement Cache
一次キャッシュをコントロールするコンポーネントです。

ZIO ZFS I/O Pipeline ZFS の入出力パイプラインをコントロールするコンポーネントです。

DVA デバイス仮想アドレス(DVA：Device Virtual Address)
VDEV 上の位置変換、データのチェックサム計算や圧縮を行います。

VDEV Virtual Device 仮想デバイスとして抽象化された VDEV は、どのディスク装置であるかを、与えられた値によって知ります。VDEV は 3 個あり、各々同じ内容を、別のブロック位置に持っています。

Conriguration 構成情報。

LDI Layered Driver Interface レイヤのドライバ情報。

なお、ソースコードの全ては未確認です。内容に大きな違いは無いかと思いますが、誤りがありましたらご指摘ください。

4-2. データディスクの構造
4-2-1. データディスクの準備

ZFS データディスクの構造を解説します【備考】。

ここでは、ストレージプールを次のように作成したディスクを調査しました。

```
# zpool create tpool c5t0d0 ⏎
```

OpenSolaris 2009.06 は X86 システム用です。ディスクの内容は Little Endian で作成されます。ディスクダンプを表示した場合、内容が Little Endian だと読みづらいため、ここでは SPARC システムで作成した Big Endian のストレージプールを使用しています。

当該ディスクのパーティション情報を次に示します。パーティション 8 の 8MB は予約されています。EFI ディスクラベル形式になっています。ブートディスクは、Cyl Head Sector の SMI ディスクラベル(VTOC)形式で、boot パーティションになります(後述)。

```
partition> print ⏎
Current partition table (original):
Total disk sectors available: 1953508750 + 16384 (reserved sectors)

Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm               256       931.51GB          1953508750
  1 unassigned    wm                 0            0                   0
  2 unassigned    wm                 0            0                   0
  3 unassigned    wm                 0            0                   0
  4 unassigned    wm                 0            0                   0
  5 unassigned    wm                 0            0                   0
  6 unassigned    wm                 0            0                   0
  7 unassigned    wm                 0            0                   0
  8    reserved    wm        1953508751       8.00MB          1953525134

partition>
```

図 4-2-1-1. データディスクのパーティション情報

セクター256～1953508750　名前：usr パーティション番号 0 → スライス 0
セクター1953508751～1953525134 名前：なし パーティション番号 8 → スライス 8

【注】セクター0～255 は未使用です。

図 4-2-1-2. データディスクのパーティションイメージ

【備考】ディスク装置の内部構造を知ることによって、ファイルシステムの特性を理解することができます。また、「物理的に読み取りできるがマウントできないディスク装置」を分析するのに役立ちます。過去、UFS、VxFS などで、部分的に破損したファイルシステムから、特定のファイルを抜き出すプログラム(データサルベージ)で、窮地をしのいだ経験があります。しかし、ZFS は頑丈なので、このようなことは全く出番が無いかも知れません。

次のようにテストファイルを作成し、ディスク上、どの位置にあるかを追跡してみます。

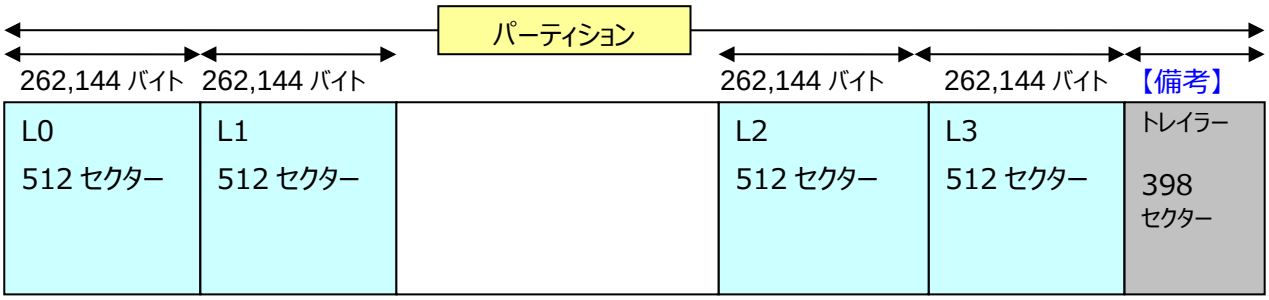
```
root@act061 # zfs list ⏎
NAME                       USED   AVAIL  REFER  MOUNTPOINT
rpool                      20.6G  46.4G  94K    /rpool
rpool/ROOT                 6.78G  46.4G  18K    legacy
rpool/ROOT/s10s_u7wos_08   6.78G  46.4G  6.78G  /
rpool/dump                 1.00G  46.4G  1.00G  -
rpool/export               10.8G  46.4G  20K    /export
rpool/export/home          10.8G  46.4G  10.8G  /export/home
rpool/swap                 2G     48.4G  16K    -
tpool                      36K    913G   19K    /tpool
tpool/testfs               20.5K  913G   20.5K  /tpool/testfs
root@act061 # cd /tpool/testfs/testdirectory ⏎
root@act061 # ls ⏎
testfile
root@act061 # cat testfile ⏎
abcdefghijklmnopqrstuvwxyzABCDEGGHIJKLMNOPQRSTUVWXYZ
1234567890-^¥@[;:],./¥!"#$%&'()=~|`{+*}<>?_*********
ABCDEGGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
*********1234567890-^¥@[;:],./¥!"#$%&'()=~|`{+*}<>?_
testfile TEST_DATA
root@act061 #
```

4-2-2. ZFS オンディスク構造体

4-2-2-1. VDEV ラベル

デバイスは 262,144 バイト(512 セクター×512 バイト)の VDEV ラベル L0、L1、L2、および L3 の 4 個を持っています。L0、L1 の VDEV ラベルは割り当てたパーティションの先頭に置かれます。L2、L3 の VDEV ラベルは、パーティションの最後に作成されます。L0、L1、L2、および L3 は同じデータが書き込まれ、ストレージプールのアクセスと検査・修復に用いられます。

VDEV ラベルの位置関係を次に示します。



【備考】 203,776 バイト(398 セクター)のトレイラーがありました。これは 1TB ディスク装置の場合の値です。他のサイズのディスク装置は未確認です。参考 WEB「ZFS On-Disk Specification Sun Microsystems,Inc」ではこのトレイラーの解説はなく、「お尻から 1024 セクター」となっています。

図 4-2-2-1(1). VDEV ラベル位置

VDEV ラベルは次の構造になっています。

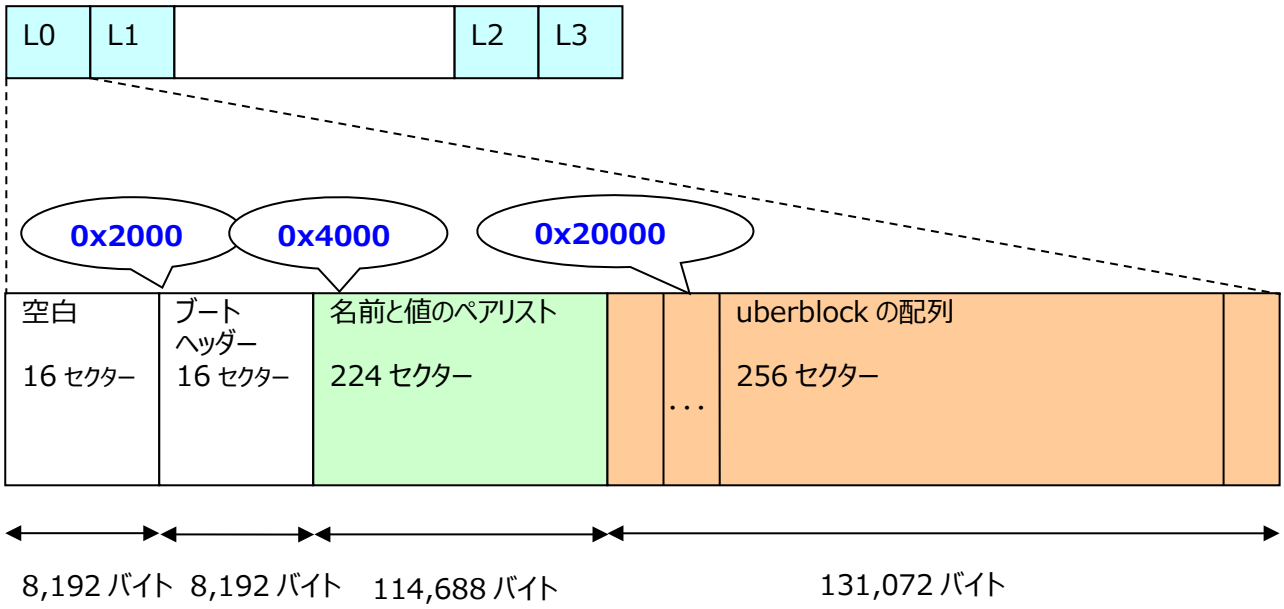


図 4-2-2-1(2). VDEV ラベル構造

名前と値のペアリストは nvlist と呼ばれています。次にダンプ例を示します。

図 4-2-2-1(3). 名前と値ペアリスト(nvlist)のダンプ例

nvlist の情報は zdb コマンドで表示することができます。わざわざディスクダンプする必要はありません。また、後述の uberblock やオブジェクトのサマリが表示されます。

zdb コマンドは ZFS デバッガです。通常運用で使用することはありません。

```
root@act061 # zdb tpool ⏎
    version=10
    name='tpool'
    state=0
    txg=1643
    pool_guid=15530684243210972302
    hostid=2202309925
    hostname='act061'
    vdev_tree
        type='root'
        id=0
        guid=15530684243210972302
        children[0]
                type='disk'
                id=0
                guid=17089111017021432710
                path='/dev/dsk/c5t0d0s0'
                devid='id1,sd@f834495254b12d0570002aeb30000/a'
                phys_path='/pci@8,700000/usb@5,3/storage@4/disk@0,0:a'
                whole_disk=1
                metaslab_array=14
                metaslab_shift=33
                ashift=9
                asize=1000191557632
                is_log=0
Uberblock

        magic = 0000000000bab10c
        version = 10
        txg = 1645
        guid_sum = 14173051186522853396
        timestamp = 1259524280 UTC = Mon Nov 30 04:51:20 2

Dataset mos [META], ID 0, cr_txg 4, 96.0K, 24 objects
Dataset tpool/testfs [ZPL], ID 21, cr_txg 136, 20.5K, 6 objects
Dataset tpool [ZPL], ID 5, cr_txg 4, 19.0K, 5 objects
                          capacity  operations    bandwidth   ---- errors ----
description             used avail   read write   read write   read write cksum
tpool                   144K  928G    156     0   186K     0      0     0     0
  /dev/dsk/c5t0d0s0     144K 928G    156     0   186K     0      0     0     0
root@act061 #
```

uberblock の
マジックナンバー

オブジェクトの
サマリ

図 4-2-2-1(4). zdb コマンドによる nvlist を含むストレージプール情報の表示

4-2-2-2. uberblock

uberblock は次の構造になっています。

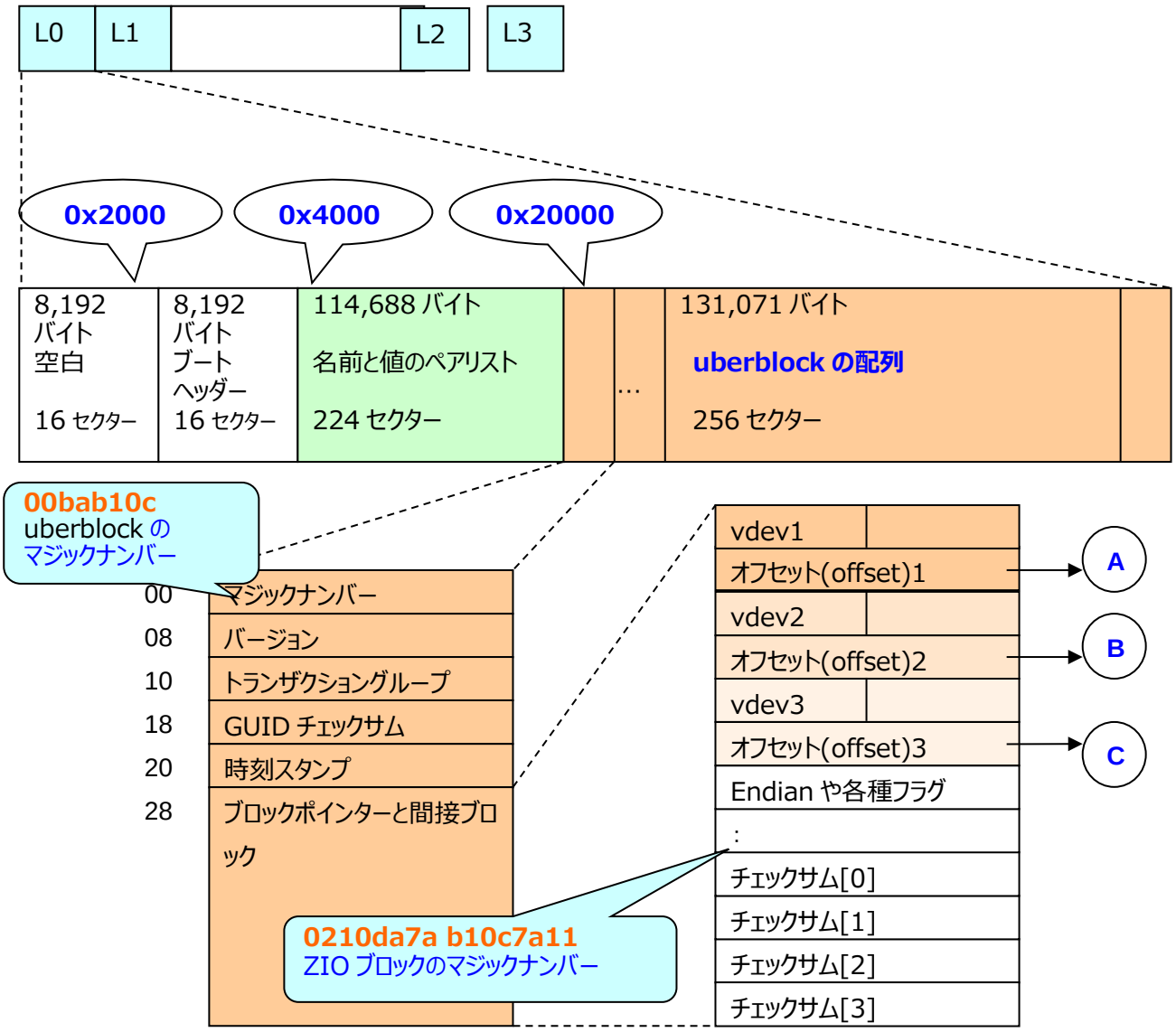


図 4-2-2-2(1). uberblock の構造

図の **A**、**B**、**C** は同じ内容のブロックポインターを示しています。vdev 部分は 32 ビットのユニークな vdevID です。物理ブロックアドレスの計算は次のように 63 ビットのオフセットを左に 9 ビットシフト(×512)した値に 0x400000(4MB)を加えて求めています。0x400000(4MB)は前述の L0 と L1 の長さです。

物理ブロックアドレス(physical block address)=(オフセット offset<<9)+0x400000(4MB)

この計算で求めたブロックに DMU(Data Management Unit)の配列があります。

次に uberblock のダンプ例を示します。

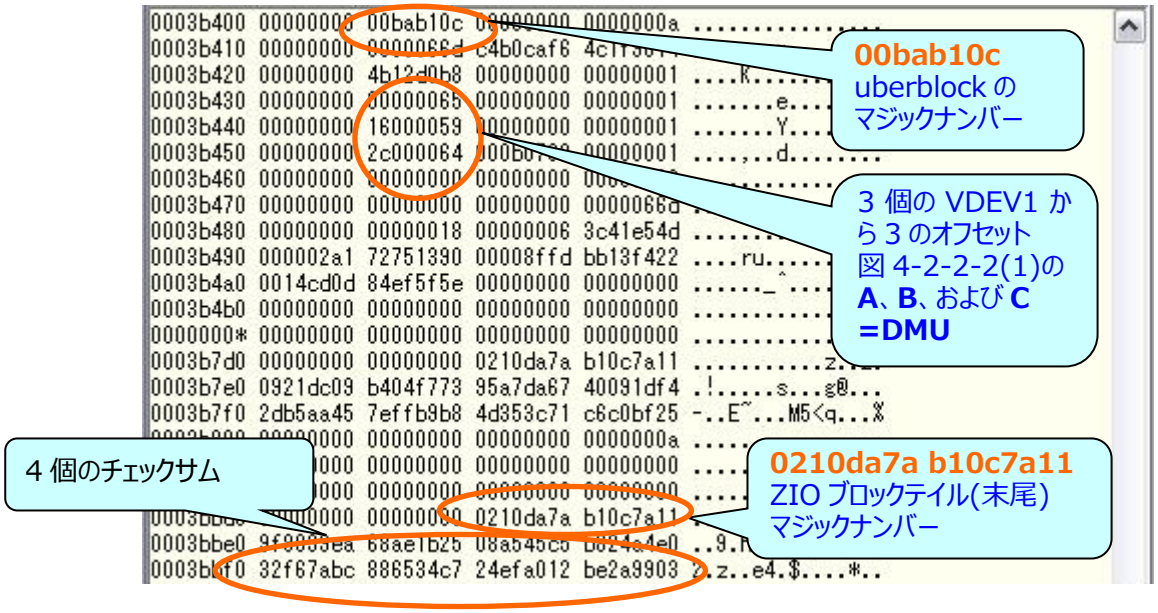


図 4-2-2-2(2). uberblock のダンプ例

DMU、**A**、**B**、および **C** のブロックが同一かを見てみます。

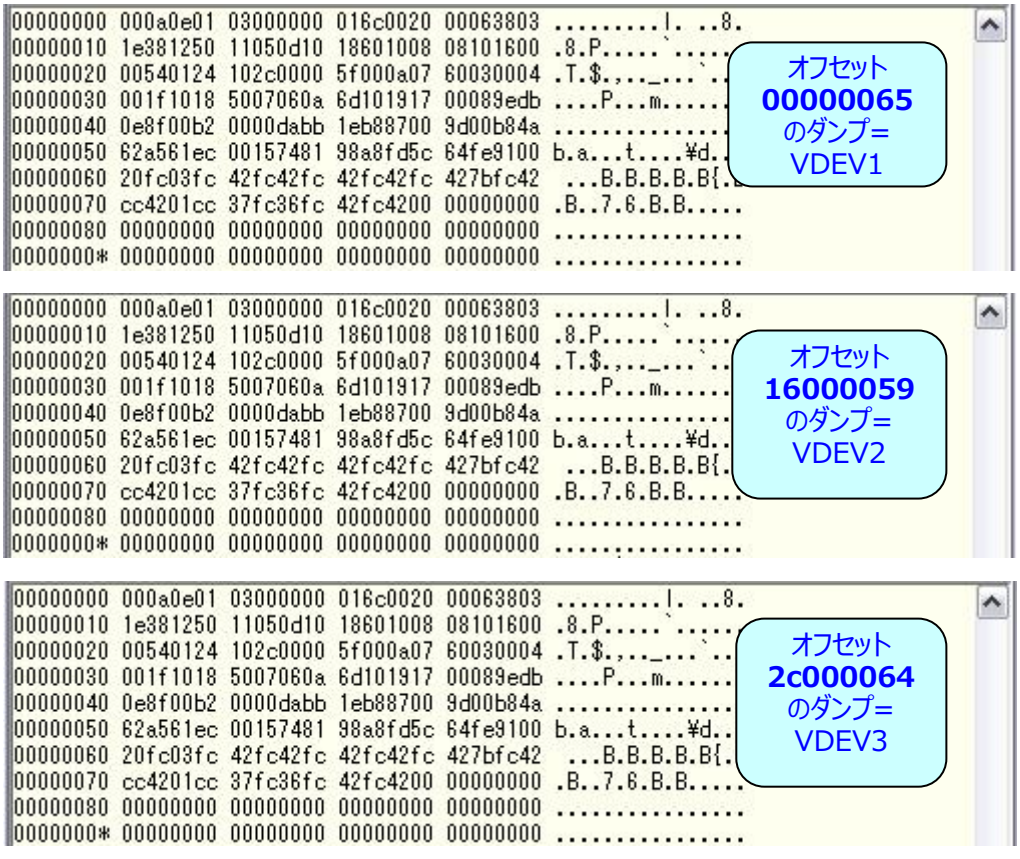


図 4-2-2-2(3). VDEV(DMU)が 3 個とも同じであるかを確認した例

DMU のアドレスは、zdb コマンドで出力することができます。

```
root@act061 # zdb -uuu tpool ⏎
Uberblock

        magic = 0000000000bab10c
        version = 10
        txg = 1645
        guid_sum = 14173051186522853396
        timestamp = 1259524280 UTC = Mon Nov 30 04:51:20 2009
        rootbp = [L0 DMU objset] 400L/200P DVA[0]=<0:ca00:200>
    DVA[1]=<0:2c0000b200:200> DVA[2]=<0:580000c800:200>
    fletcher4 lzjb BE contiguous birth=1645 fill=24
    cksum=63c41e54d:2a172751390:8ffdbb13f422:14cd0d84ef5f5e

root@act061 #
```

表示されるDMUブロックのアドレスは 9 ビット左にシフトされた値です。

図 4-2-2-2(4). zdb コマンドで DMU アドレスを表示する

4-2-2-3. DMU

DMU は、オブジェクトと呼ばれる論理ユニットです。各種のブロックやグループに使用されます。ZFS の内部では、全てがオブジェクトとして扱われています。

DMU の先頭バイトにはオブジェクトを示す型がセットされます。オブジェクト型を次に示します。

表 4-2-2-3(1). DMU のオブジェクトタイプ

10進	16進	オブジェクトタイプ	内容(**構造体の名前**)
0	0	DMU_OT_NONE	割り当てられていないオブジェクト
1	1	DMU_OT_OBJECT_DIRECTORY	DSL オブジェクトディレクトリの ZAP オブジェクト
2	2	DMU_OT_OBJECT_ARRAY	オブジェクトの配列
3	3	DMU_OT_PACKED_NVLIST	パックされた nvlist
4	4	DMU_OT_PACKED_NVLIST_SIZE	64 ビット長の nvlist の長さ
5	5	DMU_OT_BPLIST	ブロックポインタリスト（**blkptr_t**）
6	6	DMU_OT_BPLIST_HDR	BPLIST ヘッダ(**bplist_phys_t**)
7	7	DMU_OT_SPACE_MAP_HEADER	スペースマップヘッダ(**space_map_obj_t**)
8	8	DMU_OT_SPACE_MAP	SPA ディスクブロック使用リスト
9	9	DMU_OT_INTENT_LOG	インテントログ
10	a	DMU_OT_DNODE	メタブロック=dnode オブジェクト (**dnode_phys_t**)
11	b	DMU_OT_OBJSET	オブジェクトコレクション
12	c	DMU_OT_DSL_DIR	DSL ディレクトリオブジェクト(オブジェクトとデータセット間の関係とプロパティ定義：**dsl_dir_phys_t**)
13	d	DMU_OT_DSL_DIR_CHILD_MAP	小 DSL ディレクトリ情報の DSL ZAP オブジェクト
14	e	DMU_OT_DSL_DS_SNAP_MAP	スナップショット情報の DSL ZAP オブジェクト
15	f	DMU_OT_DSL_PROPS	DSL ディレクトリのプロパティを含む DSL ZAP オブジェクト
16	10	DMU_OT_DSL_DATASET	DSL データセットオブジェクト。スナップショットと、DMU_OT_OBJSET オブジェクト型の静的情報の使用に用いられる(**dsl_dataset_phys_t**)。
17	11	DMU_OT_ZNODE	ZPL メタデータ(**znode_phys_t**)
18	12	DMU_OT_ACL	ACL(アクセスコントロールリスト)オブジェクト
19	13	DMU_OT_PLAIN_FILE_CONTENTS	ZPL プレーンファイル
20	14	DMU_OT_DIRECTORY_CONTENTS	ZPL ディレクトリ ZAP オブジェクト
21	15	DMU_OT_MASTER_NODE	ZPL マスターノード ZAP オブジェクト
22	16	DMU_OT_UNLINKED_SET	
23	17	DMU_OT_ZVOL	ZFS ボリューム=ZVOL
24	18	DMU_OT_ZVOL_PROP	ZVOL のプロパティ
25	19	DMU_OT_PLAIN_OTHER	
26	1a	DMU_OT_UINT64_OTHER	
27	1b	DMU_OT_ZAP_OTHER	
28	1c	DMU_OT_ERROR_LOG	エラーログオブジェクト
29	1d	DMU_OT_SPA_HISTORY	SPA のヒストリーオブジェクト
30	1e	DMU_OT_SPA_HISTORY_OFFSETS	SPA のヒストリーオブジェクトオフセット
31	1f	DMU_OT_POOL_PROPS	
32	20	DMU_OT_NUMTYPES	

次は ZFS のオンディスクレイアウト全体を示した図です。目的のデータブロックに手作業で至るのは結構大変な手順になります。

【参考】ZFS On-Disk Layout - The Big Picture

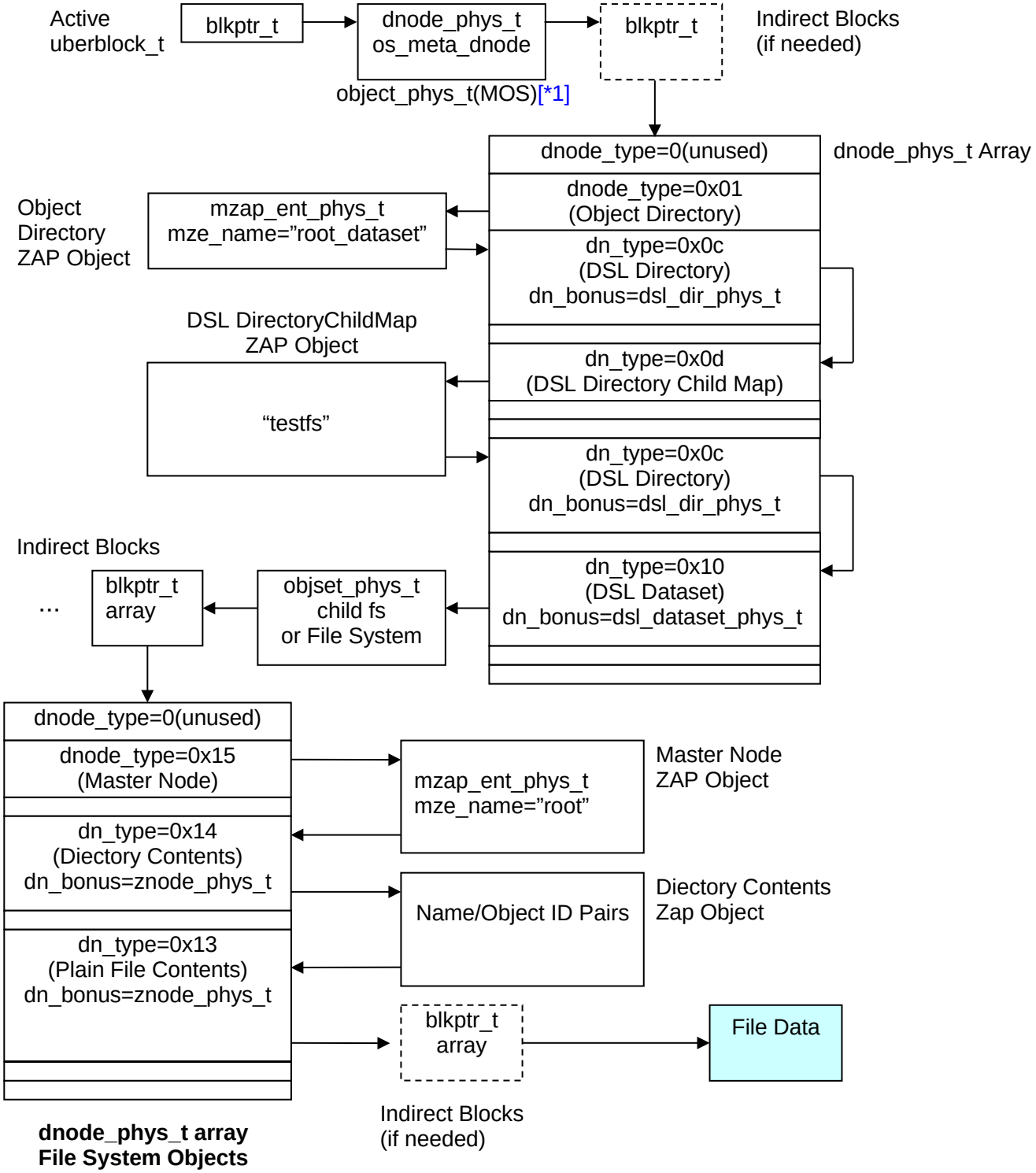


図 4-2-2-3(1). ZFS On-Disk Layout

出典：ZFS On-Disk Data Walk Sun Microsystems,Inc

[*1] MOS: Meta Object Set

【ディスク分析の大きな問題点】

各オブジェクトブロックから正確にブロックをポイントするためには、zdb コマンドの表示オプション"-R"の"d,lzjb,xxx"オプションが動作する必要があります。

現在の zdb コマンドは次のようにエラーになります(ZFS バージョン 15 も同様のエラーになる)。

```
root@act191:~# zdb -uuu rpool ⏎
Uberblock

        magic = 0000000000bab10c
        version = 14
        txg = 6410
        guid_sum = 11164885073417408878
        timestamp = 1259543315 UTC = Mon Nov 30 10:08:35 2009
        rootbp = [L0 DMU objset] 400L/200P DVA[0]=<0:760a7da00:200>
DVA[1]=<0:803e8200:200> DVA[2]=<0:ae01b8600:200> fletcher4 lzjb LE
contiguous birth=6410 fill=85
cksum=10d9867c6d:606a9954c45:11f4a65c3703a:250db89c5d655b

root@act191:~# zdb -R rpool:0:760a7da00:200:d,lzjb,400
2>/var/tmp/dba0dump ⏎
***Invalid flag arg: 'd,lzjb,400'
root@act191:~#
```

引数は、「DVA[0]のオフセット **760a7da00** の **512** バイトを **lzjb** 復元してダンプする」の意味です。

参考 WEB ではこの出力ファイルを入力にして、mdb コマンドの表示機能を使用しています。

ZFS では、データ部分の圧縮は conpression プロパティでセットします。しかしながら、ディスク内のメタデータや他の全てのオブジェクトは必ず圧縮されています。

このため、DMU を解読するには、lzjb 復元する必要があります。今回は次のツールを作成してディスク分析しています。

- DVA ブロック値を引数にして、目的のブロックをダンプして ASCII リストで表示する。
  (1) DVA ブロック値の引数に 8,192 バイトをプラス(前述の物理ブロックアドレス(physical block address)=(オフセット offset<<9)+0x400000(4MB)の計算と同じ意味です)。
  (2) (1)の値のブロックを dd コマンドで読み取り→"zdb-R"の出力と同じ機能のツール。
  (3) ASCII リストツールでテキストにして表示→"mdb"の表示と同じ機能(但し、構造体表示機能はありません)。
- dd コマンドで読み取ったデータを入力ファイルにして lzjb 復元し、出力ファイルに書き込む。
  (1) 入力ファイルの長さを得る(上記(2)で作成したファイル)。
  (2) lzjb 復元する(入力データが中途半端な長さ…読み取ったデータが短い場合はそこまで。読み取ったデータが長い場合、最後まで)。OpenSolaris lzjb.c ソースコードの lzjb_decompress 関数を参考にさせていただきました。
  (3) 復元データを出力ファイル(入力ファイルに".out"の名前を付ける)に書き込み。
  (4) ASCII リストツールでテキストにして表示→"mdb"の表示と同じ機能(但し、構造体表示機能はありません)。

ディスク分析に必要な構造体は次の通りです。

表 4-2-2-3(2)．ディスク分析に必要なオブジェクトと構造体

10進	16進	オブジェクトタイプ	内容(**構造体の名前**)
5	5	DMU_OT_BPLIST	ブロックポインタリスト(**blkptr_t**)
6	6	DMU_OT_BPLIST_HDR	BPLIST ヘッダ(**bplist_phys_t**)
7	7	DMU_OT_SPACE_MAP_HEADER	スペースマップヘッダ(**space_map_obj_t**)
10	a	DMU_OT_DNODE	メタブロック=dnode オブジェクト(**dnode_phys_t**)
12	c	DMU_OT_DSL_DIR	DSL ディレクトリオブジェクト(**dsl_dir_phys_t**)
16	10	DMU_OT_DSL_DATASET	DSL データセットオブジェクト (**dsl_dataset_phys_t**)
17	11	DMU_OT_ZNODE	ZPL メタデータ(**znode_phys_t**)

DMU は dnode と呼ばれる 512 バイトの構造体です。

次に dnode 構造を示します。他の構造体は割愛させていただきます。

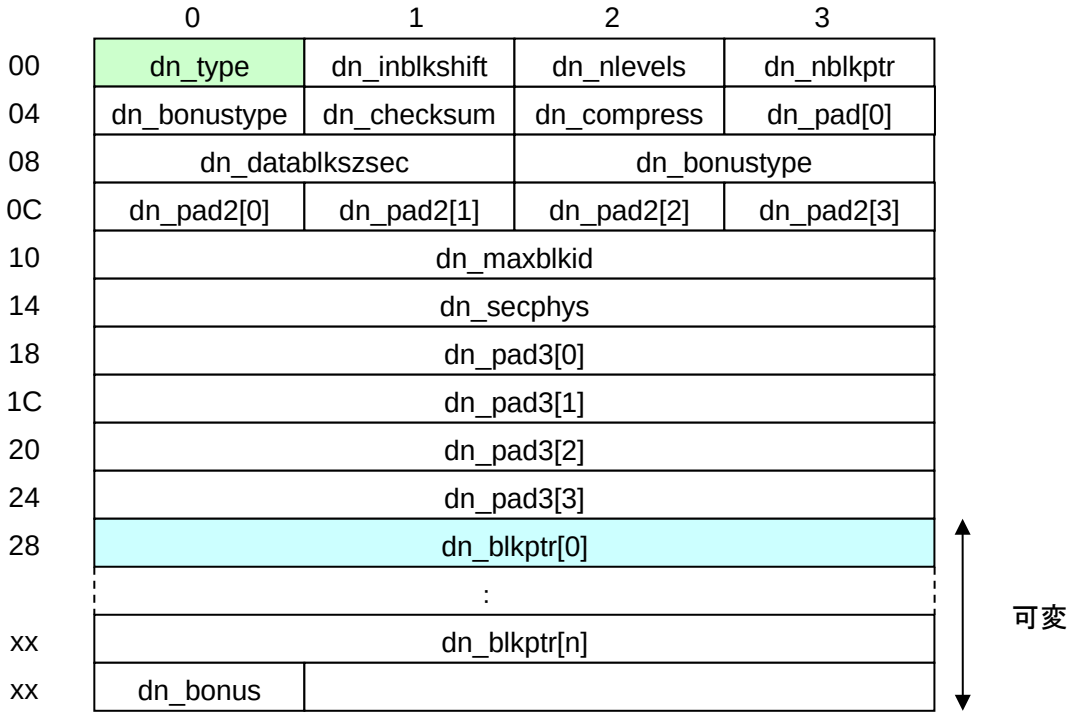
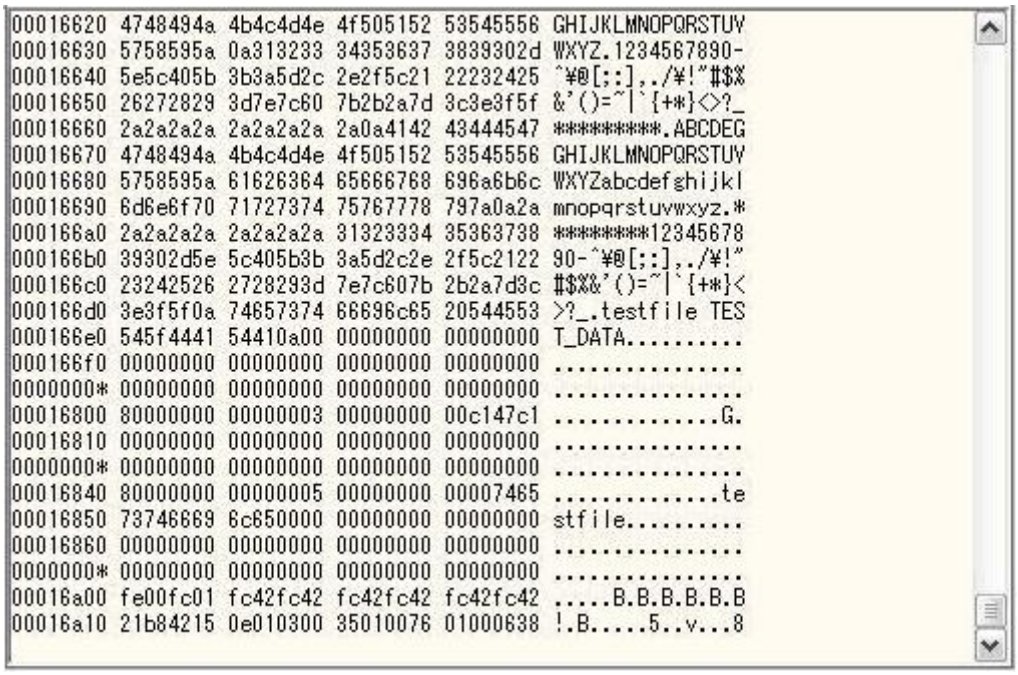


図 4-2-2-3(2)．dnode 構造体

dnode 構造体は lzjb で復元された結果、上図のように見ることができます。図 4-2-2-2(3)は lzjb で圧縮された形式です。先頭バイトの dn_type にはオブジェクトのタイプがセットされます。

目的のノードアドレスや構造体は dn_blkptr にセットされるポインタを辿っていくことになります(これは結構大変な作業です…)。

最終的な目的のデータブロックです。



```
00016620 4748494a 4b4c4d4e 4f505152 53545556 GHIJKLMNOPQRSTUV
00016630 5758595a 0a313233 34353637 3839302d WXYZ.1234567890-
00016640 5e5c405b 3b3a5d2c 2e2f5c21 22232425 ^¥@[;:],./¥!"#$%
00016650 26272829 3d7e7c60 7b2b2a7d 3c3e3f5f &'()=~|`{+*}<>?_
00016660 2a2a2a2a 2a2a2a2a 2a0a4142 43444547 **********.ABCDEG
00016670 4748494a 4b4c4d4e 4f505152 53545556 GHIJKLMNOPQRSTUV
00016680 5758595a 61626364 65666768 696a6b6c WXYZabcdefghijkl
00016690 6d6e6f70 71727374 75767778 797a0a2a mnopqrstuvwxyz.*
000166a0 2a2a2a2a 2a2a2a2a 31323334 35363738 ********12345678
000166b0 39302d5e 5c405b3b 3a5d2c2e 2f5c2122 90-^¥@[;:],./¥!"
000166c0 23242526 2728293d 7e7c607b 2b2a7d3c #$%&'()=~|`{+*}<
000166d0 3e3f5f0a 74657374 66696c65 20544553 >?_.testfile TES
000166e0 545f4441 54410a00 00000000 00000000 T_DATA..........
000166f0 00000000 00000000 00000000 00000000 ................
0000000* 00000000 00000000 00000000 00000000 ................
00016800 80000000 00000003 00000000 00c147c1 ..............G.
00016810 00000000 00000000 00000000 00000000 ................
0000000* 00000000 00000000 00000000 00000000 ................
00016840 80000000 00000005 00000000 00007465 ..............te
00016850 73746669 6c650000 00000000 00000000 stfile..........
00016860 00000000 00000000 00000000 00000000 ................
0000000* 00000000 00000000 00000000 00000000 ................
00016a00 fe00fc01 fc42fc42 fc42fc42 fc42fc42 .....B.B.B.B.B.B
00016a10 21b84215 0e010300 35010076 01000638 !.B.....5..v...8
```

図 4-2-2-3(3). データブロックのダンプ

4-3. ブートディスクの構造

ブートディスクは、SMI ディスクラベル(VTOC)形式のシリンダー単位にフォーマットされ、ブートパーティションにブートローダーがセットされる点が通常のデータディスクとの違いです。データディスクは EFI ディスクラベル形式(セクター単位)です。

```
root@opensolaris:/var/tmp# format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c8t0d0 <DEFAULT cyl 8921 alt 2 hd 255 sec 63>
          /pci@0,0/pci10de,375@f/pci1000,3150@0/sd@0,0
      :
Specify disk (enter its number): 0 ⏎
selecting c8t0d0
[disk formatted]
/dev/dsk/c8t0d0s0 is part of active ZFS pool rpool. Please see zpool(1M).
      :
partition> print ⏎
Current partition table (original):
Total disk cylinders available: 8921 + 2 (reserved cylinders)
      :
Part      Tag    Flag     Cylinders         Size            Blocks
  0       root    wm       1 - 8920        68.33GB    (8920/0/0) 143299800
  1 unassigned    wm       0                   0         (0/0/0)         0
  2     backup    wu       0 - 8920        68.34GB    (8921/0/0) 143315865
  3 unassigned    wm       0                   0         (0/0/0)         0
  4 unassigned    wm       0                   0         (0/0/0)         0
  5 unassigned    wm       0                   0         (0/0/0)         0
  6 unassigned    wm       0                   0         (0/0/0)         0
  7 unassigned    wm       0                   0         (0/0/0)         0
  8       boot    wu       0 -     0        7.84MB    (1/0/0)         16065
  9 unassigned    wm       0                   0         (0/0/0)         0
```

図 4-3-1. ブートディスクのフォーマット出力例

```
partition> quit ⏎
format> quit ⏎
root@act191:~#

root@act191: ~# dd if=/dev/dsk/c7d0s8 count=16065 of=boot ⏎
16065+0 records in
16065+0 records out
8225280 bytes (8.2 MB) copied, 0.176245 s, 46.7 MB/s
root@act191: ~# file boot ⏎
boot:               DOS executable (COM) ←ブートローダーと GRUB がセットされています。
root@act191: ~#
```

図 4-3-2. ブートパーティションのデータを確認する

図 4-3-1 を図解すると次のようになります。

シリンダ 0　　名前：boot パーティション番号 8 → スライス 8
シリンダ 1～8920　名前：root パーティション番号 0 → スライス 0

CYL	CYL
0	1　　8920
boot	root

図 4-3-3. システムディスクのパーティション図解

4-4. ZFS チューニング

NAS アプライアンスのチューニングをテストしましたが、シーケンシャルアクセスのためか、効果は見られませんでした。次に ZFS のチューニングパラメタを示します。

表 4-4-1. ZFS チューニングパラメタ

No	パラメタ	実際の値	内容
1	arc_reduce_dnlc_percent	3	
2	arc_shrink_shift	5	
3	fzap_default_block_shift	14	
4	metaslab_aliquot	0	
5	metaslab_gang_bang	0	
6	spa_max_replication_override	3	
7	spa_mode_global	3	
8	vdev_mirror_shift	21	
9	zfetch_array_rd_sz	1048576	
10	zfetch_block_cap	256	ZFS プリフェッチサイズ(NAS：16)
11	zfetch_max_streams	8	ZFS プリフェッチストリーム数(NAS：2)
12	zfetch_min_sec_reap	2	
13	zfs_arc_max	0	
14	zfs_arc_min	0	
15	zfs_arc_grow_retry	0	ARC チューニング(NAS：300)
16	zfs_arc_shrink_shift	0	ARC チューニング(NAS：7)
17	zfs_default_bs	9	
18	zfs_default_ibs	14	
19	zfs_flags	0	
20	zfs_immediate_write_sz	32768	
21	zfs_mdcomp_disable	0	
22	zfs_no_scrub_io	0	
23	zfs_no_write_throttle	0	
24	zfs_nocacheflush	0	キャッシュフラッシュ制御
25	zfs_prefetch_disable	0	プリフェッチ制御
26	zfs_read_chunk_size	1048576	
27	zfs_scrub_limit	10	
28	zfs_txg_synctime	5	トランザクション同期時間(NAS：1)
29	zfs_txg_timeout	30	
30	zfs_vdev_aggregation_limit	131072	
31	zfs_vdev_cache_bshift	16	
32	zfs_vdev_cache_max	16384	
33	zfs_vdev_cache_size	10485760	
34	zfs_vdev_max_pending	35	
35	zfs_vdev_min_pending	4	
36	zfs_vdev_ramp_rate	2	
37	zfs_vdev_time_shift	6	
38	zfs_write_limit_max	133045760	527236096 の OpenSolaris もありました。
39	zfs_write_limit_min	33554432	

40	zfs_write_limit_override	0	
41	zfs_write_limit_shift	3	
42	zil_disable	0	インテントログ制御
43	zio_injection_enabled	0	
43	zvol_immediate_write_sz	32768	

【備考 1】 内容欄に記述のないものは **T.B.D** とさせていただきます。
【備考 2】 パラメタでテストしたものは内容欄に「(NAS：値)」を付記しています。
【備考 3】 パラメタに下線があるものは WEB 等で発見したパラメタです。
【備考 4】 ZFS Version 10、14、15 と OpenSolaris ソースコードの zfs_params と違いがあります。
【備考 5】 OpenSolaris 2009.09 の値(ZFS Version 14)を mdb で確認しました。環境によって変化する値もあるようですが、NAS アプライアンスのチューニングよりも進んでいるように感じました。現在のところ、特に変更しなくてはならないパラメタは無いと考えます。なお、ZFS チューニングに関しては今後とも調査を継続します。

次章の性能測定結果から、シーケンシャルファイルアクセスに関しては特にチューニングは不要と考えます。ランダムアクセスや、ログ関連のチューニングは今後の検討課題です。

第 5 章 性能

5-1. 性能測定の条件

ZFS のパフォーマンスを測定しました。今回のポイントは、「ZFS ファイルシステムの構成によってトータルスループットがどのように違うのか」を計測しました。アクセス方法はシーケンシャルアクセスです。

なお、ランダムアクセスにつきましては、未実施です。

測定機器は次のとおりです。

表 5-1-1. 性能測定の機器

No	コンポーネント	内容
1	サーバー	Sun Fire X4140 サーバー
2	CPU	Quad-Core AMD Opteron™ Processor 2.356GH z CPU × 2
3	メモリ	4GB
4	内蔵ディスク 2.5 インチ SAS	No.0 システムディスク： 73GB 10,000rpm No.1 データディスク： 73GB 10,000rpm No.2 データディスク： 73GB 10,000rpm No.3 データディスク： 73GB 15,000rpm No.4 データディスク： 73GB 15,000rpm No.5 データディスク： 73GB 15,000rpm } 各種 ZFS 構成でテスト

性能測定ツールは次のとおりです。

表 5-1-2. 性能測定ツール

No	区分	コマンド	目的
1	測定	timex	エラプス、システム CPU、およびユーザーCPU の消費時間
2	測定	sar 等	各種性能情報(sar、iostat、vmstat、pmap、pacct など)
3	実行	dd	データをシーケンシャルにコピー
4	実行	rm	コピーデータのクリア
5	実行	sleep	dd のコピーが終了することを確認するための待ち

試験シナリオを次に示します。

表 5-1-3. テストシナリオ

No	ステップ	内容
1	出力ファイル作成	システムディスクイメージを 16GB(17,179,869,184 バイト)切り出し
2	入力ファイル作成	No.1 のデータをファイルシステムにコピー。以上は事前に設定
3	ブロックサイズ n で出力	システムディスクから 16GB ファイルをファイルシステムに書き込み
4	ブロックサイズ n で入力	ファイルシステムの 16GB ファイルを/dev/null に読み込み(No.3 と同時)
5	No.3 出力ファイル削除	メモリキャッシュをクリア
6	5 秒 sleep	dd コマンド終了待ちを確認
7	15 秒 sleep	No.3、No.4 両方の dd コマンドが終了したとき
8	ブロックサイズを×2 倍	ブロックサイズ 1,024 バイトからスタートし、8,388,608 バイトまで増加

※No.3～8 を、ブロックサイズを 1K～8MB まで変化させて計 14 回繰り返し。

テスト時、各プログラムは次のように動作します。

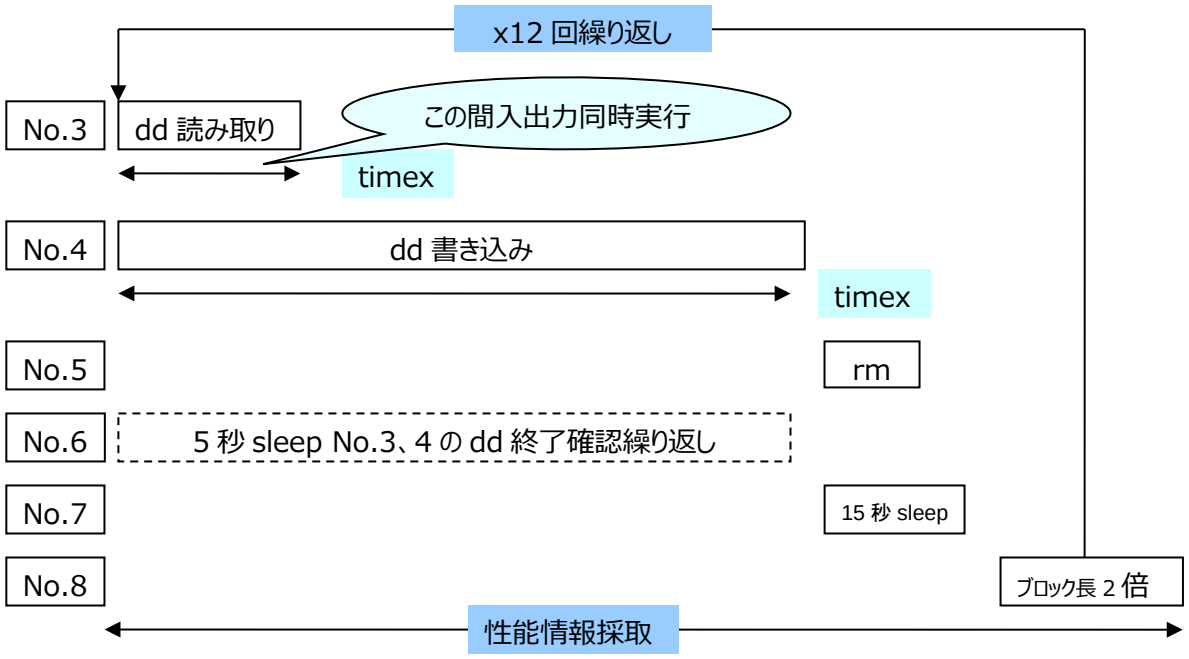


図 5-1-1. 測定プログラムの流れ

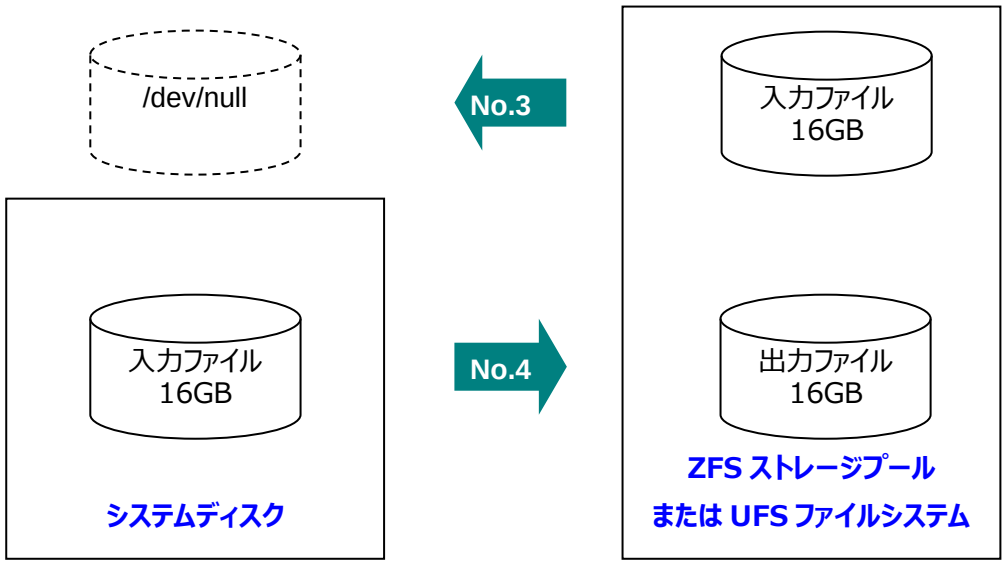


図 5-1-2. ファイルの位置とデータコピー

次の構成パターンで性能測定しました。

表 5-1-4. ZFS 性能測定の構成

No	方式	RAID 設定	UFS	ZFS	RAID Z	目的
1	JBOD	-	-	○	○	ZFS 基本性能測定
2	HW RAID	RAID 0	○	○	-	ハードウェア RAID との比較
3	SW RAID	RAID 0	○	○	-	ソフトウェア RAID との比較
4		RAID 5	○	○	-	

5-2. ZFS と UFS、SVM、ハードウェア RAID の比較

次に RAID 構成と ZFS のスループットの比較を示します。列の右側ほど高い値が出ていることを示します。ZFS の性能の良さが現れています。HW RAID0 ZFS で良い値が出ています【備考】。

表 5-2-1. スループットの比較(単位：MB/秒)

区分	項目	SVM RAID5 UFS	SVM RAID0 UFS	SVM RAID5 ZFS	HW RAID0 UFS	RAIDZ	ZFS	SVM RAID0 ZFS	HW RAID0 ZFS
最小値	ライト	3	37	27	49	40	40	42	40
	リード	20	55	37	42	89	100	100	100
	合計	29	92	64	91	129	140	142	141
平均値	ライト	10	46	28	75	75	76	78	76
	リード	23	79	99	73	209	299	305	316
	合計	34	125	127	148	285	375	383	391
最大値	ライト	13	49	30	81	81	81	82	81
	リード	31	82	121	94	242	333	340	362
	合計	36	131	150	176	322	414	422	440

次の図は最大値の合計スループットをグラフにしたものです。差が歴然で UFS の倍以上の性能値です。ZFS RAID Z のライトはパリティ計算のためにやや遅くなっています。

ハードウェア、ソフトウェア RAID にかかわらず、ZFS はリードスループットで良好な値が出ています。

なお、“ZFS”は、ストレージプール作成時のデフォルトで RAID 0 相当、“RAIDZ”はストレージプール作成時に“raidz”パラメタを指定して作成するもので、RAID 5 相当です。

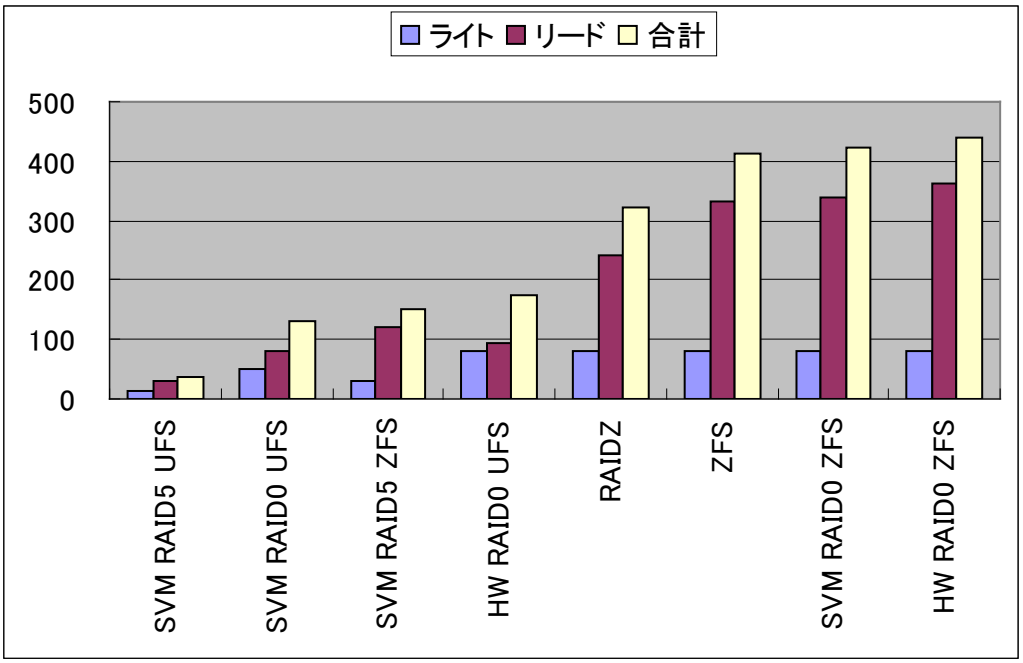


図 5-2-1. スループットの比較(リードライトの合計 単位：MB/秒)

【備考】だからといって、ここで HW RAID0 に流れてはだめです。システムリソースの消費状況を総合的、かつ的確に評価する必要があります。以降に、CPU、メモリ、ディスク入出力の振る舞いを、順を追って解説します。

ブロックサイズによるトータルスループットの変化を次に示します。8,192 バイトを超えるあたりから頭打ちとなっています。

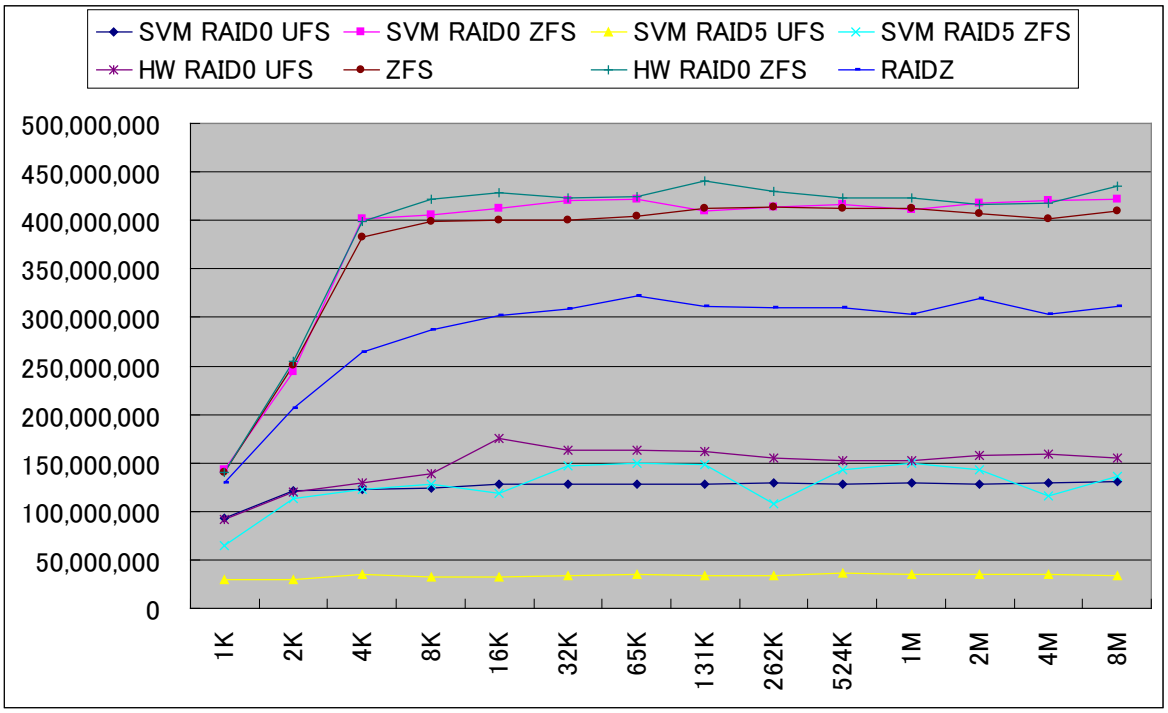


図 5-2-2. ブロックサイズの違いによるトータルスループットの変化(単位：バイト/秒)

変わった所見が見られました。SVM RAID 5 の ZFS 構成です。RAID 5 の ZFS ストライプサイズと、SVM ストライプサイズが干渉していると見られるユニークなグラフです。

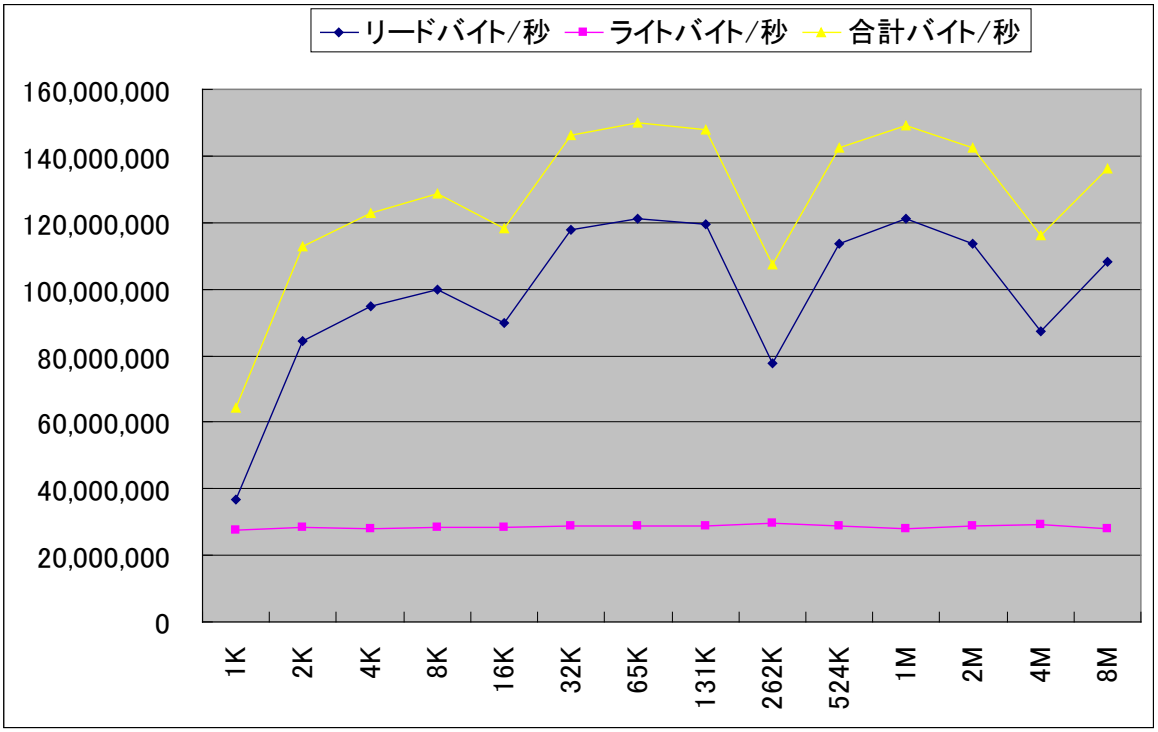


図 5-2-3. SVM RAID 5 と ZFS のストライプ干渉と見られるグラフ(単位：バイト/秒)

5-3. CPU、メモリ、およびディスクの入出力性能情報

5-3-1. CPU

SVM RAID5 UFS は低いスループットで、CPU 稼働は僅少です。

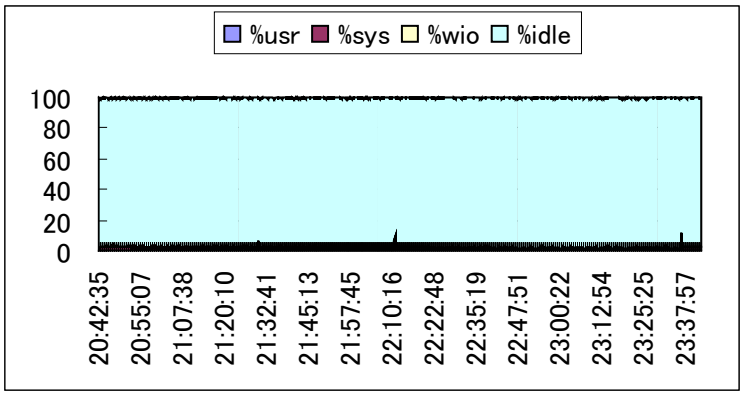


図 5-3-1-1. SVM RAID5 UFS の CPU 消費

SVM RAID0 UFS では、小さいブロックサイズのファイルコピーで CPU 消費が約 20%に上がります。

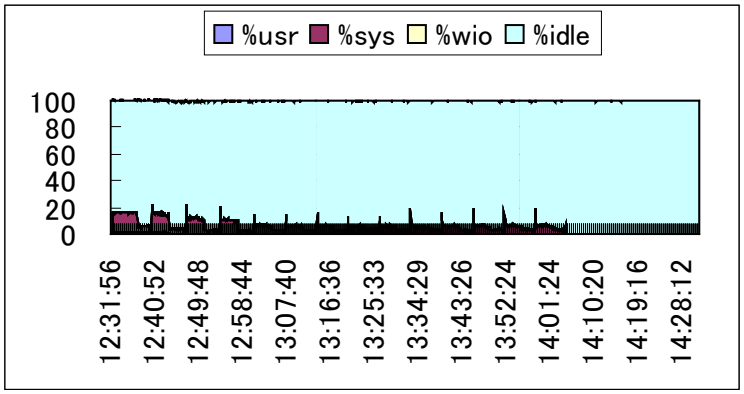


図 5-3-1-2. SVM RAID0 UFS の CPU 消費

SVM RAID5 ZFS は SVM のパリティ計算で CPU 消費が 20%を超えます。

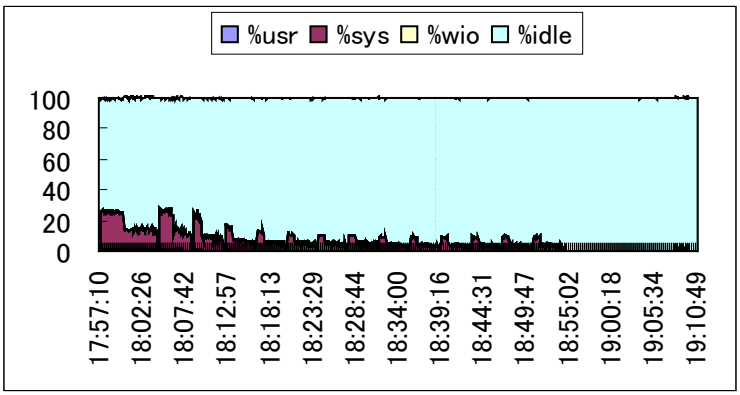


図 5-3-1-3. SVM RAID5 ZFS の CPU 消費

HW RAID0 では純粋に UFS の処理で CPU が消費されていると見られます。

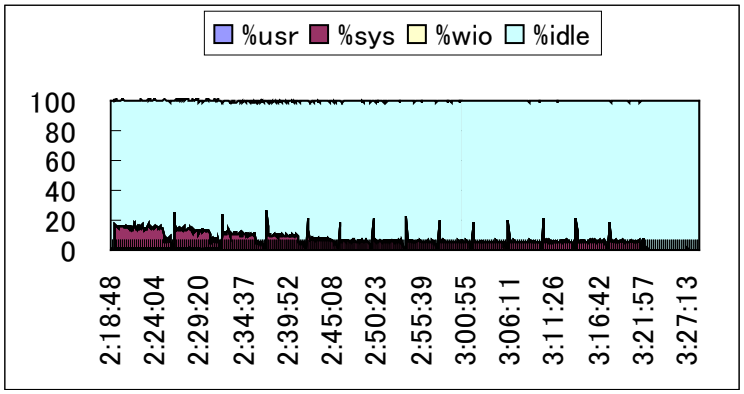


図 5-3-1-4. HW RAID0 UFS の CPU 消費

RAIDZ の CPU 消費は約 30%を超えることがあります。パリティ処理と考えられます。一方、SVM や、HW RAID 0 の UFS よりも大幅にコピー時間が短くなっており、スループットが高くなっています。ディスクアクセスが効率的に行われているもようです(テストを 2 セット実行しています)。

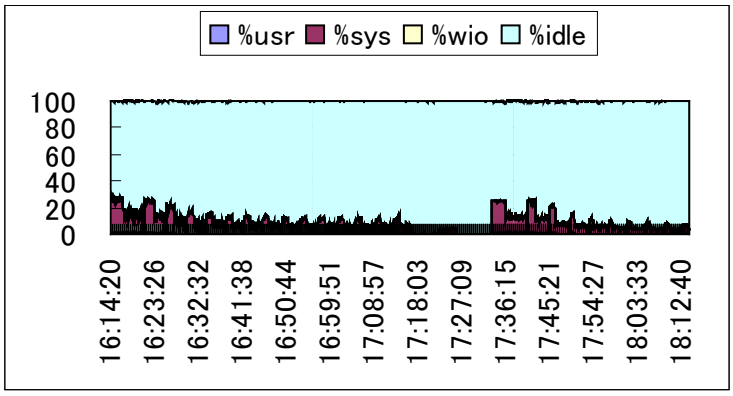


図 5-3-1-5. RAIDZ の CPU 消費

シンプルな ZFS 構成です。CPU 消費はパリティ計算が無い分、RAIDZ に比べて少なくなっています。

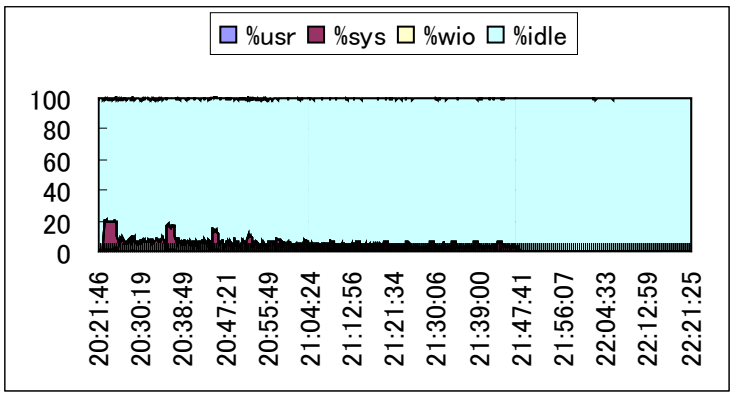


図 5-3-1-6. ZFS の CPU 消費

SVM RAID0 ZFS では、SVM の CPU 消費が上乗せされているもようです。

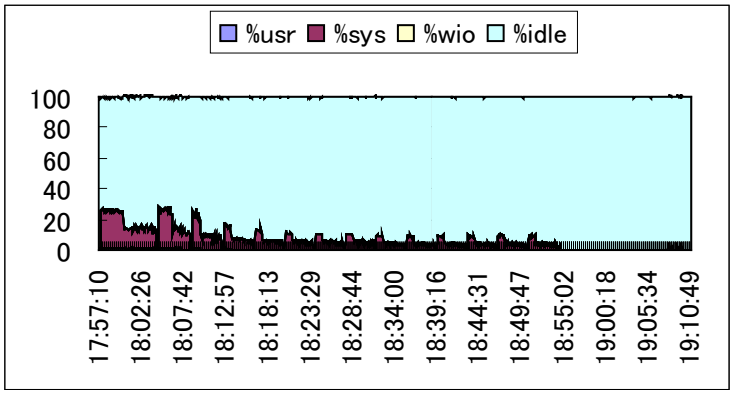


図 5-3-1-7. SVM RAID0 ZFS の CPU 消費

HW RAID0 ZFS で、最も高いスループットが記録されましたが、CPU 消費が大きい傾向にあります。

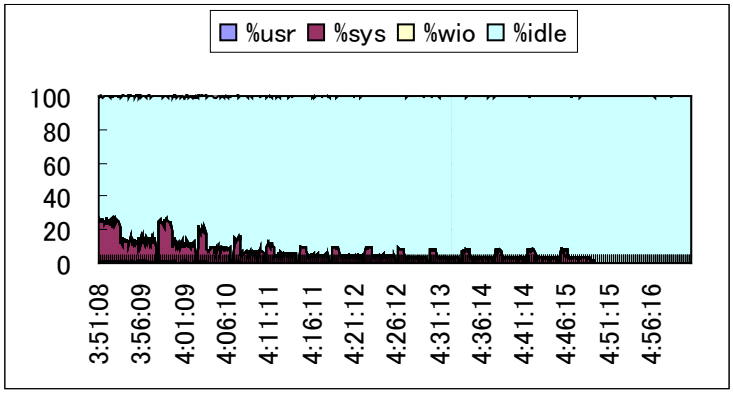


図 5-3-1-8. HW RAID0 ZFS の CPU 消費

5-3-2. メモリ

本性能評価ではファイル数の増減が無く、固定のファイル名を使用したため、メモリ使用状況に顕著な差異は見られませんでした。次にファイルシステム構成による空きメモリと空きスワップの情報を示します。

表 5-3-2-1. freemem と freeswap

No	ファイルシステムの構成	freemem【備考 1】 freeswap【備考 2】	最小値	最大値	平均値
1	SVM RAID5 UFS	freemem	24,772	126,180	46,862
		freeswap	4,141,021	4,513,354	4,443,760
2	SVM RAID0 UFS	freemem	27,744	144,315	92,817
		freeswap	4,183,952	4,479,182	4,285,408
3	SVM RAID5 ZFS	freemem	126,371	149,500	144,041
		freeswap	4,145,050	4,329,974	4,286,495
4	HW RAID0 UFS	freemem	24,901	145,139	76,844
		freeswap	4,149,370	4,588,659	4,309,501
5	RAIDZ	freemem	128,288	192,092	144,817
		freeswap	4,155,907	4,666,434	4,289,831
6	ZFS	freemem	653,009	707,865	693,465
		freeswap	8,349,934	8,788,486	8,673,818
7	SVM RAID0 ZFS	freemem	126,371	149,500	144,041
		freeswap	4,145,050	4,329,974	4,286,495
8	HW RAID0 ZFS	freemem	124,645	183,077	143,595
		freeswap	4,131,670	4,599,061	4,282,939

【備考 1】freemem(空きメモリページ)の単位はページで、4,096 バイトです。
【備考 2】freeswap(空きスワップブロック)の単位はブロックで、512 バイトです。

本システムの場合、メモリの freemem が 16,348 ページを下回るとページングが発生します。表中、最小値でもそれに至っていないため、ページングの特徴的な振る舞いは発見できませんでした。カーネルメモリの所見も大きな違いはありませんでした。

ZFS ではファイル操作を多く実行するとカーネルメモリを消費するようです。次に別環境でのメモリ所見をご紹介します。ZFS 内にファイルを大量に作成することでカーネルメモリが消費され、freemem と freeswap が減少していく様子を示しています。

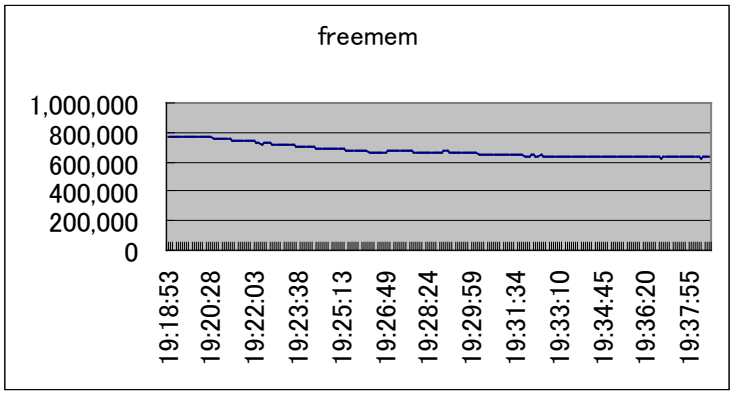


図 5-3-2-1. ファイル大量割り当て時の freemem

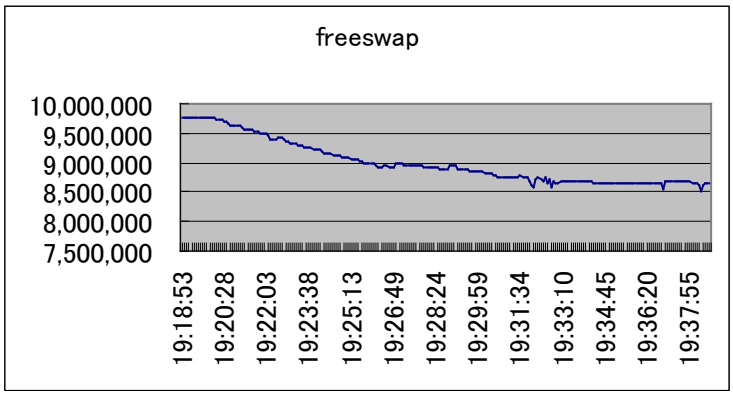


図 5-3-2-2. ファイル大量割り当て時の freeswap

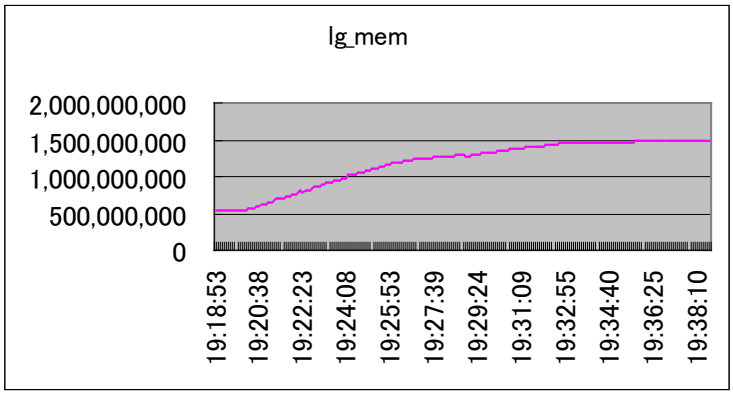


図 5-3-2-3. ファイル大量割り当て時のラージカーネルメモリ

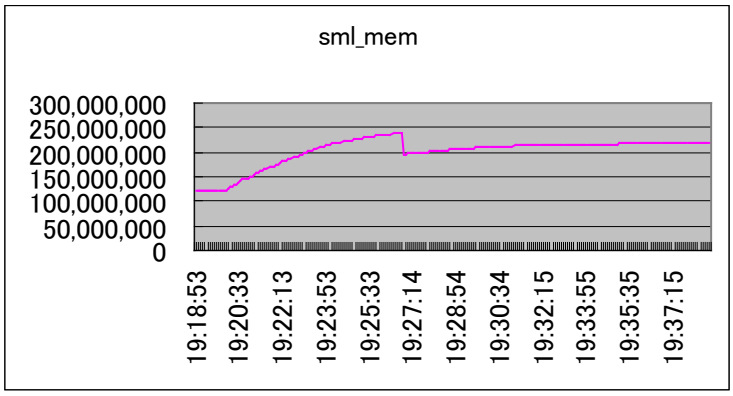


図 5-3-2-4. ファイル大量割り当て時のスモールカーネルメモリ

表 5-3-2-2. ZFS のメモリキャッシュ消費量

	cache name	buff size	buff in use	buff total	memory in use	alloc succeed	alloc fail
前	zfs_znode_cache	192	36,129	36,220	7,417,856B	43,531	0
後		192	129,089	129,980	26,619,904B	449,955	0

表 5-3-2-3. ZFS の仮想メモリ消費量

	vmem name	memory in use	memory total	memory import	alloc succeed	alloc fail
前	zfs_file_data_buf	155,279,360B	155,279,360B	155,279,360B	3,647	0
後		160,137,216B	160,137,216B	160,137,216B	4,833	0

メモリキャッシュはデータ量が多いと必然的に増加します。カーネルメモリは ZFS 関連の構造体以外に、ファイルテーブル、仮想ノードなど多くのメモリを使用します。本説明書では割愛させていただきます。
なお、ZFS ライブラリも確認しましたが僅少なサイズ(約 5MB 程度)のため、割愛させていただきます。

次にメモリのまとめとして、HW RAID0 UFS と ZFS のメモリ割り当て状況を示します。
ほぼ同等のメモリ割当状況と考えて良いです。

表 5-3-2-4. メモリ割り当て状況

No	項目	HW RAID0 UFS		ZFS	
		ページ【備考 1】	バイト	ページ【備考 1】	バイト
1	構成物理メモリ【備考 2】	1,048,379	4,294,160,384	1,048,379	4,294,160,384
2	物理メモリ【備考 3】	1,046,330	4,285,767,680	1,046,330	4,285,767,680
3	UNIX(カーネル)	2,049	8,392,704	2,049	8,392,704
4	平均カーネルメモリ	940,519	3,852,365,824	361,925	1482444,800
5	ページ構造体アレイ	16,380	67,092,480	16,380	67,092,480
6	平均空きメモリ	76,844	314,753,024	693,465	2,840,432,640
7	ユーザメモリ	16,922	69,312,512	18,460	75,612,160
8	平均空きスワップ	538,687	2,206,461,952	1,084,227	4,440,993,792

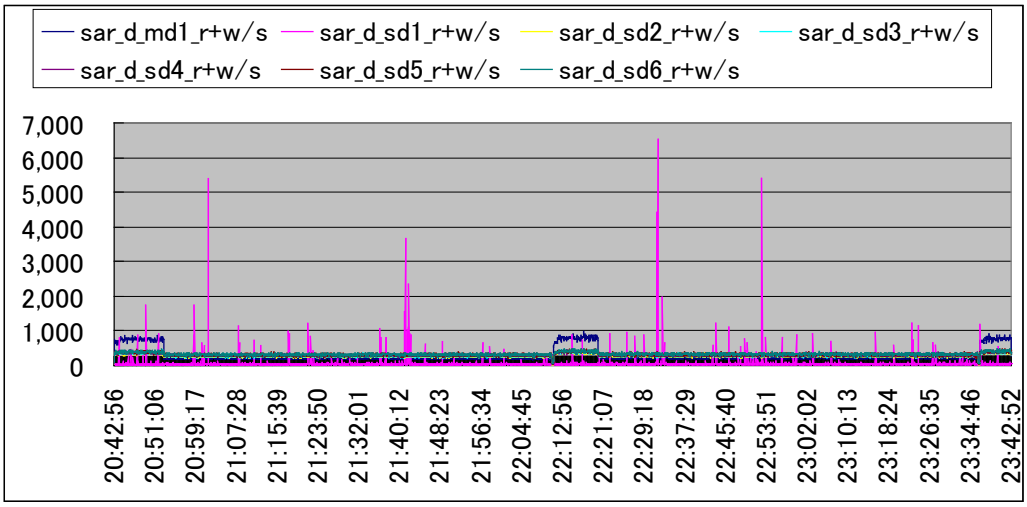
【備考 1】ページは 4,096 バイト 1 ページです。
【備考 2】physinstalled のカーネル変数にセットされる値です。
【備考 3】physmem のカーネル変数にセットされる値です。physinstalled からこの physmem を引き算すると UNIX(カーネル) の サイズがわかります。

【参考】ZFS メモリキャッシュと仮想メモリ消費量は、“mdb -k”コマンドの“：：kmastat”ファンクションで表示することができます。また、各種のカーネル変数も mdb コマンドで表示することができます。

5-3-3. ディスク

SVM RAID5 UFS では、メタデバイス(md1=mdd50)のビジーが高くなっています。

rw/s



busy

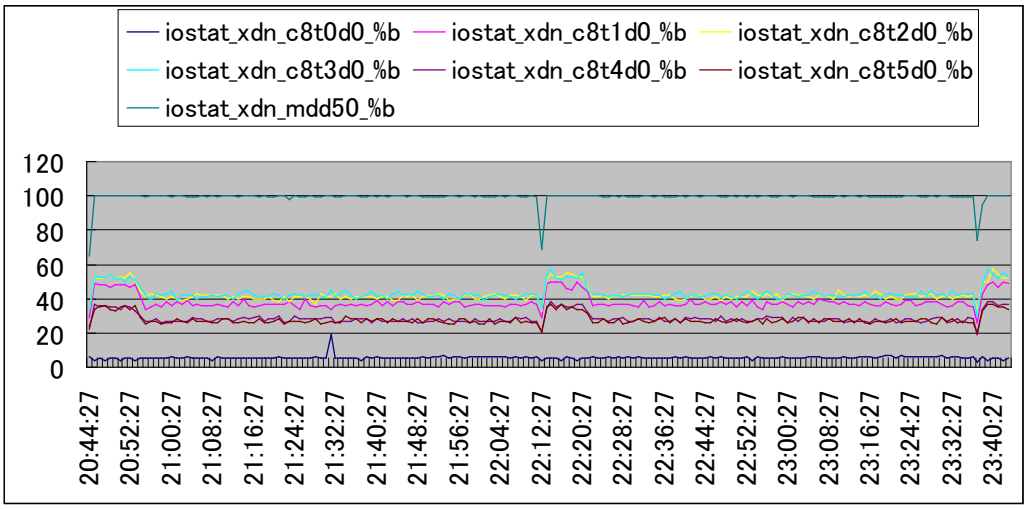
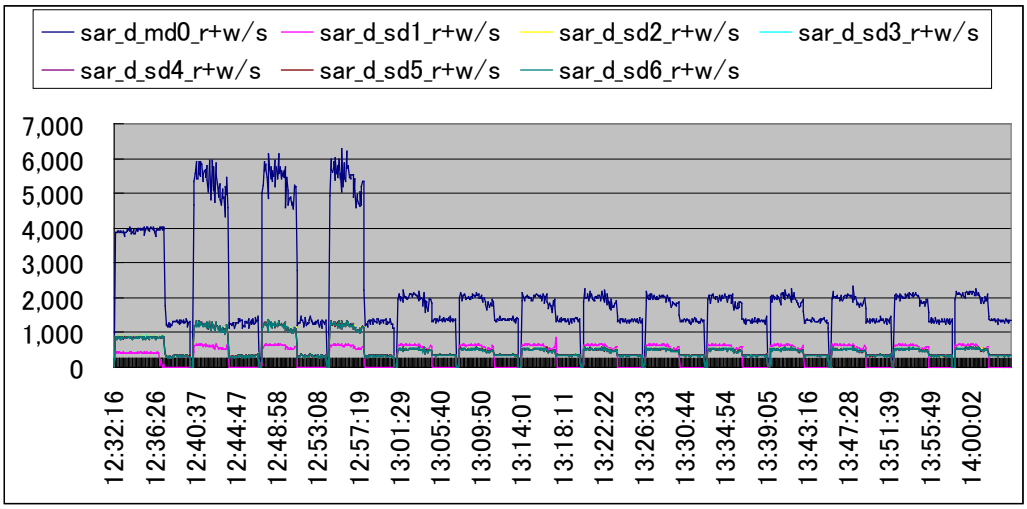


図 5-3-3-1. SVM RAID5 UFS の秒あたりリードライト回数とディスクビジー

【備考】 sarコマンドの%busyはCPUを合計して値を表示していることがわかりました。例えば"600%"のように表示されます。このため、本解説書では、iostatコマンドの%bを用います。デバイス名表記は、sarコマンドはsdx、やmdx、iostatコマンドは cntnd0と表示されます。

SVM RAID0 UFS では小ブロックサイズで、高いアクセス数が記録されています。

rw/s



busy

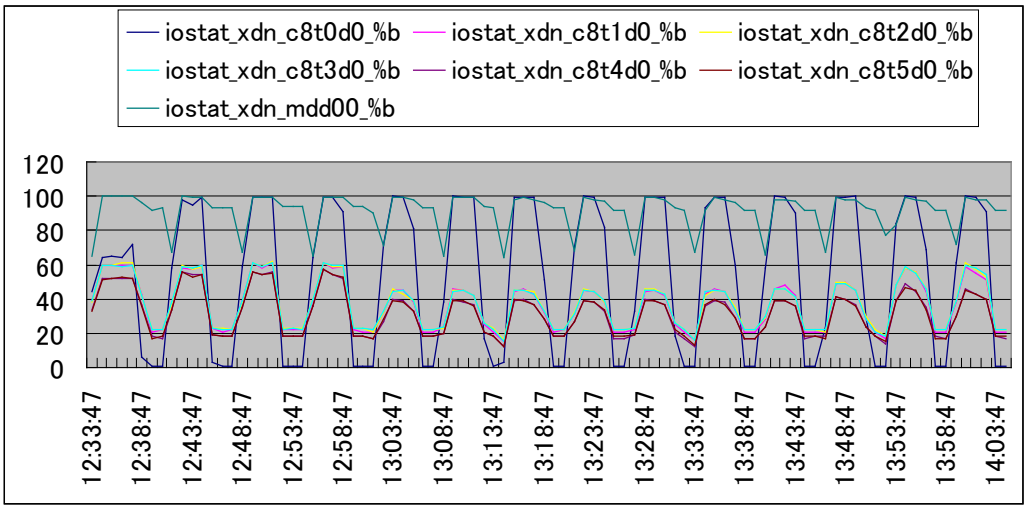
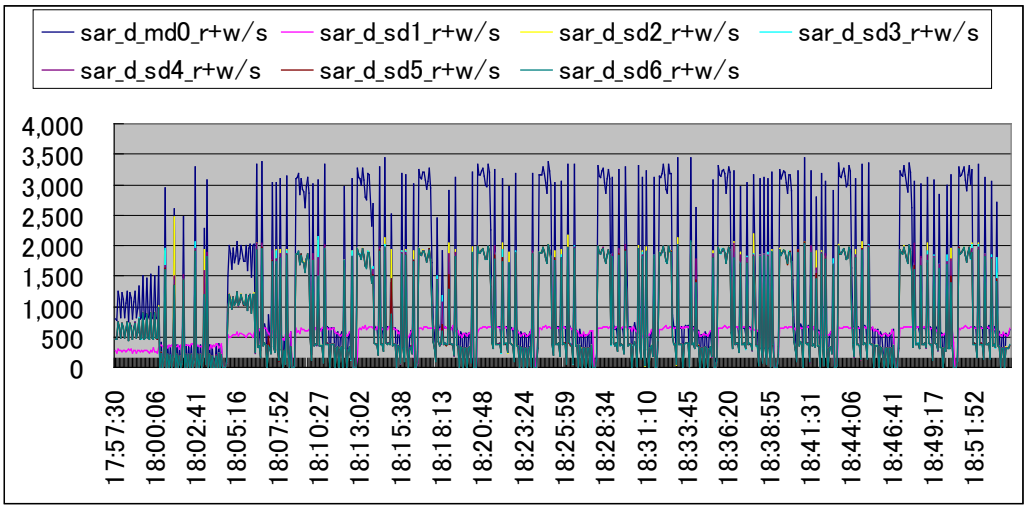


図 5-3-3-2. SVM RAID0 UFS の秒あたりリードライト回数とディスクビジー

SVM RAID5 ZFS は、ZFS のアクセスと SVM が競合しているように見えます。ディスクビジーはブロックサイズが 8KB を超えるころから 100%近くになっています。

rw/s



busy

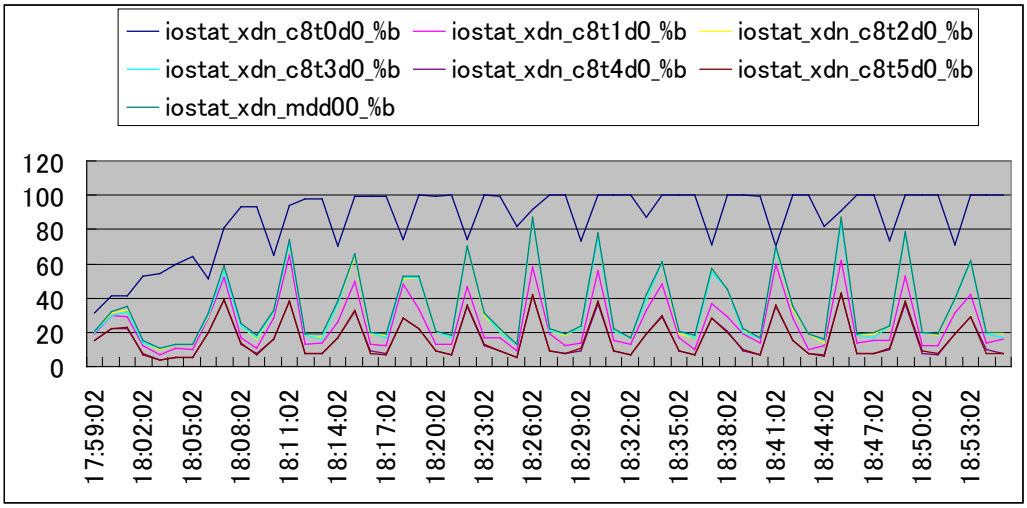
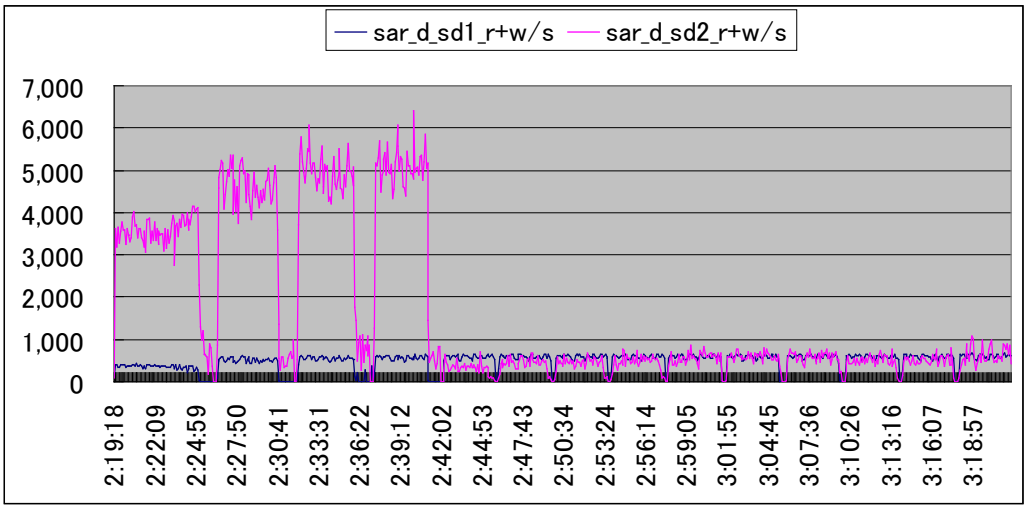


図 5-3-3-3. SVM RAID5 ZFS の秒あたりリードライト回数とディスクビジー

HW RAID のため、採取情報が sd1 と sd2 の 2 ドライブのみです。

rw/s



busy

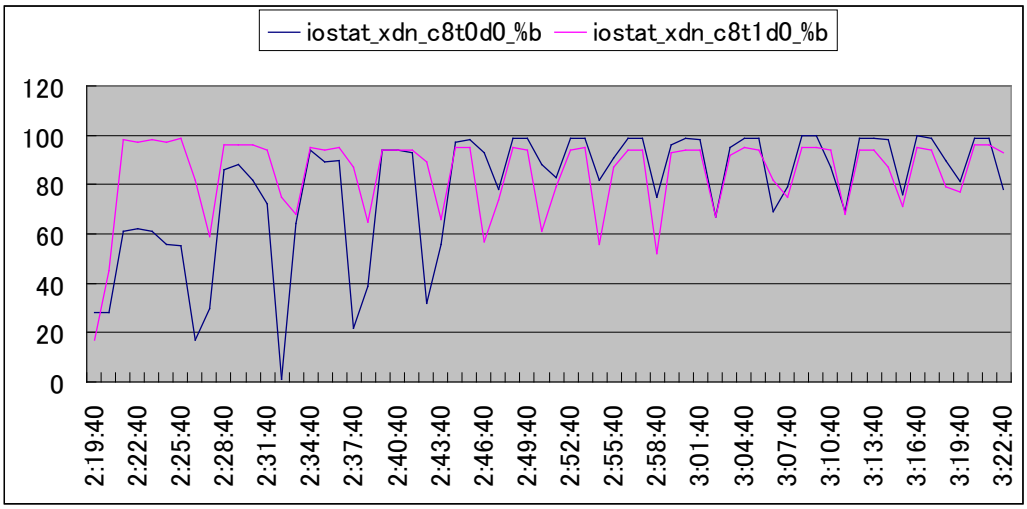
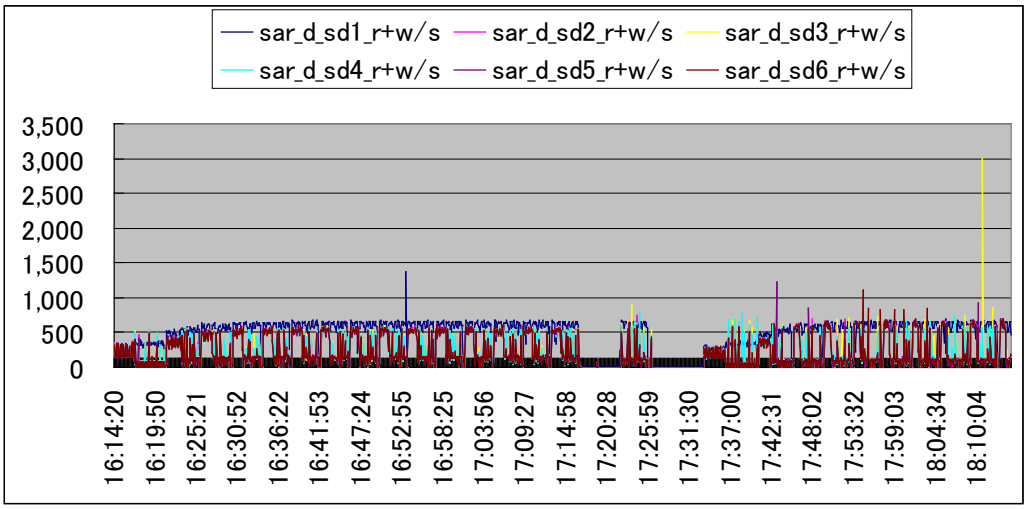


図 5-3-3-4. HW RAID0 UFS の秒あたりリードライト回数とディスクビジー

RAIDZ ではディスクアクセス回数が極端に少なくなっています。

rw/s



busy

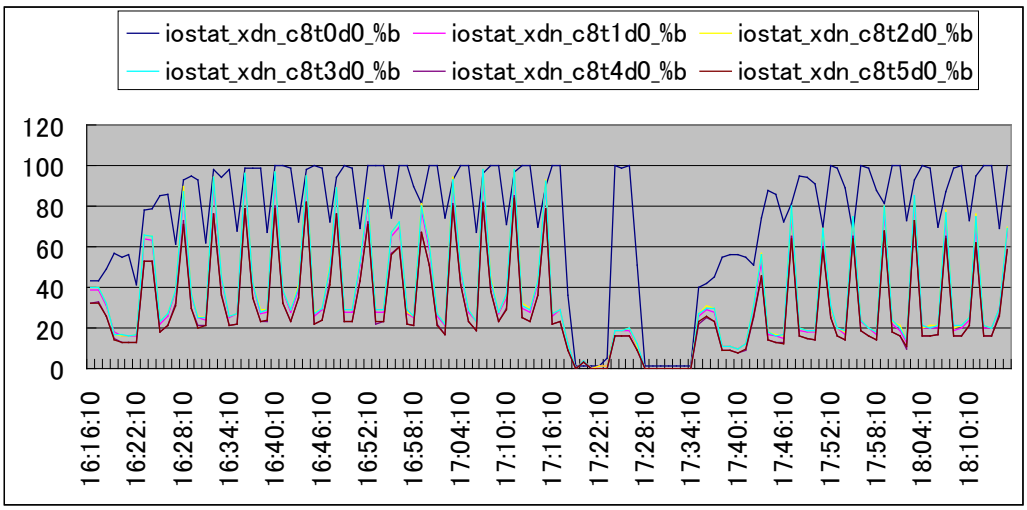
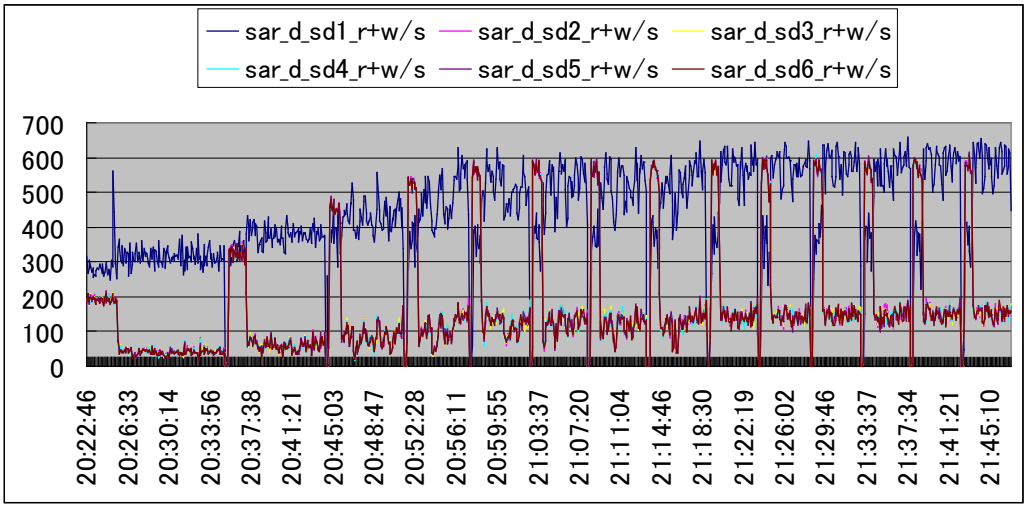


図 5-3-3-5. RAIDZ の秒あたりリードライト回数とディスクビジー

ネイティブの ZFS です。何と、ディスクビジーが 100%を切っています。ディスクアクセスが効率的になっているといえます。

ディスク装置数が増えると、より高いスループットを期待することができます。

rw/s



busy

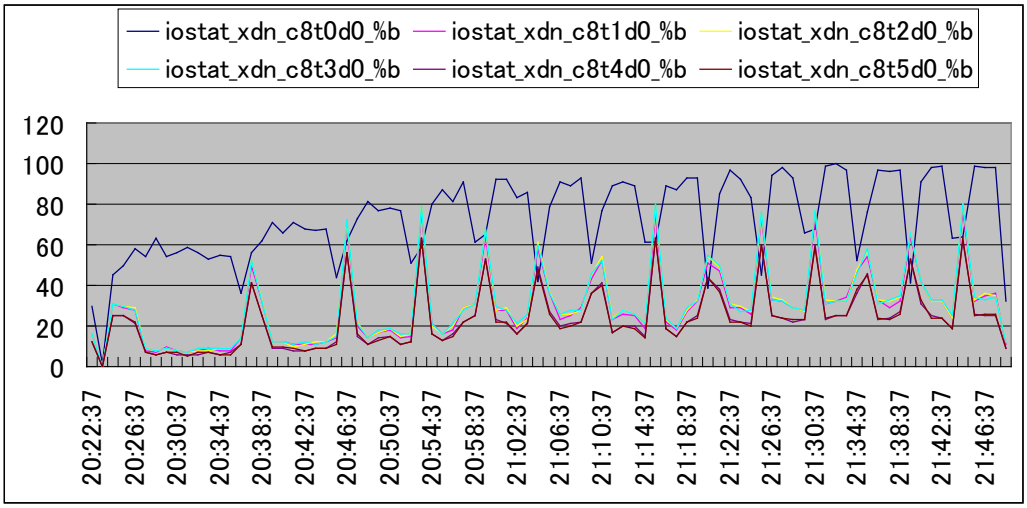
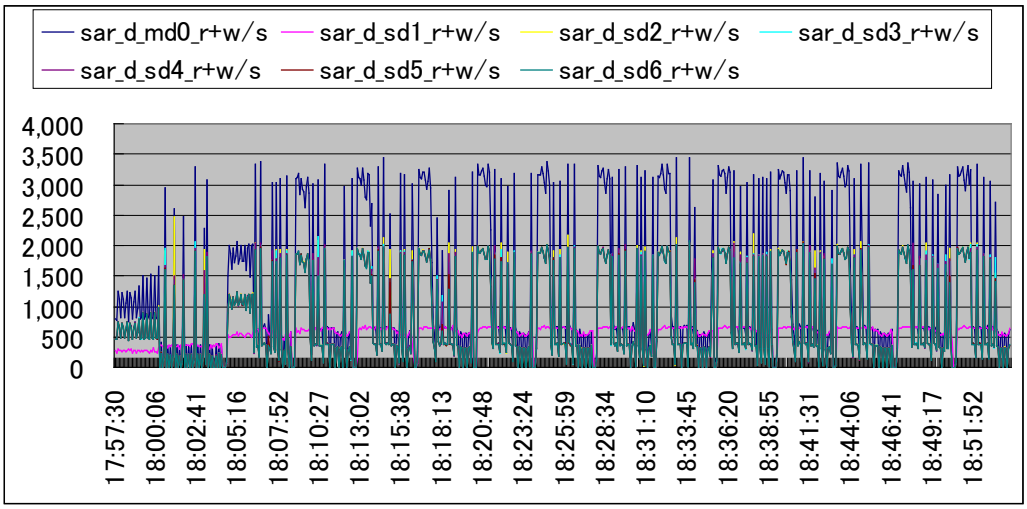


図 5-3-3-6. ZFS の秒あたりリードライト回数とディスクビジー

ディスクアクセス数が多いです。ディスクビジーも高い値になっています。

rw/s



busy

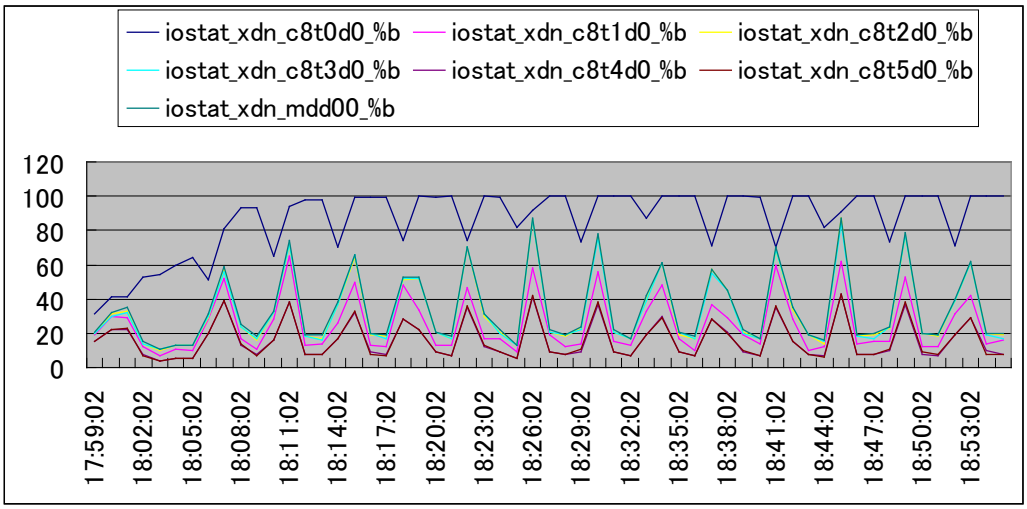


図 5-3-3-7. SVM RAID0 ZFS の秒あたりリードライト回数とディスクビジー

HW RAID0 ZFS は最も高いトータルスループットを引き出しましたが、ディスクアクセス回数が多く、ディスクビジーも高い値になっています。

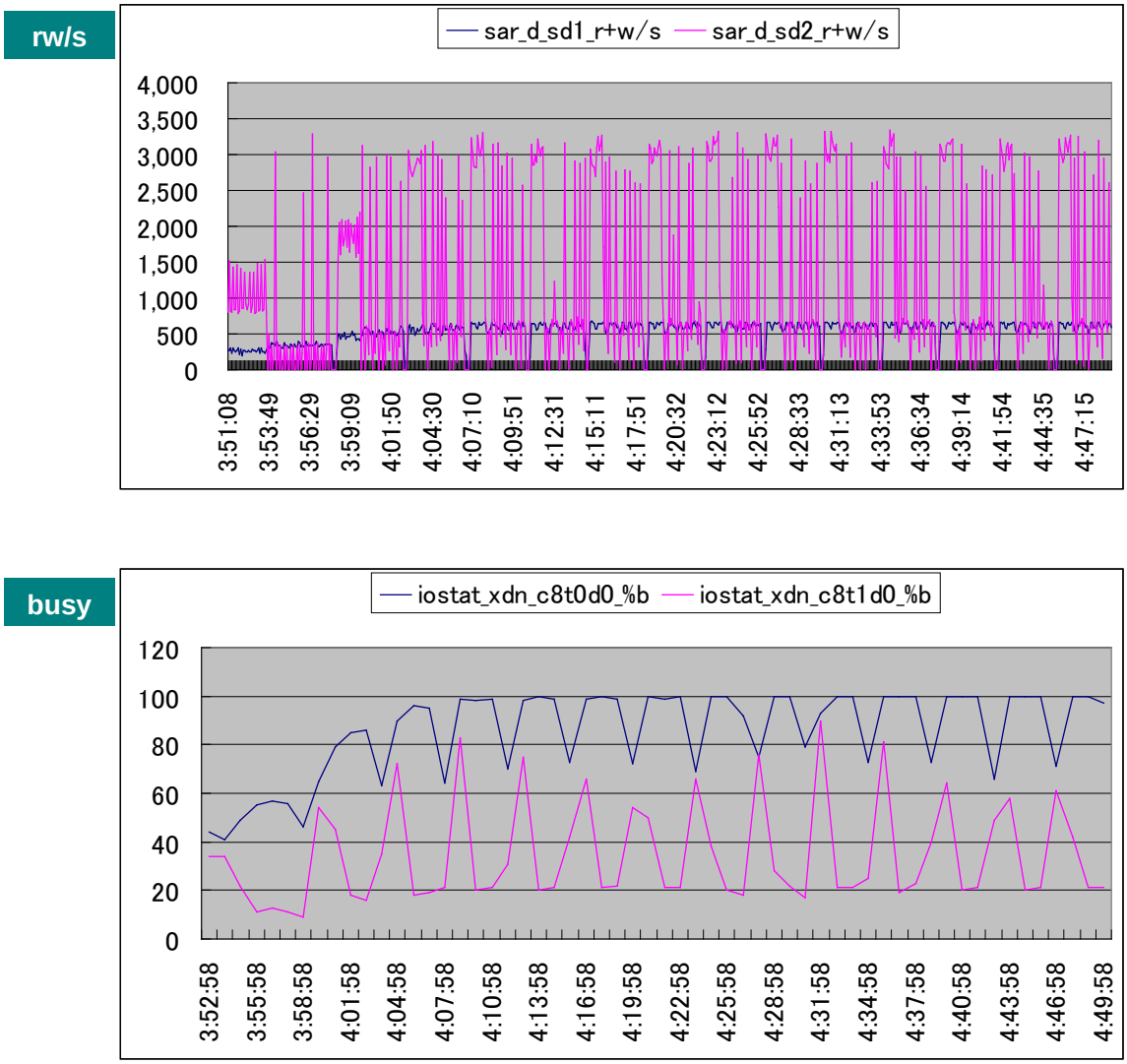


図 5-3-3-8. HW RAID0 ZFS の秒あたりリードライト回数とディスクビジー

ネイティブの ZFS は、ディスク装置を増やすことでより高い性能を引き出す余地があります。SVM や HW RAID は高いスループットではあるものの、高い CPU 負荷や、アクセス数の多さから、より高い性能は望めないと考えられます。

なお、ディスク装置の性能値は次のとおりです。

表 5-3-3-1. ディスク装置の諸元

ディスク名	インタフェース	外部転送速度 (MB/s)	回転数 (rpm)	外周から内周転送速度 (MB/s)
Savvio 10K.2	3Gb/s SAS	300	10,000	89-55
Savvio 15K	3Gb/s SAS	300	15,000	112-79

第 6 章 TIPS

今回の ZFS 評価で判明した情報をご紹介します。

6-1. 32 ビットカーネルの場合ディスク装置単体のサイズは 1TB 以下

32 ビットカーネルを使用している場合、ディスク装置単体のサイズは 1TB 以下でなくてはなりません。カーネルがディスク装置のブロックサイズを、2,147,483,648 個（2 ギガブロック）に制限しているためです。

次のメッセージが、/var/adm/messages ファイルに記録されます。

1.5TB ディスク装置の場合

Nov 27 16:02:00 opensolaris disk has **2930277164** blocks, which is too large for a 32-bit kernel

2TB ディスク装置の場合

Nov 27 16:02:00 opensolaris disk has **3907029168** blocks, which is too large for a 32-bit kernel

なお、32 ビットカーネルでも、ストレージプールのサイズは 1TB を越えて構成することはできます。次に合計 2TB(1.8TB)の構成例を示します。

```
root@opensolaris:~# isainfo -b ⏎
32
root@opensolaris:~# format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c7d0 <DEFAULT cyl 9722 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@1f,1/ide@0/cmdk@0,0
       1. c11t0d0 <Hitachi-VFG100R5DH-V5DO-232.89GB>
          /pci@0,0/pci1028,154@1d,7/storage@1/disk@0,0
       2. c12t0d0 <ST310003-33AS--931.51GB>
          /pci@0,0/pci1028,154@1d,7/storage@2/disk@0,0
       3. c13t0d0 <HDT72252-VDS41LT8CK-V44O-232.89GB>
          /pci@0,0/pci1028,154@1d,7/storage@3/disk@0,0
       4. c14t0d0 <HDT72252-5DLA380-A9BA-232.89GB>
          /pci@0,0/pci1028,154@1d,7/storage@4/disk@0,0
       5. c15t0d0 <Hitachi-VFA100R10V-V5DO-232.89GB>
          /pci@0,0/pci1028,154@1d,7/storage@5/disk@0,0
Specify disk (enter its number): ^C
root@opensolaris:~# zpool create -f xpool c11t0d0 c12t0d0 c13t0d0 c14t0d0 c15t0d0
⏎
root@opensolaris:~# zpool list ⏎
NAME    SIZE   USED  AVAIL    CAP  HEALTH  ALTROOT
rpool     74G  3.65G  70.4G     4%  ONLINE  -
xpool   1.82T  2.83M  1.82T     0%  ONLINE  -
root@opensolaris:~# zfs list ⏎
NAME                     USED  AVAIL  REFER  MOUNTPOINT
rpool                   4.01G  68.8G  77.5K  /rpool
rpool/ROOT              3.01G  68.8G    19K  legacy
rpool/ROOT/opensolaris  3.01G  68.8G  2.87G  /
rpool/dump               511M  68.8G   511M  -
rpool/export             956K  68.8G    21K  /export
```

```
rpool/export/home        935K   68.8G   21K    /export/home
rpool/export/home/taro   914K   68.8G   914K   /export/home/taro
rpool/swap               512M   69.2G   137M   -
xpool                    198K   1.79T   19K    /xpool
root@opensolaris:~#
```

次に 64 ビットカーネルでのストレージプール構成例です。次は合計 5.25TB(4.8TB)の構成例です。

```
root@act191:~# isainfo -b ⏎
64
root@act191:~# format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c7d0 <DEFAULT cyl 9722 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@e/ide@0/cmdk@0,0
       1. c9t0d0 <Hitachi-    JK1171YAGWH2KN-A20N-1.82TB>
          /pci@0,0/pci1028,1f4@b,1/storage@7/disk@0,0
       2. c10t0d0 <DEFAULT cyl 60798 alt 2 hd 255 sec 189> ←1.5TB【参考】
          /pci@0,0/pci1028,1f4@b,1/storage@8/disk@0,0
       3. c11t0d0 <ST310003-33AS--931.51GB>
          /pci@0,0/pci1028,1f4@b,1/storage@6/disk@0,0
       4. c12t0d0 <HDT72252-VDS41LT8CK-V44O-232.89GB>
          /pci@0,0/pci1028,1f4@b,1/storage@5/disk@0,0
       5. c13t0d0 <HDT72252-5DLA380-A9BA-232.89GB>
          /pci@0,0/pci1028,1f4@b,1/storage@3/disk@0,0
       6. c14t0d0 <Hitachi-     VFA100R10V-V5DO-232.89GB>
          /pci@0,0/pci1028,1f4@b,1/storage@4/disk@0,0
Specify disk (enter its number): ^C
root@act191:~# zpool create -f xpool c9t0d0 c10t0d0 c11t0d0 c12t0d0 c13t0d0
c14t0d0 ⏎
root@act191:~# zpool list ⏎
NAME SIZE USED AVAIL   CAPHEALTHALTROOT
rpool  74G   4.93G   69.1G 6%    ONLINE -
xpool  4.76T   272K    4.76T    0%    ONLINE -
root@act191:~# zfs list ⏎
NAME                     USED   AVAIL   REFER   MOUNTPOINT
rpool                    6.80G  66.0G   77.5K   /rpool
rpool/ROOT               2.86G  66.0G   19K     legacy
rpool/ROOT/opensolaris   2.86G  66.0G   2.85G   /
rpool/dump               1.97G  66.0G   1.97G   -
rpool/export             882K   66.0G   21K     /export
rpool/export/home        862K   66.0G   21K     /export/home
rpool/export/home/taro   840K   66.0G   840K    /export/home/taro
rpool/swap               1.97G  67.9G   101M    -
xpool                    77.5K  4.68T    19K    /xpool
root@act191:~#
```

【参考】ZFS でストレージプールを作成すると、ディスク上、EFI 形式(cyl 60798 alt 2 hd 255 sec 189 ではなく、Sector nnn の形式)になります。

```
       2. c10t0d0 <ST315003-41AS-0009-1.36TB>
          /pci@0,0/pci1028,1f4@b,1/storage@8/disk@0,0

partition> print ⏎
Current partition table (original):
Total disk sectors available: 2930260746 + 16384 (reserved sector
```

EFI 形式

```
Part      Tag    Flag     First Sector         Size         Last Sector
  0        usr    wm               256       1.36TB          2930260746
  1 unassigned    wm                 0          0               0
  2 unassigned    wm                 0          0               0
```

```
  3 unassigned    wm                0               0                0
  4 unassigned    wm                0               0                0
  5 unassigned    wm                0               0                0
  6 unassigned    wm                0               0                0
  8      reserved    wm       2930260747          8.00MB           2930277130

partition>
```

6-2. copies を大きくするとデータブロックがその数コピーされる

ZFS ではデータブロックの破損を検出すると自動的に修復されると言われています。これは copies プロパティによって複数のデータブロックが存在する時の機能と考えられます。ディスク分析によるとメタデータは copies プロパティの数が 1 でも、3 個作成されます。

データブロックを修復する目的で copies プロパティを 3 にする 場合は、ストレージプールの設計で、全データサイズ×3 倍の容量を考慮して構成する必要があります。

```
root@act191:~# format -e ⏎
Searching for disks...done


AVAILABLE DISK SELECTIONS:
       0. c7d0 <DEFAULT cyl 9722 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@e/ide@0/cmdk@0,0
       1. c11t0d0 <DEFAULT cyl 981 alt 2 hd 64 sec 32>
          /pci@0,0/pci1028,1f4@b,1/storage@6/disk@0,0
Specify disk (enter its number): 1 ⏎
selecting c11t0d0
[disk formatted]
    :
partition> print ⏎
Current partition table (original):
Total disk cylinders available: 981 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders        Size            Blocks
  0 unassigned    wm       0                0         (0/0/0)           0
  1 unassigned    wm       0                0         (0/0/0)           0
  2       backup    wu       0 - 980       981.00MB    (981/0/0) 2009088
  3 unassigned    wm       0                0         (0/0/0)           0
  4 unassigned    wm       0                0         (0/0/0)           0
  5 unassigned    wm       0                0         (0/0/0)           0
  6 unassigned    wm       0                0         (0/0/0)           0
  7 unassigned    wm       0                0         (0/0/0)           0
  8       boot    wu       0 -    0         1.00MB     (1/0/0)        2048
  9 unassigned    wm       0                0         (0/0/0)           0

partition>
    :
root@act191:~# zpool create tpool c11t0d0 ⏎
root@act191:~# zpool list ⏎
NAME    SIZE    USED   AVAIL   CAPHEALTHALTROOT
rpool   74G     4.93G  69.1G   6%    ONLINE -
tpool   968M    89.5K  968M    0%    ONLINE -
root@act191:~# zfs get all | grep copies ⏎
rpool                     copies   1    default
rpool/ROOT                copies   1    default
rpool/ROOT/opensolaris    copies   1    default
rpool/dump                copies   1    default
rpool/export              copies   1    default
```

```
rpool/export/home       copies  1    default
rpool/export/home/taro  copies  1    default
rpool/swap              copies  1    default
tpool     copies  1    default
root@act191:~# zfs create tpool/test ⏎
root@act191:~# cd /tpool/test ⏎
root@act191:/tpool/test# dd if=/dev/dsk/c7d0s0 of=testfile count=1828100 ⏎
root@act191:/tpool/test# ls -l ⏎
total 907425
-rw-r--r-- 1 root root 935987200 2009-11-27 23:22 testfile
root@act191:/tpool/test# zfs list
NAME                    USED   AVAIL   REFER   MOUNTPOINT
rpool                   6.80G  66.0G   77.5K   /rpool
rpool/ROOT              2.86G  66.0G   19K     legacy
rpool/ROOT/opensolaris  2.86G  66.0G   2.85G   /
rpool/dump              1.97G  66.0G   1.97G    -
rpool/export            882K   66.0G   21K     /export
rpool/export/home       862K   66.0G   21K     /export/home
rpool/export/home/taro   840K  66.0G   840K    /export/home/taro
rpool/swap              1.97G  67.9G   101M    -
tpool                   894M   42.3M   21K     /tpool
tpool/test              893M   42.3M   893M    /tpool/test
root@act191:/tpool/test# zfs set copies=3 tpool/test ⏎
root@act191:/tpool/test# zfs get all | grep copies | grep test ⏎
tpool/test   copies  3    local
root@act191:/tpool/test# zfs list ⏎
NAME                    USED   AVAIL   REFER MOUNTPOINT
rpool                   6.80G  66.0G   77.5K   /rpool
rpool/ROOT              2.86G  66.0G   19K     legacy
rpool/ROOT/opensolaris  2.86G  66.0G   2.85G   /
rpool/dump              1.97G  66.0G   1.97G   -
rpool/export            882K   66.0G   21K     /export
rpool/export/home       862K   66.0G   21K     /export/home
rpool/export/home/taro   840K  66.0G   840K    /export/home/taro
rpool/swap              1.97G  67.9G   101M    -
tpool                   594M   342M    21K     /tpool
tpool/test              613M   323M    613M    /tpool/test
root@act191:/tpool/test# ls -l | grep testfile ⏎
-rw-r--r-- 1 root root 257032704 2009-11-27 23:29 testfile
root@act191:/tpool/test# zfs list | grep test ⏎
tpool/test              736M    200M    736M    /tpool/test
root@act191:/tpool/test# ls -l | grep testfile ⏎
-rw-r--r-- 1 root root 308412928 2009-11-27 23:30 testfile
root@act191:/tpool/test# zfs list | grep test ⏎
tpool/test              903M   32.7M    903M    /tpool/test
root@act191:/tpool/test# ls -l | grep testfile ⏎
-rw-r--r-- 1 root root 325845504 2009-11-27 23:30 testfile
root@act191:/tpool/test# zfs list | grep test ⏎
tpool/test              933M   2.69M    933M    /tpool/test
root@act191:/tpool/test# ls -l | grep testfile ⏎
-rw-r--r-- 1 root root 327024640 2009-11-27 23:31 testfile
root@act191:/tpool/test# zfs list | grep test ⏎
tpool/test              936M       0    936M    /tpool/test
root@act191:/tpool/test#

【別の端末より実行】
root@act191:/tpool/test# dd if=/dev/dsk/c7d0s0 of=testfile count=1828100 ⏎
dd: writing to `testfile': No space left on device
638721+0 records in
638720+0 records out
```

327024640 bytes (327 MB) copied, 143.514 s, 2.3 MB/s
root@act191:/tpool/test#

6-3. ZFS の機能確認は USB デバイスや Sun VirtualBox を利用

ZFS の機能確認は USB や Sun VirtualBox を利用すると存分にテストすることができます。USB デバイスは大変便利です。本書では、付録に Sun VirtualBox で OpenSolaris 2009.06 をインストールして ZFS を使用する例を解説しています。参考にしてください。

なお、性能評価は実機を使用すべきと考えます。

6-4. システムディスクは HW RAID+データディスクは ZFS が強力

ZFS のデータ保全は万全です。一方、システムディスク(rpool)は、ミラー設定できますが、設定が煩雑で手間がかかります。また、BE の動作はあまり芳しいとは言えません。このため、

システムディスクはハードウェア RAID + データディスクは ZFS

が強力な組み合わせと考えます。ZFS のホットスペアを構成しておくと万が一に備え、安心です。

6-5. 高回転ディスク(または SSD)+たくさんのディスクを構成する

ログやキャッシュデバイスを NVRAM や SSD に構成すると、より良い性能を引き出すことができます。ZFS の強みはストライプ幅を可変にして、多くのディスク装置に分散する点です。このため、高性能を引き出すには、高回転ディスク(または SSD)で構成し、加えて、チャネルパスを分けると最も良い性能を発揮することができると考えます。

6-6. プールをエクスポート･インポートする場合の TIPS

ZFS で export したストレージプールを他のシステムで import する場合の TIPS をご紹介します。

(1) SPARC から x86、x86 から SPARC へストレージプールの export/import が自由にできます。ディスク内の記録形式(Big Endian、Little Endian)はフラグのみが書き換えられ、データは元の create された時点の Endian で操作されるようです。また、OpenSolaris、Solaris 10 で互換性があります。x86 で、32 ビット、64 ビットも互換性があります。但し、前述の 32 ビットカーネル、ディスク装置のサイズ(1TB より大きいか小さいか)と、後述の ZFS バージョンにご注意ください。

(2) zpool upgrade するとストレージプールのバージョンが新しくなります。
一旦 zpool upgrade でストレージプールのバージョンが新しくなると、低いバージョンには戻ることはできませんので注意してください。次の警告メッセージが表示されます。

OpenSolaris 2009 06 の場合

```
root@act191:~# zpool import tpool ⏎
root@act191:~# zpool status ⏎
  pool: rpool
 state: ONLINE
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
```

```
            c7d0s0      ONLINE       0     0     0

errors: No known data errors

  pool: tpool
 state: ONLINE
status: The pool is formatted using an older on-disk format.  The pool can
        still be used, but some features are unavailable.
action: Upgrade the pool using 'zpool upgrade'.  Once this is done, the
        pool will no longer be accessible on older software versions.
 scrub: none requested
config:

        NAME          STATE     READ WRITE CKSUM
        tpool         ONLINE       0     0     0
          c11t0d0     ONLINE       0     0     0

errors: No known data errors
root@act191:~#
```

Solaris 10 10/09 の場合 日本語で表示されるので分かりやすいです(LANG 環境変数?)…

```
root@act062 # zpool import tpool ⏎
root@act062 # zpool status ⏎
  プール: rpool
 状態: ONLINE
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          c1t1d0s0    ONLINE       0     0     0

エラー: 既知のデータエラーはありません

  プール: tpool
 状態: ONLINE
状態: このプールはディスク上の古い形式を使用してフォーマットされています。
        プールは引き続き使用できますが、一部の機能は利用できません。
アクション: 'zpool upgrade' を使用してプールをアップグレードします。この操作を実行すると、
        以前のバージョンのソフトウェアからプールにアクセスできなくなります。
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE     READ WRITE CKSUM
        tpool         ONLINE       0     0     0
          c5t0d0      ONLINE       0     0     0

エラー: 既知のデータエラーはありません
root@act062 #
```

※アップグレードすると次のようにメッセージが表示され、アップグレードされます。

```
root@act062 # zpool upgrade tpool ⏎
このシステムでは現在、バージョン 15 の ZFS プールが動作しています。

'tpool' をバージョン 10 からバージョン 15 にアップグレードすることに成功しました

root@act062 #
```

(3) zpool upgrade しなくてもストレージは使用可能です。ただし、新しい機能を使用する時に制限されることがあります(上記の警告メッセージをご覧ください)。

(4) 低いバージョンのプールで通したい場合は、プール作成時に指定することでバージョンを固定することができます。次はバージョン 10 でストレージプールを作成する例です。

```
# zpool create -o version=10 tpool c1t0d0 c1t2d0 ⏎
```

(5) 新しいバージョンのストレージプールは古いバージョンの ZFS でインポートできません。次のメッセージが表示されます。

OpenSolaris 2009.06 の場合

```
root@act191:~# zpool import tpool ⏎
cannot import 'tpool': pool is formatted using a newer ZFS version
root@act191:~#
```

Solaris 10 10/09 の場合

```
root@act061 # zpool import tpool ⏎
'tpool' をインポートできません: プールはより新しいバージョンの ZFS を使用してフォーマットされています
root@act061 #
```

(6) 他でエクスポートされたディスク装置に対してストレージプールを create しようとすると、次のメッセージが表示されます。

Solaris 10 10/09 の場合

```
root@act061 # zpool create tpool c5t0d0 ⏎
仮想デバイスの指定が無効です
次のエラーを無効にするには、'-f' を使用してください:
/dev/dsk/c5t0d0s0 は、エクスポートされたまたはアクティブな可能性のある ZFS プール tpool に含まれています。zpool(1M) を参照してください。
root@act061 #
```

新たなストレージプールをそのディスク装置に作成する場合は、“-f”オプションを指定して createします。元のデータは消えます。

```
root@act061 # zpool create -f tpool c5t0d0 ⏎
root@act061 #
```

(7) インポートはそれなりに時間がかかる?

時間測定は未実施ですが、ストレージプール内のデータやファイル量が多いとそれなりに時間がかかると思われます。これは、テスト中にディスクの DMU(DVA)アドレスが変化することが判明したためです(理由は不明)。エクスポートはファイルシステム SYNC してアンマウントするだけなので、あまり時間はかからないと思います。

6-7. USB デバイスにストレージプールを作成できないことがある

Solaris 10 で USB を使用しているとき、「仮想デバイスの指定が無効です」のエラーが発生しました。

```
root@act061 # format -e ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c1t1d0 <SUN72G cyl 14087 alt 2 hd 24 sec 424>
          /pci@8,600000/SUNW,qlc@4/fp@0,0/ssd@w21000004cfa1030a,0
       1. c1t2d0 <SEAGATE-ST373405FSUN72G-0438-68.37GB>
          /pci@8,600000/SUNW,qlc@4/fp@0,0/ssd@w21000004cf9b6278,0
       2. c4t0d0 <ST310003-33AS--931.51GB>
          /pci@8,700000/usb@5,3/storage@3/disk@0,0
Specify disk (enter its number): ^C
root@act061 # zpool create -f tpool c4t0d0 ⏎
仮想デバイスの指定が無効です
次のエラーは手動で修復する必要があります:
'/dev/dsk/c4t0d0' のドライブにメディアが入っていません
root@act061 #
```

※何をやっても効果が無かったため、volfs を停止したところストレージプールを作成することができました。

```
root@act061 # svcs -a | grep vol ⏎
online             0:57:58 svc:/system/filesystem/volfs:default
root@act061 # svcadm disable svc:/system/filesystem/volfs:default ⏎
root@act061 # svcs -a | grep vol ⏎
disabled           1:18:25 svc:/system/filesystem/volfs:default
root@act061 # zpool create -f tpool c4t0d0 ⏎
root@act061 # zpool list ⏎
NAME         SIZE     USED     AVAIL     CAP      HEALTH   ALTROOT
rpool        68G      18.5G     49.5G     27%      ONLINE    -
tpool        928G     111K      928G       0%      ONLINE    -
root@act061 # zpool status ⏎
  プール: rpool
 状態: ONLINE
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE       READ WRITE CKSUM
        rpool         ONLINE         0     0     0
          c1t1d0s0    ONLINE         0     0     0

エラー: 既知のデータエラーはありません

  プール: tpool
 状態: ONLINE
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE       READ WRITE CKSUM
        tpool         ONLINE         0     0     0
          c4t0d0      ONLINE         0     0     0

エラー: 既知のデータエラーはありません
root@act061 #
```

6-8. ディスク装置の電源を入れ忘れてブートすると…

ディスク装置の電源を OFF したままシステムをブートすると次のようなエラーになります。

```
root@act061 # zpool status ⏎
  プール: rpool
 状態: ONLINE
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          c1t1d0s0    ONLINE       0     0     0

エラー: 既知のデータエラーはありません

  プール: tpool
 状態: UNAVAIL
状態: 1 つまたは複数のデバイスを開くことができませんでした。
        複製が不足しているため、プールは動作を継続できません。
アクション:見つからなかったデバイスを接続し、'zpool online'を使用してオンラインにしてください。
次のサイトを参照してください: http://www.sun.com/msg/ZFS-8000-3C
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE     READ WRITE CKSUM
        tpool         UNAVAIL      0     0     0  複製が不足しています
          c5t0d0      UNAVAIL      0     0     0  オープンに失敗しました。
root@act061 #
```

ディスク装置の電源を入れると自動的に認識され、マウントされます。

```
root@act061 # zpool status ⏎
  プール: rpool
 状態: ONLINE
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          c1t1d0s0    ONLINE       0     0     0

エラー: 既知のデータエラーはありません

  プール: tpool
 状態: ONLINE
 スクラブ: 何も要求されませんでした
構成:

        NAME          STATE     READ WRITE CKSUM
        tpool         ONLINE       0     0     0
          c5t0d0      ONLINE       0     0     0

エラー: 既知のデータエラーはありません
root@act061 #
```

6-9. ディスク装置の増減とリカバリ手順のまとめ

6-9-1. ディスク装置の増減

表 6-9-1-1. ディスク装置の増減

RAID レベル	増設	交換	減設
ZFS	add	replace	remove
mirror	attach または add	replace	detach
RAIDZ	add	replace	×
RAIDZ2	add	replace	×

6-9-2. ホットスペアを構成している場合のリカバリ手順

表 6-9-2-1. ホットスペア構成時のリカバリ手順

RAID レベル	故障時の状態およびメッセージ	使用コマンド	復旧手順
ZFS	Pool State : UNAVAIL Pool Msg : insufficient replicas Dev State : UNAVAIL Dev Msg : エラーメッセージ Spare : “-f”指定で作成できますが、故障時に消えます	×	×
mirror	Pool State : DEGRADED Mirror State : DEGRADED Dev State : UNAVAIL Dev Msg : エラーメッセージ Spare State : INUSE Spare Msg : currently in use	detach ↓ clear	自動でスペアが“INUSE”になり、 “○○K resilvered” が表示されます。 1. “detach”コマンドで壊れたデバイスを切り離します。 2. “clear”コマンドで“too many errors”が出ているデバイスのエラーを消します。
RAIDZ	Pool State : DEGRADED RAIDZ State : DEGRADED Dev State : UNAVAIL Dev Msg : エラーメッセージ Spare State : INUSE Spare Msg : currently in use	detach	自動でスペアが“INUSE”になり、 “○○K resilvered” が表示されます。 1. “detach”コマンドで壊れたデバイスを切り離します。
RAIDZ2	Pool State : DEGRADED RAIDZ2 State : DEGRADED Dev State : UNAVAIL Dev Msg : エラーメッセージ Spare State : INUSE Spare Msg : currently in use	detach	自動でスペアが“INUSE”になり、 “○○K resilvered” が表示されます。 1. “detach”コマンドで壊れたデバイスを切り離します。

6-9-3. ホットスペアを構成していない場合のリカバリ手順

表 6-9-3-1. ホットスペア無しの場合のリカバリ手順

RAID レベル	故障時の状態およびメッセージ	使用コマンド	復旧手順
ZFS	Pool State : UNAVAIL Pool Msg : insufficient replicas Dev State : UNAVAIL Dev Msg : エラーメッセージ	×	×
mirror	Pool State : DEGRADED Mirror State : DEGRADED Dev State : UNAVAIL Dev Msg : エラーメッセージ	detach ↓ attach または add spare ↓ replace ↓ detach ↓ clear	**スペアを追加しない場合** 1. “detach”コマンドで壊れたデバイスを切り離します。 2. “attach”コマンドで既存のデバイスと新しいデバイスをミラー化します。 自動で resilvered されます。 新しいデバイスに “○○K resilvered” が表示されます。 **スペアを追加した場合** 1. “add”コマンドでスペアを追加します。 2. “replace”コマンドで故障したデバイスと追加したスペアを入れ替え作業します。ここでスペアが“INUSE”になり、“○○K resilvered”が表示されます。 3. 故障したデバイスを“detach”コマンドで切り離します。 4. “clear”コマンドで“too many errors”が出ているデバイスのエラーを消します。
RAIDZ	Pool State : DEGRADED RAIDZ State : DEGRADED Dev State : UNAVAIL Dev Msg : エラーメッセージ	replace または add spare ↓ replace ↓ detach	**スペアを追加しない場合** 1. “replace”コマンドで故障しているデバイスと新しいデバイスを入れ替えます。 自動で resilvered されます。 新しいデバイスに “○○K resilvered” が表示されます。 **スペアを追加した場合** 1. スペアを“add”コマンドで追加します。 2. “replace”コマンドで故障したデバイスと追加したスペアを入れ替え作業します。ここでスペアが“INUSE”になり、“○○K resilvered”が表示されます。 3. 故障したデバイスを“detach”コマ

			ンドで切り離します。
RAIDZ2	Pool State : DEGRADED RAIDZ2 State : DEGRADED Dev State : UNAVAIL Dev Msg : エラーメッセージ	replace または add spare ↓ replace ↓ detach	**スペアを追加しない場合** 1．“replace”コマンドで故障しているデバイスと新しいデバイスを入れ替えます。 自動で resilvered されます。 新しいデバイスに “○○K resilvered” が表示されます。 **スペアを追加した場合** 1．スペアを“add”コマンドで追加します。 2．“replace”コマンドで故障したデバイスと追加したスペアを入れ替え作業します。ここでスペアが“INUSE”になり、“○○K resilvered”が表示されます。 3．故障したデバイスを“detach”コマンドで切り離します。

6-10. Windows における GPT 保護の解除

直接 ZFS とは関係しませんが、ZFS で使用したディスクを Windows で使用した場合の TIPS を紹介します。

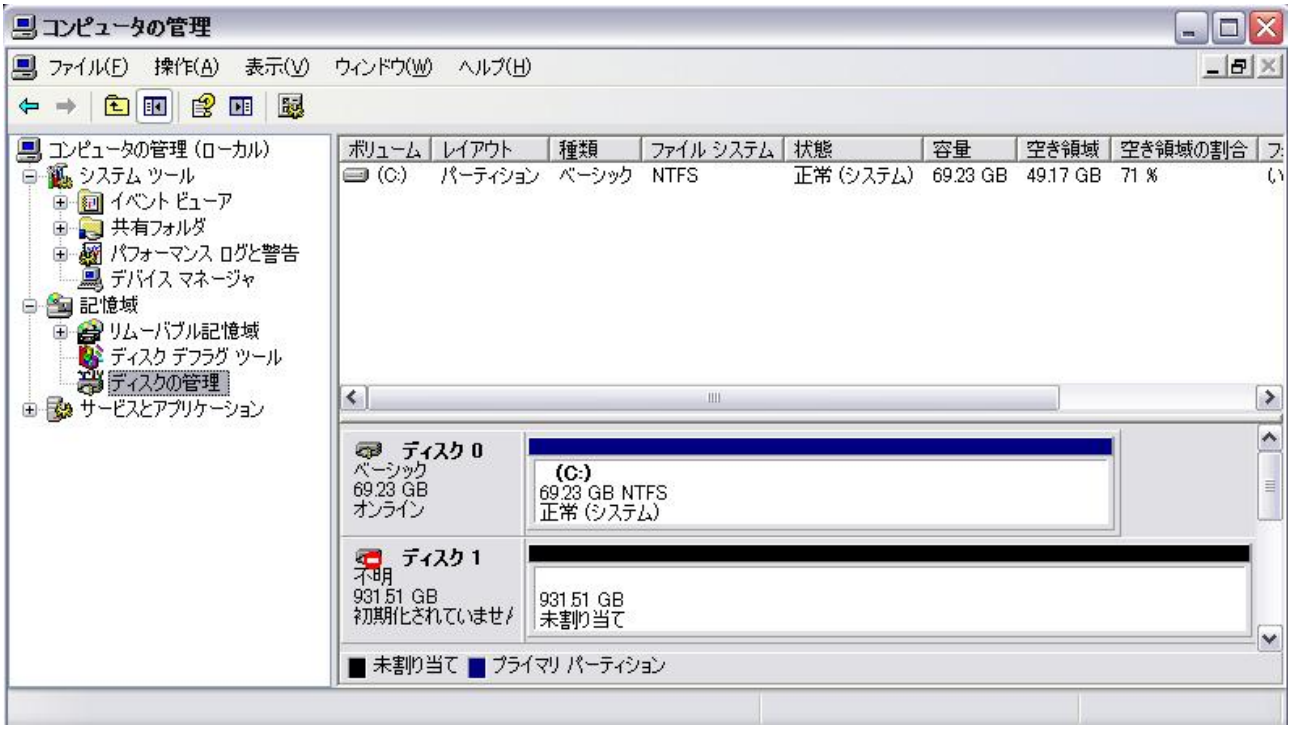
ディスク装置を USB で ZFS に使用した後、そのディスク装置を Windows で使用する場合、「コンピュータの管理」→「ディスクの管理」で次のように、「GPT 保護パーティション」と表示されます。GPT(GUID Partition Table)で保護されているという意味で、EFI 形式でフォーマットされているためです。

この場合、ディスクを選択して右クリックしても、サブメニューがグレー表示され、機能を選択できません。このため、ボリュームのパーティションをフォーマットするような操作が一切できません。

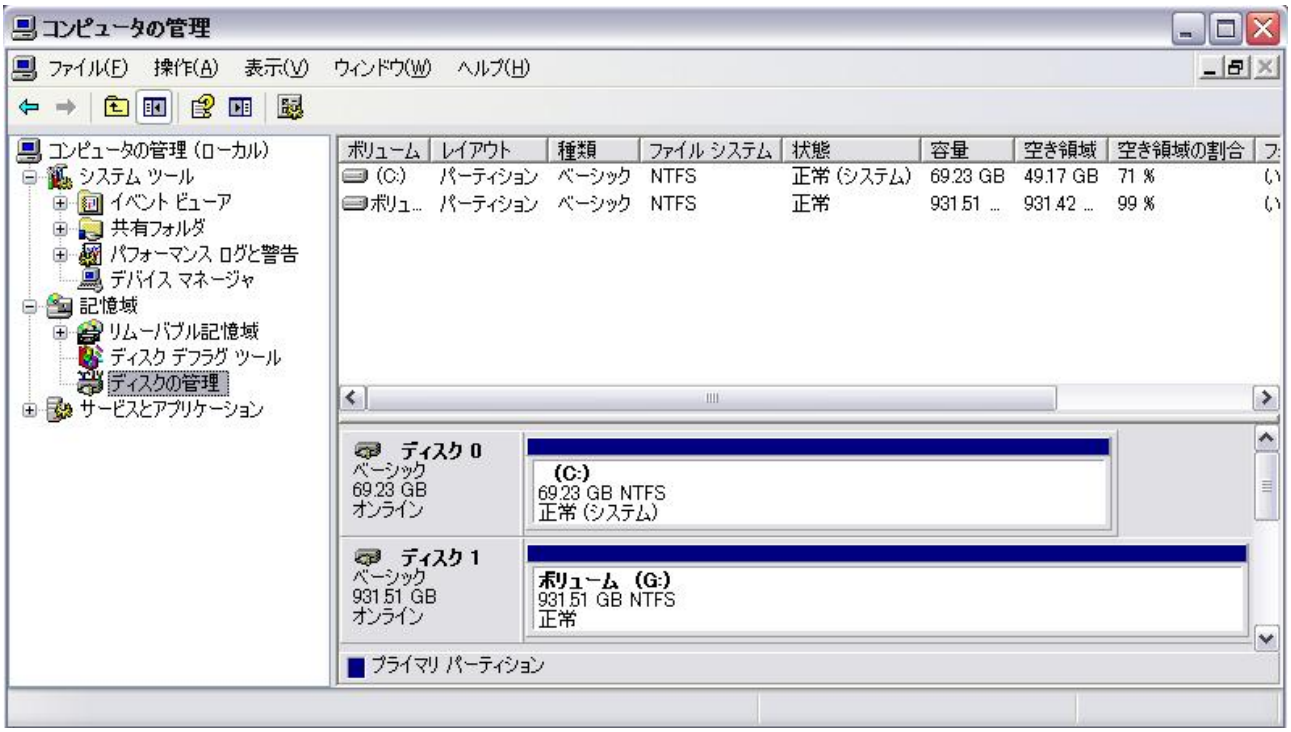
この「GPT 保護パーティション」を解除するには、次の手順でディスクをクリアして使用して下さい。

```
C:¥Documents and Settings¥taro>diskpart ⏎
Microsoft DiskPart version 5.1.3565
Copyright (C) 1999-2003 Microsoft Corporation.
コンピュータ: ACT102
DISKPART> list disk ⏎
  Disk ###  Status         Size     Free     Dyn  Gpt
  --------  -------------  -------  -------  ---  ---
  Disk 0    オンライン          69 GB      0 B
  Disk 1    オンライン         932 GB      0 B
DISKPART> select disk 1 ⏎
ディスク 1 が現在選択されているディスクです。
DISKPART> clean ⏎
DiskPart はディスクを正常にクリーンな状態にしました。
DISKPART> exit ⏎
DiskPart を終了しています...
C:¥Documents and Settings¥taro>
```

これによって、ディスクの操作が可能になります。



フォーマットすると次のように使用可能になります。



【備考】OpenSolaris や Solaris では、“**format -e**”コマンドによって、EFI か SMI ディスクラベルを選択して、ディスク装置をフ ォ ー マットすることができます。

関連用語

ABC 順用語

ACL
Access Control List : アクセス制御リスト

ACLとは、オブジェクト（受動体）に付属する許可属性のリスト。コンピュータセキュリティにおけるアクセス制御を実現するために、誰にどのリソース（受動体）に対するどの操作を許可するかを列挙したもの。例えば、ファイル X についてのアクセス制御リストに要素(Alice,delete)があれば、Alice はファイル X を削除することができます。

ARC
Adaptive Replacement Cache

ZFS のメモリキャッシュです。1 次キャッシュとして用いられるものです。

beadm ユーティリティー

beadm ユーティリティーは、OpenSolaris ソフトウェアでブート環境を管理するためのユーザーインタフェースです。“beadm”コマンドは、“luupgrade”や“lucreate”などの Solaris Live Upgrade コマンドの置き換えです。

Big Endian
ビッグエンディアン

SPARC のようなコンピュータ内部のバイト表記を表します。1 ワード 4 バイトのバイトシーケンスを示しています。例えば“1234”は“0x31323334”と表現されます。Intel 系のコンピュータは Little Endian と呼ばれ、バイトシーケンスはこの逆“0x34333231”になります。

CIFS
Common Internet File System : コモンインターネットファイルシステム

Microsoft 独自の SMB を正式にドキュメント化して仕様を公開することで、Windows 以外の OS やインターネット上を介してファイル共有サービス等を利用できるように拡張したもの。

EFI ディスクラベル
Extensible Firmware Interface : 拡張ファームウェアインタフェース

通常、システムディスク装置は、シリンダー(ヘッド、セクター)の単位でパーティショニングされ、ブートやルートファイルシステムが作成されます(SMI ディスクラベル)。ZFS で、ディスク装置をまるごと指定(例えば c9t1d0 のように)してデータディスクを作成すると、シリンダーやヘッド単位のパーティショニングは行われず、全部のセクターがまとまって見えます。このディスクラベル形式をいいます。

/etc/netboot ディレクトリ

WAN ブートインストールに必要なクライアント構成情報とセキュリティーデータが格納されている、WAN ブートサーバー上のディレクトリ。

/etc ディレクトリ

重要なシステム構成ファイルや保守コマンドが収められているディレクトリ。

/export ファイルシステム

OS サーバー上のファイルシステムで、ネットワーク上の他のシステムと共有されます。たとえば、“/export”ファイルシステムには、ディスクレスクライアント用のルート(/)ファイルシステムとスワップ空間、それにネットワーク上のユーザーのホームディレクトリを収めることができます。ディスクレスクライアントは、起動と実行の際に OS サーバー上の“/export”ファイルシステムに依存します。

EXT
Extended File System : エクステンデッドファイルシステム

Linux オペレーティングシステム向けに作成されたファイルシステム。

fdisk パーティション

x86 ベースのシステム上にある特定のオペレーティングシステム専用のディスクドライブの論理パーティション。Solaris ソフトウェアをインストールするには、x86 システム上に 1 つ以上の Solaris fdisk パーティションを設定する必要があります。x86 ベースのシステムでは、1 台のディスクに最大 4 つの fdisk パーティションを作成できます。これらのパーティションは、個別のオペレーティングシステムをインストールして使用できます。各オペレーティングシステムは、独自の fdisk パーティション上に存在しなければなりません。個々のシステムの Solaris fdisk パーティションの数は、1 台のディスクにつき 1 つに限られます。

GRUB

GNU GRand Unified Bootloader : グランドユニフィードブートローダー

x86 のみ：GRUB は、簡単なメニューインタフェースを備えたオープンソースのブートローダーです。メニューには、システムにインストールされているオペレーティングシステムのリストが表示されます。GRUB を使用すると、Solaris OS、Linux、または Microsoft Windows などの様々なオペレーティングシステムを、簡単にブートすることができます。

GRUB 編集メニュー

x86 のみ：GRUB メインメニューのサブメニューであるブートメニュー。このメニューには、GRUBコマンドが表示されます。これらのコマンドを編集して、ブート動作を変更できます。

GRUB メインメニュー

x86 のみ：システムにインストールされているオペレーティングシステムがリストされたブートメニュー。このメニューから、BIOS または fdisk パーティションの設定を変更することなく、簡単にオペレーティングシステムをブートできます。

i ノード(アイノード)

Linux などの UNIX 系 OS では、パーティションが Ext2 ファイルシステムや Ext3 ファイルシステムなどでフォーマットされると、ファイルシステムの管理領域とデータ領域などがパーティション内に作成されます。データ領域にはファイルの実データなどが記録されます。管理領域には、ファイルの実データが格納されている位置やファイルサイズ、変更時刻などの管理情報が記録されます。この管理情報を“i-node(アイノード)”といいます。

iSCSI

Internet Small Computer System Interface : アイスカジー

IETF によって RFC(request for comment)として公表された公式な規格への提案であり、SCSI プロトコルを TCP/IP ネットワーク上で使用する規格。iSCSI は SCSI-3 で規定されるフレームワークではトランスポート層に相当します。トランスポート層には他に並列（パラレル）SCSI やファイバーチャネルがあります。

ISO イメージ

1 つのファイル内でオペレーティングシステム全体を構成するソフトウェアのコレクション。ISO イメージは、インターネットから入手できます。ISO イメージには、ブート可能な CD または DVD の作成に適したファイルシステムが含まれます。ISO イメージはブート可能であり、インストールやその他の目的に使用できます。

L2ARC

Level 2 ARC

ZFS の ARC 読み取り用拡張キャッシュです。SSD や NVRAM を使用することで、読み取り性能を向上させることができます。

Little Endian

リトルエンディアン

Intel 系のコンピュータで、コンピュータ内部の 1 ワード内のバイトシーケンスを示します。例えば“1234”は、“0x34333231”となります。Big Endian の逆です。

LV

logical volume : ロジカルボリューム

物理的に複数のハードディスクまたはパーティションをグ

ループ化して、仮想的に 1 つのボリュームとしたもの。

LVM
logical volume manager :
論理ボリュームマネージャ

LVM とは、UNIX 系の大規模ストレージ/ディスクマネージメント機能の総称。商用 UNIX ベンダがそれぞれの UNIX において LVM を提供しています。

lzjb

エルゼットジェービー

Sun Microsystems の ZFS 開発者 Jeff Bonwick 氏の開発した圧縮・復元アルゴリズムです。

MD5
Message Digest Algorithm 5 :
エムディーファイブ

デジタル署名などのメッセージ認証に使用する繰り返し暗号化のハッシュ関数。

NFS
Network File System :
ネットワークファイルシステム

ローカルに接続されたストレージをネットワークを介してリモートの計算機に提供する分散ファイルシステムとそのプロトコル。マウントされた NFS ボリュームは、ネットワーク上にあることを意識せずローカルと同じように利用できます。

NTFS
NT File System : NT ファイルシステム

Windows NT 系 OS の標準ファイルシステム。

RAID
Redundant Arrays of Inexpensive (もしくは Independent)Disks : レイド

複数台のハードディスクを組み合わせることで仮想的な 1 台のハードディスクとして運用する技術。ディスクアレイの代表的な実装形態で、主に信頼性の向上をねらって用いられるものです。

RBAC セキュリティモデル
Role Based Access Control
セキュリティモデル

管理者が非 root ユーザー、または UNIX グループに対してロールを割り当て、それぞれのロールに操作(オペレーション)とオブジェクト(権限)を与え、実行を制御するセキュリティモデルです。

root

複数の項目から成る階層構造の最上位。ルートは、他のすべての項目を子孫として持つ唯一の項目です。

SHA-1
Secure Hashing Algorithm :
セキュアハッシングアルゴリズム

このアルゴリズムは、長さが 2^{64} 未満の入力に対して演算を行い、メッセージダイジェストを生成します。

SMI ラベル

Sun Microsystems Incorporate や、**Storage Management Initiative** との説があります。現在のところ定かではありません。VTOC を持つディスク装置のラベル形式で、シリンダー、ヘッド単位でパーティションが作成されます。

Solaris ゾーン

ソフトウェアによるパーティション分割技術。オペレーティングシステムのサービスを仮想化し、隔離された安全なアプリケーション実行環境を提供します。非大域ゾーンを作成すると、そのアプリケーション実行環境で実行されるプロセスは、他のゾーンと隔離されます。このように隔離することで、あるゾーンで実行中のプロセスが他のゾーンで実行中のプロセスを監視したり操作したりすることを防ぐことができます。

SVM
Solaris Volume Manager :
ソラリスボリュームマネージャ

SVM は Solaris 9 から OS に統合されていますが、も

ともとビジネス基幹系(高信頼性/高可用性)の機能に弱い SUN が独自に追加した論理ボリュームマネージャ(LVM)機能。

uberblock
ウーバブロック

ソースコードでは、“oo-ba-block”と記されています。ZFS のディスク内では、実際のマジックナンバーが Big Endian で“0x00bab10c”になっています(駄洒落のようです)。Little Endian ではバイトシーケンスが逆で、“0x0cb1ba00”です。UFS のスーパーブロックと同様、プールの内容にアクセスするのに必要な情報を含んでいるラベル部分です(メタデータ)。

UFS
Unix File System : ユニックスファイルシステム

UNIX 系列の OS において使用されるファイルシステム。また固有のファイルシステムを指す言葉ではなく、Version 7 Unix のファイルシステムおよびそこから派生した一連のファイルシステムの総称。一般に UFS と呼ぶ場合は 4.2BSD で実装された Fast File System(FFS)のことを指す場合が多く、他には FFFS、UFS2、UFS Logging 等が存在します。

USB イメージ

1 つのファイル内でオペレーティングシステム全体を構成するソフトウェアのコレクション。USB フラッシュドライブにコピーできるイメージ。コピーの唯一の方法が usbcopy ユーティリティーであり、このユーティリティーは OpenSolaris 上で使用できます。USB イメージはブート可能なイメージであり、インストールやその他の目的に使用できます。

【注意】Distribution Constructor では、他のタイプのフラッシュメモリデバイスで動作する可能性のある USB イメージが出力されますが、OpenSolaris にドライバのサポートがないために他のデバイスが動作しない可能性があります。

/usr ファイルシステム

スタンドアロンシステムまたはサーバー上のファイルシステム。標準 UNIX プログラムの多くが格納されています。ローカルコピーを保持する代わりに、大きな“/usr”ファイルシステムをサーバーと共有することにより、システム上で Solaris ソフトウェアをインストールおよび実行するために必要なディスク容量を最小限に抑えることができます。

VTOC
Volume Table of Contents : ブイトック

メインフレームなどで磁気ディスク装置内のデータセットの位置情報などを記録するための領域

/var ファイルシステム

システムの存続期間にわたって変更または増大が予想されるシステムファイルが格納されている(スタンドアロンシステム上の)ファイルシステムまたはディレクトリ。これらのファイルには、システムログ、“vi”ファイル、メールファイル、UUCP ファイルなどがあります。

VxFS
VERITAS File System :
ベリタスファイルシステム

VERITAS 社の開発した最初の商用ジャーナルファイルシステムであり、エクステント（ファイルのプリアロケーション機能）を採用したファイルシステム。OEM 供給されることが多く、VxFS は HP-UX オペレーティングシステムの主要ファイルシステムとして使われています（ただし、HP-UX はそれを JFS と呼んでいます）。他にも AIX、Linux、Solaris、UnixWare などでサポートされています。VxFS は、本来 AT&T の UNIX Systems Laboratories のために開発されました。VxFS は VERITAS Foundation Suite の一部としてパッケージ化されています（他に VERITAS Volume Manager を含む）。

ZFS
Zettabyte File System:
ゼタバイトファイルシステム

ストレージプールを使用して物理ストレージを管理する Sun Microsystems の新しいファイルシステム。

ZFS ファイルシステム

ZFS ストレージプール内のシステム名前空間内にマウントされ、別のファイルシステムのように動作する「ファイルシステムタイプの ZFS データセットです。

ZIL
ZFS Intent Log

ZFS の書き込みログです。コミットが終了すると開放されます。SSD や NVRAM を使用すると効果的です。

五十音順用語

アクション

パッケージの名前とキー属性。パッケージはファイル、ディレクトリ、リンク、ドライバ、依存関係を決められた形式にまとめたコレクションです。このコレクションはパッケージのインストール可能なオブジェクトを表します。このコレクションをアクションと呼びます。

アップグレード

ファイルを既存のファイルとマージし、可能な場合には変更を保持するインストール。

OpenSolaris OS のアップグレードでは、OpenSolaris OS の新しいバージョンがシステムのディスク上の既存のファイルにマージされます。アップグレードでは、既存の OS に対して行った変更は可能な限り保存されます。

OpenSolaris リリースでは、現在のイメージに含まれているすべてのインストール済みパッケージを、利用可能な最新のバージョンにアップグレードする場合、“pkg image-upgrade”コマンドを使用します。

イメージ

オペレーティングシステム全体を構成する、パッケージ内のソフトウェアのコレクション。このパッケージはインストールに適しています。

仮想デバイス

ZFS プール内の論理デバイス。物理デバイス、ファイル、または一連のデバイスを仮想デバイスに設定できます。

カタログ

オーソリティーによって公開されるリポジトリ内のすべてのパッケージ。カタログ内のパッケージは、特定のオーソリティーに関連付けられています。

クライアント

通信用のクライアントサーバーモデルでは、計算機能や大容量のメモリといったサーバーの資源にリモートアクセスするプロセスがクライアントに相当します。

クローン

正確なコピー。インストールの場合、クローンはオペレーティングシステム、ファイルシステム、またはボリュームの正確なコピーを指すことがあります。このコピーは、元の内容と完全な互換性があります。

再同期化

あるデバイスのデータを別のデバイスに転送する処理のことを“再同期化”と言います。たとえば、ミラーコンポーネントが置き換えられてオフラインになっている場合には、最新のミラーコンポーネントのデータが新しく復元されたミラーコンポーネントにコピーされます。この処理は、従来のボリューム管理製品では“ミラー再同期化”と呼ばれています。

スナップショット

特定の時点でのファイルシステムまたはブート環境の読み取り専用イメージ。スナップショットはブート可能ではありません。

スライス

ソフトウェアごとに分割される、ディスク領域の区分。

スワップ空間

メモリに再ロードできる状態になるまで、メモリ領域の内容を一時的に保持するスライスまたはファイル。“/swap”または“swap ボリューム”とも呼ばれます。

属性

パッケージまたはアクションの設定の表現。

大域ゾーン

Solarisゾーンでは、大域ゾーンはシステムのデフォルトのゾーンであり、システム全体での管理に使用されるゾーンでもあります。非大域ゾーンの構成、インストール、管

理、およびアンインストールは、大域ゾーンからのみ行うことができます。物理デバイス、ルーティング、動的再構成(DR)といったシステムインフラストラクチャーの管理は、大域ゾーンでのみ行うことができます。大域ゾーンで実行されるプロセスは、適切な権限が付与されていれば、他のゾーンに関連付けられているオブジェクトにもアクセスできます。

チェックサム

ファイルシステムブロック内の 256 ビットのハッシュデータ。チェックサム機能には、単純で高速な fletcher2(デフォルト)から SHA256 などの暗号強度の高いハッシュまで、様々なものがあります。

データセット

クローン、ファイルシステム、スナップショット、またはボリュームなど、ZFS で扱うエンティティの総称名です。各データセットは、ZFS 名前空間内で一意の名前で識別されます。元は、IBM メインフレームコンピュータで扱うファイルを指す用語です。

パッケージ

定義された形式での、ファイル、ディレクトリ、リンク、ドライバ、および依存関係のコレクション。

ハッシュ

入力よりもかなり短い数値を生成する処理によって得られる数値。同じ入力に対しては、常に同じ値が出力されます。ハッシュ関数は、テーブル検索アルゴリズム、エラー検出、改ざん検出などに使用できます。改ざん検出に使用する場合は、同じ結果を生成する別の入力を見つけにくいようなハッシュ関数を選択します。1 方向のハッシュ関数の一例としては、MD5 および SHA-1 があります。たとえば、メッセージダイジェストはディスクファイルなどの可変長入力を受け取り、小さい値に変換します。

ハッシュ化

文字列を変換して、この元の文字列を表す値(キー)を得る処理。

非大域ゾーン

Solaris オペレーティングシステムの単一のインスタンス内に作成された仮想オペレーティングシステム環境。非大域ゾーンでは、システムの他の部分と相互に作用することなく、1 つ以上のアプリケーションを実行できます。非大域ゾーンはゾーンとも呼ばれます。

ファイルサーバー

ネットワーク上のシステムに対して、ソフトウェアやファイルの記憶領域を提供するサーバー。

ブート環境(BE)

マウントポイント、ファイルシステム、ZFS データセット、および場合によっては非大域ゾーンのセットで構成されるブート可能な OpenSolaris 環境のインスタンス。ブート環境は、Solaris OS のオペレーションにきわめて重要な必須ファイルシステムの集まりです。アクティブなブート環境とは、現在ブートしている環境を指します。単一のアクティブなブート環境からだけブートできます。アクティブでないブート環境とは、現在ブートしていないが、次回のリブート時にアクティブ化できる状態にある環境のことを指します。

プール

デバイスの論理グループ。使用可能なストレージのレイアウトおよび物理特性を記述します。データセットの領域は、プールから割り当てられます。

ボリューム

物理デバイスをエミュレートするために使用するデータセット。たとえば、スワップデバイスとして ZFS ボリュームを作成できます。

ミラー

ZFS では、複数のディスク上にデータの同一コピーを格納する仮想デバイス。ミラー内のいずれかのディスクに障害が発生した場合は、そのミラー内の他の任意のディスクが同じデータを提供できます。

マウントポイント

リモートマシン上に存在するファイルシステムのマウント先となる、ワークステーション上のディレクトリ。

ルート(/)ファイルシステム

他のすべてのファイルシステムの元となる最上位ファイルシステム。ルート(/)ファイルシステムは他のすべてのファイルシステムがマウントされる元となり、マウント解除されることはありません。ルート(/)ファイルシステムには、カーネル、デバイスドライバ、システムの起動(ブート)に使用されるプログラムなど、システムの稼働に不可欠なディレクトリやファイルが含まれています。

ルートディレクトリ(root)

他のすべてのディレクトリの元となる最上位ディレクトリ。

ロールバック

特定のトランザクションより前に実行されたブート環境に戻すこと。ブート環境のアクティブ化の処理中や、ブート対象として指定したブート環境に問題または望ましくない動作が発生する場合にはロールバックを行います。ロールバックは、Solaris Live Upgrade ではフォールバックと呼ばれます。

論理デバイス

システムで単一のデバイスとして扱われる、1 つまたは複数のディスク上にある物理スライスの集まり。論理デバイスは、Solaris ボリュームマネージャではボリュームと呼ばれています。ボリュームは、アプリケーションやファイルシステムにとって物理ディスクと同じように機能します。

付録 A. ZFS CLI コマンドリファレンス

シェルにおけるコード例を次に示します。

- C シェル

machine_name% **command y|n** [*filename*]

- C シェルのスーパーユーザー

machine_name# **command y|n** [*filename*]

- Bourne シェルおよび Korn シェル

$ **command y|n** [*filename*]

- Bourne シェルおよび Korn シェルのスーパーユーザー

command y|n [*filename*]

“[ ]”は省略可能な項目を示します。上記の例は、*filename* を省略してもよいことを示しています。

“|”は区切り文字(セパレータ)です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英字で、頭文字を大文字で示します。ただし、キーボードによっては“Enter キー”が“Return キー”の動作をします。

ハイフン(-)は 2 つのキーを同時に押すことを示します。たとえば、“Ctrl-A”は“Control キー”を押したまま“A キー”を押すことを意味します。

コマンドライン入力の完了機能を使用すれば、コマンドの入力や繰り返しを簡単に行うことができます。次にコマンドライン入力完了機能のキー操作を示します。

表 A-1. コマンドライン入力完了機能

キー操作	説明
Tab	コマンドライン入力を完了します。最低 1 文字を入力してから Tab キーを押すと、コマンドライン入力が完了します。複数の候補が存在する場合は、Tab キーを繰り返し押すと、すべての候補が順に表示されます。
↑	前に入力したコマンドを逆順にスクロールします。
↓	前に入力したコマンドを正順にスクロールします。
Ctrl-A	コマンドラインの先頭にカーソルを移動します。
Ctrl-E	コマンドラインの最後の位置にカーソルを移動します。

A-1. zpool コマンド

zpool コマンドは、ZFS ストレージプールを構成します。ストレージプールは、ZFS データセット用の物理ストレージおよびデータ複製を提供するデバイスの集合体です。

ストレージプール内部のすべてのデータセットは、同一の領域を共有します。

次に zpool コマンドの概要を示します。No 欄の番号は、表 A-1-2 の参照番号です。

表 A-1-1. zpool コマンド概要

No	コマンド	概要
1	[-?]	ヘルプメッセージを表示します。
2	create [-fn] [-o property=*value*] ... [-O *file-system-property=value*] ... [-m *mountpoint*] [-R *root*] *pool vdev* ...	コマンド行で指定した仮想デバイスを含む新規ストレージプールを作成します。
3	destroy [-f] *pool*	指定したプールを破棄し、すべてのデバイスを解放してほかの用途で使用できるようにします。
4	add [-fn] *pool vdev*	指定された仮想デバイスを指定したプールに追加します。
5	remove *pool vdev*	指定された vdev をプールから削除します。
6	list [-H] [-o *property*[,...]] [*pool*] ...	指定したプールについて、健全性状態および領域使用状況を一覧表示します。
7	iostat [-v] [*pool*] ... [*interval* [*count*]]	指定したプールの I/O 統計を表示します。
8	status [-xv] [*pool*] ...	指定したプールの詳細な健全性状態を表示します。
9	offline [-t] *pool device* ...	指定した物理デバイスをオフラインにします。
10	online *pool device* ...	指定した物理デバイスをオンラインにします。
11	clear pool [*device*] ...	プール内のデバイスエラーをクリアします。
12	attach [-f] *pool device new_device*	new_device を既存の zpool デバイスに接続します。
13	detach *pool device*	device をミラーから切り離します。
14	replace [-f] *pool device* [*new_device*]	old_device を new_device で置き換えます。
15	scrub [-s] *pool* ...	消し込みを開始します。
16	export [-f] *pool*	指定したプールをシステムからエクスポートします。
17	import [-d dir | -c cachefile] [-D]	インポートに使用可能なプールを一覧表示します。
18	import [-o *mntopts*] [-o *property=value*] ... [-d *dir* | -c *cachefile*] [-D] [-f] [-R *root*] *pool* |*id* [*newpool*]	特定のプールをインポートします。
19	import [-o *mntopts*] [-o *property=value*] ... [-d *dir* | -c *cachefile*] [-D] [-f] [-R *root*] -a	検索ディレクトリ内で検出されたプールをすべてインポートします。
20	upgrade	ディスク上にある ZFS とは別のバージョンを使ってフォーマットされたプールをすべて表示します。
21	upgrade -v	現在のソフトウェアでサポートされている ZFS バージョンを表示します。
22	upgrade [-V *version*] -a | *pool* ...	指定したプールを最新のオンディスクバージョンにアップグレードします。
23	history [-il] [*pool*] ...	指定したプールのコマンド履歴を表示します。
24	get "all" | *property*[,...] *pool* ...	指定されたストレージプールのプロパティを表示します。

25	set *property=value pool*	指定したプールのプロパティを設定します。

表 A-1-2. zpool コマンドリファレンス

No	コマンド名	概要
	コマンド形式	
	解説	
	使用例	
1	[-?]	ヘルプメッセージを表示します。
	zpool [-?]	
	zpool コマンドのキーワード、および使用方法について、簡潔な説明を表示します。	
	使用例 1：zpool [-?]コマンドの例を示します。 root@opensolaris:~# **zpool -?** ⏎ usage: zpool command args ... where 'command' is one of the following: create [-fn] [-o property=value] ... [-O file-system-property=value] ... [-m mountpoint] [-R root] <pool> <vdev> ... destroy [-f] <pool> ・ ・ ・ get <"all" | property[,...]> <pool> ... set <property=value> <pool> root@opensolaris:~#	
2	create	コマンド行で指定した仮想デバイスを含む新規ストレージプールを作成します。
	zpool create [-fn] [-o *property=value*] ... [-O *file-system-property=value*] ... [-m *mountpoint*] [-R *root*] *pool vdev* ...	
	プール名の先頭は文字でなければなりません。また、プール名に含めることができるのは、英数字、下線"_"、ハイフン"-"、およびピリオド"."です。プールは予約されている「mirror」、「raidz」、および「spare」と、パターン「c[0-9]」で始まるデバイス名を設定できます。 このコマンドは、指定された各デバイスがアクセス可能であり、現在別のサブシステムにより使用されていないことを確認します。ある種の用法(現在マウントされている、専用のダンプデバイスとして指定されているなど)では、ZFS によるデバイスの使用が妨げられます。"-f"オプションを使用すると、その他の用法(既存の UFS ファイルシステムの使用など)を上書きできます。 このコマンドは、プールの複製方法が一貫しているかどうかもチェックします。"-f"オプションを指定せずに、単一のプール内で冗長ストレージと非冗長ストレージの結合を試みたり、ディスクとファイルの混在を試みると、エラーが発生します。"-f"オプションを設定せずに、単一の raidz またはミラーグループ内でサイズの異なるデバイスを使用しても、エラーのフラグが付けられます。 "-R"オプションを指定しない場合の、デフォルトのマウントポイントは「*/pool*」です。このマウントポイントは存在しないか、空でなければなりません。空でない場合、ルートデータセットをマウントすることはできません。これは、"-m"オプションを使って上書きできます。 -f　使用中と表示されていたり、競合する複製レベルが指定されていたとしても、vdev の使用を強制します。この方法で、すべてのデバイスを上書きできるわけではありません。 -n　使用する設定を、実際にプールを作成せずに表示します。実際のプール作成は、権限の不足またはデバイス共有のために失敗する可能性があります。 -o *property=value*　指定したプールプロパティをセットします。 -O *file-system-property=value*　プールのルートファイルシステムで指定したファイルシステムプロパティをセットします。 -R *root*　代替 *root* を使ってプールを作成します。この操作の一環として、ルートデータセットのマウントポイントが「/」に設定されます。 -m *mountpoint*　ルートデータセットのマウントポイントを設定します。デフォルトのマウントポ	

	イントは、「*/pool*」です。このマウントポイントは絶対パス、「legacy」または「none」でなければなりません。
	使用例 1：zpool create コマンドの例を示します。 root@opensolaris:~# **zpool create tpool raidz2 c9t0d0 c9t1d0 c9t2d0 c9t3d0 c9t4d0 c9t5d0 spare c9t6d0** ⏎ root@opensolaris:~#
3	destroy　指定したプールを破棄し、すべてのデバイスを開放して他の用途で使用できるようにします。
	zpool destroy [-f] *pool*
	このコマンドは、プールを破棄する前に、アクティブなデータセットすべてのマウント解除を試みます。 -f　プール内に含まれるアクティブなデータセットすべてのマウント解除を強制的に行います。
	使用例 1：zpool destroy コマンドの例を示します。 root@opensolaris:~# **zpool destroy tpool** ⏎ root@opensolaris:~#
4	add　指定された仮想デバイスを指定したプールに追加します。
	zpool add [-fn] *pool vdev ...*
	-f　使用中と表示されていたり、競合する複製レベルが指定されていたとしても、vdev の使用を強制します。この方法で、すべてのデバイスを上書きできるわけではありません。 -n　vdev を実際に追加することなく、使用される設定を表示します。実際のプール作成は、権限の不足またはデバイス共有のために失敗する可能性があります。 現在設定されているディスクは、定足数デバイスとして zpool に追加しないでください。ディスクが一度 zpool に追加されると、そのディスクは定足数デバイスとして設定可能になります。
	使用例 1：zpool add コマンドの例を示します。 root@opensolaris:~# **zpool add tpool c9t7d0** ⏎ root@opensolaris:~#
5	remove　指定された vdev をプールから削除します。
	zpool remove *pool vdev*
	このコマンドが現在サポートしているのは、ホットスペアの削除だけです。ミラーの一部となっているデバイスは、“zpool detach”コマンドを使用して削除できます。raidz と最上位 vdev は、プールから削除できません。
	使用例 1：zpool remove コマンドの例を示します。 root@opensolaris:~# **zpool remove tpool c9t7d0** ⏎ root@opensolaris:~#
6	list　指定したプールについて、健全性状態および領域使用状況を一覧表示します。
	zpool list [-H] [-o *property*[,...]] [*pool*] ...
	引数を指定しない場合、システム内のすべてのプールが表示されます。 -H　スクリプトモード。ヘッダーを表示せず、フィールドを任意の空白文字ではなく単一のタブで区切ります。 -o *property*　表示するフィールドのコンマ区切りのリスト。各フィールドには、次のいずれかを指定する必要があります。 name　Pool name size　Total size used　Amount of space used available　Amount of space available capacity　Percentage of pool space used health　Health status デフォルトはすべてのフィールドです。 このコマンドにより、ストレージプールで使用可能な実際の物理容量がレポートされます。物理容量は、内部の全データセットが実際に使用可能な総容量とは異なる場合があります。raidz 設定で使用される容量は、書き込まれるデータの特性により異なります。また、ZFS は、“zfs”コマンドが考慮に入れる内部アカウンティングのための容量を確保します。しかし、“zpool”コマンドでは、内部アカウンティングを考慮しません。容量に余裕のある適切なサイズのプールの場合、これらの影響は見えません。小さなプールまたは容量全体がほぼ使用されているプールの場合、これらの相違はより目立つようになります。

使用例 1：zpool list コマンドの例を示します。

```
root@opensolaris:~# zpool list ⏎
NAME  SIZE USED AVAILCAPHEALTHALTROOT
rpool8.94G3.50G5.44G39% ONLINE -
tpool11.9G2.81M  11.9G  0% ONLINE -
root@opensolaris:~#
```

7 | iostat | 指定したプールの I/O 統計を表示します。

zpool iostat [-v] [*pool*] ... [*interval* [*count*]]

間隔を指定した場合、Ctrl-C キーを押すまで、統計が *interval* 秒ごとに出力されます。*pools* を指定しない場合、システム内のすべてのプールの統計が表示されます。*count* を指定した場合、このコマンドは *count* レポートの出力後に配置されます。

-v　詳細な統計。プール全体の統計と共に、プール内の各 *vdevs* の使用状況統計をレポートします。

使用例 1：zpool iostat コマンドの例を示します。

```
root@opensolaris:~# zpool iostat -v ⏎
                           capacity     operations    bandwidth
pool                    used  avail   read  write   read  write
----------------------  -----  -----  -----  -----  -----  -----
---                                       --              -      -
rpool                   3.50G  5.44G      9      2   596K  43.2K
  c7d0s0                3.50G  5.44G      9      2   596K  43.2K
----------------------  -----  -----  -----  -----  -----  -----
---                                       --              -      -
tpool                   2.81M  11.9G      0     37     81  72.0K
  raidz2                2.81M  11.9G      0     37     81  72.0K
    c9t0d0                  -      -      1     24  64.9K   262K
    c9t1d0                  -      -      1     23  64.9K   262K
    c9t2d0                  -      -      1     24  65.6K   263K
    c9t3d0                  -      -      1     24  65.6K   263K
    c9t4d0                  -      -      1     23  64.9K   262K
    c9t5d0                  -      -      1     23  65.6K   261K
----------------------  -----  -----  -----  -----  -----  -----
---                                       --              -      -

root@opensolaris:~#
```

8 | status | 指定したプールの詳細な健全性状態を表示します。

zpool status [-xv] [*pool*] ...

pool を指定しない場合、システムの各プールの状態が表示されます。

消し込みまたは再同期化が進行中の場合、このコマンドは、実行済みの割合(パーセント)および推定完了時間をレポートします。プール内のデータ量およびシステムのその他のワークロードは変化する場合があるため、これらはどちらも概算値に過ぎません。

-x　エラーが発生しているか、使用不可能なプールの状態だけを表示します。
-v　詳細なデータエラー情報を表示し、前回のプール消し込みが完了して以降のデータエラーすべての完全なリストを出力します。

使用例 1：zpool status コマンドの例を示します。

```
root@opensolaris:~# zpool status tpool ⏎
  pool: tpool
 state: ONLINE
 scrub:  none requested
config:

        NAME  STATE   READ WRITE CKSUM
        tpool  ONLINE       0     0     0
          raidz2  ONLINE       0     0     0
```

```
            c9t0d0 ONLINE       0     0     0
            c9t1d0 ONLINE       0     0     0
            c9t2d0 ONLINE       0     0     0
            c9t3d0 ONLINE       0     0     0
            c9t4d0 ONLINE       0     0     0
            c9t5d0 ONLINE       0     0     0
        spares
          c9t6d0 AVAIL

errors: No known data errors
root@opensolaris:~#
```

9	offline	指定した物理デバイスをオフラインにします。

zpool offline [-t] *pool device* ...

オフライン状態にある *device* に対して、読み取りや書き込みは行われません。
このコマンドはスペアには適用されません。

-t 「temporary」。リブートすると、指定した物理デバイスは以前の状態に戻ります。

使用例 1：zpool offline コマンドの例を示します。

```
root@opensolaris:~# zpool offline tpool c9t0d0 ⏎
root@opensolaris:~#
```

10	online	指定した物理デバイスをオンラインにします。

zpool online *pool device* ...

このコマンドは、スペアには適用されません。

使用例 1：zpool online コマンドの例を示します。

```
root@opensolaris:~# zpool online tpool c9t0d0 ⏎
root@opensolaris:~#
```

11	clear	プール内のデバイスエラーをクリアします。

zpool clear *pool* [*device*] ...

引数を指定しない場合、プール内のすべてのデバイスエラーがクリアされます。
1 つ以上のデバイスを指定すると、指定したデバイスに関連するエラーだけがクリアされます。

使用例 1：zpool clear コマンドの例を示します。

```
root@opensolaris:~# zpool clear tpool c9t0d0 ⏎
root@opensolaris:~#
```

12	attach	*new_device* を既存の zpool デバイスに接続します。

zpool attach [-f] *pool device new_device*

既存のデバイスを、raidz 構成の一部にすることはできません。*device* が現在ミラー構成の一部ではない場合、*device* は *device* および *new_device* の 2 方向のミラーに自動的に変換されます。*device* が 2 方向ミラーの一部である場合、*new_device* を接続すると 3 方向ミラーが作成され、以下同様に作成されます。どの場合でも、*new_device* の再同期化がすぐに開始されます。

-f *new_device* が使用中と表示されている場合でも、これを強制的に使用します。
この方法で、すべてのデバイスを上書きできるわけではありません。

使用例 1：zpool attach コマンドの例を示します。

```
root@opensolaris:~# zpool attach -f rpool c7d0s0 c7d1s0 ⏎
Please be sure to invoke installgrub(1M) to make 'c7d1s0' bootable.
root@opensolaris:~#
```

13	detach	*device* をミラーから切り離します。

zpool detach *pool device*

データの有効な複製がほかに存在しない場合、この操作は拒否されます。

使用例 1：zpool detach コマンドの例を示します。

```
root@opensolaris:~# zpool detach tpool c9t0d0 ⏎
root@opensolaris:~#
```

14	replace	*old_device* を *new_device* で置き換えます。

zpool replace [-f] *pool old_device* [*new_device*]

これは、*new_device* を接続し、再同期化の実行まで待機してから *old_device* を切り離す操作と同じです。

new_device のサイズは、ミラーまたは raidz 構成内の全デバイスの最小サイズ以上でなければなりません。

new_device を指定しない場合、デフォルトの *old_device* が使用されます。この置換形式は、既存のディスクで障害が発生したためにディスクを物理的に交換したあとで実行する場合に役立ちます。この場合、実際には別のディスクであっても、新規ディスクには以前のデバイスと同じ“/dev/dsk”パスが使用されます。これは、ZFS により認識されます。

-f　　*new_device* が使用中と表示されている場合でも、これを強制的に使用します。
この方法で、すべてのデバイスを上書きできるわけではありません。

使用例 1：zpool replace コマンドの例を示します。
root@opensolaris:~# **zpool replace tpool c9t0d0 c9t6d0** ⏎
root@opensolaris:~#

15	scrub	消し込みを開始します。

zpool scrub [-s] *pool* ...

消し込みにより、指定したプール内のすべてのデータが検査され、チェックサムが適正に検証されます。複製された(ミラーまたは raidz)デバイスの場合、消し込み中に検出されたすべての損傷は、ZFS により自動的に修復されます。また、完了時には消し込み結果の概要を出力します。

消し込みおよび再同期化は、非常に類似した操作です。相違点は、再同期化では ZFS が期限切れであることを認識しているデータだけが検査されますが(新規デバイスをミラーに接続する場合や、既存のデバイスを置換する場合など)、消し込みではすべてのデータが検査されるため、ハードウェア障害やディスク障害に起因する非表示のエラーも検出されます。

消し込みと再同期化は、I/O 集約的な操作であるため、ZFS では一度に 1 つの操作だけが許可されます。進行中の消し込みが存在する場合に“zpool scrub”コマンドを実行すると、進行中の消し込みが終了して、新規消し込みが開始されます。進行中の再同期化が存在する場合、ZFS は、再同期化が完了するまで書き込みの開始を許可しません。

-s　　消し込みを停止します。

使用例 1：zpool scrub コマンドの例を示します。
root@opensolaris:~# **zpool scrub rpool** ⏎
root@opensolaris:~#

16	export	指定したプールをシステムからエクスポートします。

zpool export [-f] *pool* ...

すべてのデバイスにエクスポート済みのマークが付けられますが、別のサブシステムからは引き続き使用中と見なされます。十分な数のデバイスが存在するかぎり、このデバイスをシステム間(エンディアンの異なるシステムを含む)で移動またはインポートすることが可能です。

プールをエクスポートする前に、プール内のすべてのデータセットがマウント解除されます。

プールを可搬性のあるものにするため、“zpool”コマンドに単なるスライスではなく、ディスク全体を指定する必要があります。これにより、ZFS は可搬性のある EFI ラベルをディスクに設定します。これを行わない場合、エンディアンの異なるプラットフォーム上のディスクドライバは、ディスクを認識しません。

-f　　“unmount -f”コマンドを使用して、すべてのデータセットを強制的にマウント解除します。

使用例 1：zpool export コマンドの例を示します。
root@opensolaris:~# **zpool export tpool** ⏎
root@opensolaris:~#

17	import	インポートに使用可能なプールを一覧表示します。

zpool import [-d *dir* | -c *cachefile*] [-D]

“-d”オプションを指定しない場合、このコマンドは“/dev/dsk”内のデバイスを検索します。“-d”オプションは、複数回指定できます。このオプションを指定すると、すべてのディレクトリが検索されます。デバイスがエクスポートされたプールの一部として表示される場合、このコマンドは、プールの名前、数値識別子、および各デバイスやファイル

の *vdev* レイアウトと現在の健全性を含むプールの概要を表示します。"-D"オプションを指定しないかぎり、破棄されるプールや"zpool destroy"コマンドを使って以前に破棄されたプールは表示されません。

数値識別子は一意であり、エクスポートされた同名のプールを複数利用可能な場合にプール名の代わりに使用できます。

-c *cachefile*	*cachefile* プールプロパティで作成された、特定の cachefile から構成を読み取ります。
-d *dir*	デバイスまたはファイルを *dir* 内で検索します。"-d"オプションは複数回指定できます。
-D	破棄されたプールだけを一覧表示します。

使用例 1：zpool import コマンドの例を示します。

```
root@opensolaris:~# zpool import ⏎
   pool: tpool
     id: 10914955493912306664
  state: ONLINE
action: The pool can be imported using its name or numeric identifier.
config:

        tpool  ONLINE
          raidz2  ONLINE
            c9t0d0 ONLINE
            c9t1d0 ONLINE
            c9t2d0 ONLINE
            c9t3d0 ONLINE
            c9t4d0 ONLINE
            c9t5d0 ONLINE
        spares
          c9t6d0
root@opensolaris:~#
```

18 | import | 特定のプールをインポートします。

zpool import [-o *mntopts*] [-o *property=value*] ... [-d *dir* | -c *cachefile*] [-D] [-f] [-R *root*] *pool* |*id* [*newpool*]

プールは、名前または数値識別子を使って識別できます。*newpool* を指定すると、名前 *newpool* を使ってプールがインポートされます。それ以外の場合は、エクスポートされた名前と同じ名前を使ってインポートが行われます。

"zpool export"を最初に実行せずにシステムからデバイスを削除すると、そのデバイスは潜在的にアクティブな状態として表示されます。これがエクスポートの失敗を示すのか、デバイスが別のホストにより実際に使用されていることを示すのかを見分けることができません。この状態のプールをインポートするには、"-f"オプションを指定する必要があります。

-o *property=value*	指定したプールプロパティをセットします。
-c *cachefile*	*cachefile* プールプロパティで作成された、特定の cachefile から構成を読み取ります。
-d *dir*	デバイスまたはファイルを *dir* 内で検索します。"-d"オプションは複数回指定できます。
-D	破棄されたプールをインポートします。"-f"オプションも指定する必要があります。
-f	プールが潜在的にアクティブであると表示されている場合でも、インポートを強制的に実行します。
-o *mntopts*	プール内のデータセットのマウント時に使用するマウントオプションのコンマ区切りのリスト。
-R *root*	代替 root を使ってプールをインポートします。

使用例 1：zpool import コマンドの例を示します。

```
root@opensolaris:~# zpool import tpool ⏎
root@opensolaris:~#
```

19 | import | 検索ディレクトリ内で検出されたプールをすべてインポートします。

zpool import [-o *mntopts*] [-o *property=value*] ... [-d *dir* | -c *cachefile*] [-D] [-f] [-R *root*] -a

十分な数のデバイスを使用可能なプールがすべてインポートされることを除き、前のコマンドと同じです。“-D”オプションを指定しないかぎり、破棄されるプールや“zpool destroy”コマンドを使って以前に破棄されたプールはインポートされません。

-o *mntopts*	プール内のデータセットのマウント時に使用するマウントオプションのコンマ区切りのリスト。
-o *property=value*	指定したプールプロパティをセットします。
-c *cachefile*	*cachefile* プールプロパティで作成された、特定の cachefile から構成を読み取ります。
-d *dir*	デバイスまたはファイルを *dir* 内で検索します。“-d”オプションは複数回指定できます。
-D	破棄されたプールだけをインポートします。“-f”オプションも指定する必要があります。
-f	プールが潜在的にアクティブであると表示されている場合でも、インポートを強制的に実行します。
-a	検出されたプールをすべてインポートします。
-R *root*	代替 root を使ってプールをインポートします。

使用例 1：zpool import コマンドの例を示します。

```
root@opensolaris:~# zpool import -D -f -a ⏎
root@opensolaris:~#
```

20	upgrade	ディスク上にある ZFS とは別のバージョンを使ってフォーマットされたプールをすべて表示します。

zpool upgrade

以前のバージョンは引き続き使用できますが、一部の機能は利用できない場合があります。これらのプールは、“zpool upgrade -a”コマンドを使用してアップグレードできます。より新しいバージョンでフォーマットされたプールも表示されますが、システム上でこれらのプールにアクセスすることはできません。

使用例 1：zpool upgrade コマンドの例を示します。

```
root@opensolaris:~# zpool upgrade ⏎
This system is currently running ZFS pool version 14.

The following pools are out of date, and can be upgraded.  After being
upgraded, these pools will no longer be accessible by older software versions.

VERPOOL
------ ------------
 1 tpool

Use 'zpool upgrade -v' for a list of available versions and their associated
features.
root@opensolaris:~#
```

21	upgrade	現在ソフトウェアでサポートされている ZFS バージョンを表示します。

zpool upgrade -v

現在の ZFS バージョンおよびサポートされる以前のバージョンすべてが、各バージョンで提供される機能に関する説明と共に表示されます。

使用例 1：zpool upgrade コマンドの例を示します。

```
root@opensolaris:~# zpool upgrade -v ⏎
This system is currently running ZFS pool version 14.

The following versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
```

```
 7    Separate intent log devices
 8    Delegated administration
 9    refquota and refreservation properties
 10   Cache devices
 11   Improved scrub performance
 12   Snapshot properties
 13   snapused property
 14   passthrough-x aclinherit support
For more information on a particular version, including supported releases, see:

http://www.opensolaris.org/os/community/zfs/version/N

Where 'N' is the version number.
root@opensolaris:~#
```

22 | upgrade | 指定したプールを最新のオンディスクバージョンにアップグレードします。

zpool upgrade [-V *version*] -a | *pool* ...

これを実行すると、以前のバージョンのソフトウェアを実行するシステム上で、プールにアクセスすることはできなくなります。また、プールを作成後、“zpool set”コマンドでのダウングレードは実行できません。ただし、プール作成時に“-o”オプションを指定して作成することで、低いバージョンのプールが作成可能です。

-a	すべてのプールをアップグレードします。
-V *version*	指定されたバージョンにアップグレードします。“-V”オプションを指定しなかった場合、最新のバージョンにアップグレードされます。

使用例 1：zpool upgrade コマンドの例を示します。

```
root@opensolaris:~# zpool upgrade -V 10 tpool ⏎
This system is currently running ZFS pool version 14.

Successfully upgraded 'tpool' from version 1 to version 10

root@opensolaris:~#
```

23 | history | 指定したプールのコマンド履歴を表示します。

zpool history [-il] [*pool*] ...

ここでプールを指定しなかった場合、すべてのプールのコマンド履歴が表示されます。

-i	内部の ZFS イベントログに加えて、ユーザーが開始したイベントを表示します。
-l	ユーザー名、ホスト名、ゾーンを加えた長い形式で記録したログを表示します。

使用例 1：zpool history コマンドの例を示します。

```
root@opensolaris:~# zpool history tpool ⏎
History for 'tpool':
2009-10-14.17:07:31 zpool create tpool raidz2 c9t0d0 c9t1d0 c9t2d0 c9t3d0 c9t4d0
c9t5d0 spare c9t6d0
2009-10-14.17:08:19 zfs create tpool/data1
2009-10-14.17:08:38 zfs create tpool/data2
2009-10-14.17:11:02 zfs set mountpoint=/data1 tpool/data1
2009-10-14.17:12:51 zfs set mountpoint=legacy tpool/data2
2009-10-14.17:15:02 zfs set compression=on tpool/data1
2009-10-14.17:16:41 zfs create -o compression=on tpool/data3
2009-10-14.17:23:21 zfs set mountpoint=/tpool/data1 tpool/data1
2009-10-14.17:25:30 zfs snapshot tpool/data1@snap
2009-10-14.17:28:47 zfs rollback tpool/data1@snap
2009-10-14.17:29:38 zfs destroy tpool/data1@snap
2009-10-14.17:30:26 zfs destroy tpool/data1
2009-10-14.17:30:43 zfs destroy tpool/data2
2009-10-14.17:31:25 zfs destroy tpool/data3
2009-10-19.16:47:51 zpool clear tpool c9t0d0

root@opensolaris:~#
```

24	get	指定されたストレージプールのプロパティを表示します。

zpool get "all" | *property*[,...] *pool* ...

指定されたストレージプールのプロパティ、または“all”が指定されていればすべてのプロパティのリストを表示します。
プロパティごとに次の列が表示されます。

name Name of storage pool
Property Property name
value Property value
source Property source, either ‘default’ or ‘local’.

使用例 1：zpool get コマンドの例を示します。

```
root@opensolaris:~# zpool get all rpool ⏎
NAME PROPERTY VALUE SOURCE
rpool size 8.94G -
rpool used 3.44G -
rpool available 5.49G -
rpool capacity 38% -
rpool altroot - default
rpool health ONLINE -
rpool guid 6824308044772421765 default
rpool version 14 default
rpool bootfs rpool/ROOT/opensolaris-1 local
rpool delegation on default
rpool autoreplace off default
rpool cachefile - default
rpool failmode wait default
rpool listsnapshots off default
root@opensolaris:~#
```

25	set	指定したプールのプロパティを設定します。

zpool set *property=value pool*

指定されたプールのプロパティを、各データセット用に設定した値に設定します。

使用例 1：zpool set コマンドの例を示します。

```
root@opensolaris:~# zpool set listsnapshots=on tpool ⏎
root@opensolaris:~#
```

A-2. zfs コマンド

zfs コマンドは、ZFS ストレージプール内の ZFS データセットを構成します。データセットの識別には、ZFS 名前空間内部の一意のパスが使用されます。

次に zfs コマンドの概要を示します。No 欄の番号は、表 A-2-2 の参照番号です。

表 A-2-1. zfs コマンド概要

No	コマンド	概要
1	[-?]	ヘルプメッセージを表示します。
2	create [-p] [-o *property=value*] ... *filesystem*	新しい ZFS ファイルシステムを作成します。
3	create [-ps] [-b *blocksize*] [-o *property=value*] ... -V *size volume*	指定したサイズのボリュームを作成します。
4	destroy [-rRf] *filesystem*|*volume*|*snapshot*	指定されたデータセットを破棄します。
5	clone [-p] [-o *property=value*] ... *snapshot* *filesystem*|*volume*	指定したスナップショットのクローンを作成します。
6	promote *clone-filesystem*	特定のクローンファイルシステムへの移行を促し、そのファイルシステムが「元」のスナップショットに依存しないようにします。
7	rename *filesystem*|*volume*|*snapshot* [*filesystem*|*volume*|*snapshot*]	指定したデータセットの名前を変更します。
8	snapshot [-r] [-o *property=value*] ... *filesystem@snapname*|*volume@snapname*	指定した名前のスナップショットを作成します。
9	rollback [-rRf] *snapshot*	指定したデータセットを以前のスナップショットにロールバックします。
10	list [-rH] [-o *property*[,...]] [-t *type*[,...]] [-s *property*] ... [-S *property*] ... [*filesystem*|*volume*|*snapshot*] ...	指定したデータセットのプロパティ情報を、表形式で表示します。
11	set *property=value* *filesystem*|*volume*|*snapshot* ...	プロパティを、各データセット用に設定した値に設定します。
12	get [-rHp] [-o *field*[,*field*]...] [-s *source*[,*source*]...] *all* | *property*[,*property*] . *filesystem*|*volume*|*snapshot* ..	指定したデータセットのプロパティを表示します。
13	inherit [-r] *property* *filesystem*|*volume*|*snapshot* ...	指定したプロパティをクリアして、そのプロパティが祖先から継承されるようにします。
14	mount	マウントされている ZFS ファイルシステムを表示します。
15	mount [-vO] [-o *options*] -a | *filesystem*	使用可能な ZFS ファイルシステムをマウントします。
16	unmount [-f] -a | *filesystem*|*mountpoint*	マウントされているファイルシステムをマウント解除します。
17	share -a | *filesystem*	使用可能な ZFS ファイルシステムを共有します。
18	unshare -a *filesystem*|*mountpoint*	現在共有されている ZFS ファイルシステムを共有解除します。
19	send [-vR] [-[iI] *snapshot*] *snapshot*	snapshot2 のストリーム表現を作成します。
20	receive [-vnF] *filesystem*|*volume*|*snapshot* receive [-vnF] -d *filesystem*	スナップショットを作成します。内容は、標準入力から受信したストリームの指定に基づきます。
21	upgrade [-v]	最新のバージョンではないファイルシステムを

		表示します。
22	upgrade [-r] [-V *version*] -a | *filesystem*	すべてのプール、または指定したプールを最新のオンディスクバージョンにアップグレードします。
23	allow [-ldug] "everyone"|*user*|*group*[,...] *perm*|*@setname*[,...] *filesystem*|*volume* allow [-ld] -e *perm*|*@setname*[,...] *filesystem*|*volume*	ファイルシステムの ZFS 管理アクセス権を別のアクセス権を持っていないユーザーに付与します。
24	allow -c *perm*|*@setname*[,...] *filesystem*|*volume*	作成時のアクセス権を付与します。
25	allow -s *@setname perm*|*@setname*[,...] *filesystem*|*volume*	アクセス権を「アクセス権セット」にまとめて、指定します。
26	unallow [-rldug] "*everyone*"|*user*|*group*[,...] [*perm*|*@setname*[,...]] *filesystem*|*volume* unallow [-rld] -e [*perm*|*@setname*[,...]] *filesystem*|*volume* unallow [-r] -c [*perm*|*@setname*[...]] *filesystem*|*volume*	委任したアクセス権を削除します。
27	unallow [-r] -s *@setname* [*perm*|*@setname*[,...]] *filesystem*|*volume*	アクセス権セットのアクセス権を削除します。

表 A-2-2. zfs コマンドレファレンス

No	コマンド名	概要
	コマンド形式	
	解説	
	使用例	
1	?	ヘルプメッセージを表示します。
	zfs [-?]	
	zfs コマンドのキーワード、および使用方法について、簡潔な説明を表示します。	
	使用例 1：zfs [-?]コマンドの例を示します。 root@opensolaris:~# **zfs -?** ⏎ usage: zfs command args ... where 'command' is one of the following: create [-p] [-o property=value] ... <filesystem> create [-ps] [-b blocksize] [-o property=value] ... -V <size> <volume> destroy [-rRf] <filesystem|volume|snapshot> ・ ・ ・ unallow [-r] -c [<perm|@setname>[,...]] <filesystem|volume> unallow [-r] -s @setname [<perm|@setname>[,...]] <filesystem|volume> Each dataset is of the form: pool/[dataset/]*dataset[@name] For the property list, run: zfs set|get For the delegated permission list, run: zfs allow|unallow root@opensolaris:~#	
2	create	新しい ZFS ファイルシステムを作成します。
	zfs create [-p] [-o *property=value*] ... *filesystem*	
	ファイルシステムは、親から継承した“mountpoint”プロパティに従って自動的にマウントされます。 -p　同じ名前のファイルシステムが既に存在していても、強制的に上書きして作成されます。 -o property=value　データセットの作成時に“zfs set property=value”が呼び出された場合と同様に、指定されたプロパティを設定します。編集可能なすべての ZFS プロパティは、作成時にも設定可能です。複数の“-o”オプションを指定できます。複数の“-o”オプション内で同じプロパティを指定した場合は、エラーが発生します。	
	使用例 1：zfs create コマンドの例を示します。 root@opensolaris:~# **zfs create -o compression=on tpool/data3** ⏎ root@opensolaris:~#	
3	create	指定したサイズのボリュームを作成します。
	zfs create [-ps] [-b *blocksize*] [-o *property=value*] ... -V *size volume*	
	ボリュームは、“/dev/zvol/{dsk,rdsk}/*path*”内にブロックデバイスとしてエクスポートされます。ここで、*path* は ZFS 名前空間内のボリュームの名前です。このサイズは、デバイスによりエクスポートされる論理サイズを表します。デフォルトでは、同サイズの予約が作成されます。 *size* は、ボリュームが *blocksize* に関係なく整数のブロックを持つように、もっとも近い 128KB に自動的に切り上げられます。 -p　同じ名前のファイルシステムが既に存在していても、強制的に上書きして作成されます。 -s　予約なしで疎ボリュームを作成します。 -o property=value　データセットの作成時に“zfs set property=value”が呼び出された場合と同様に、指定されたプロパティを設定します。編集可能なすべての ZFS プロパティは、作成時にも設定可能です。複数の“-o”オプションを指定できます。複数の“-o”オプション内で同じプロパティを指定した場合は、エラーが発生します。 -b *blocksize*　“-o volblocksize=*blocksize*”と同等です。このオプションを“-o volblocksize”と	

	組み合わせて指定した場合の動作は、定義されていません。
	使用例 1：zfs create コマンドの例を示します。 root@opensolaris:~# **zfs create -s -V 30g tpool/data1**⏎ root@opensolaris:~#
4	destroy　指定されたデータセットを破棄します。
	zfs destroy [-rRf] *filesystem*|*volume*|*snapshot*
	デフォルトでは、このコマンドは共有中のファイルシステムすべての共有を解除し、マウント中のファイルシステムすべてをマウント解除し、アクティブな依存関係(子、スナップショット、クローン)を持つデータセットの破棄を拒否します。 -r　すべての子を再帰的に破棄します。スナップショットが指定された場合、子孫ファイルシステム内でこの名前を持つすべてのスナップショットを破棄します。 -R　ターゲット階層外にあるクローンファイルシステムを含む、すべての依存関係を再帰的に破棄します。スナップショットが指定された場合、子孫ファイルシステム内でこの名前を持つすべてのスナップショットを破棄します。 -f　“unmount -f”コマンドを使用してすべてのファイルシステムを強制的にマウント解除します。非ファイルシステムやマウント解除されたファイルシステムは、このオプションの影響を受けません。 “-r”や“-f”オプションを適用すると、プールのかなりの部分を破棄することが可能で、使用中のマウントされたファイルシステムで予期しない動作が発生する場合があるため、これらのオプションは特に注意深く使用するようにしてください。
	使用例 1：zfs destroy コマンドの例を示します。 root@opensolaris:~# **zfs destroy tpool/data1** ⏎ root@opensolaris:~#
5	clone　指定したスナップショットのクローンを作成します。
	zfs clone [-p] [-o *property=value*] ... *snapshot filesystem*|*volume*
	ターゲットのデータセットは、ZFS 階層内の任意の場所に配置できます。作成されたデータセットは元のデータセットと同じタイプになります。 -p　同じ名前のスナップショットが既に存在していても、強制的に上書きして作成されます。作成された時点で自動的にマウントします。 -o *property=value*　指定したプロパティをセットします。
	使用例 1：zfs clone コマンドの例を示します。 root@opensolaris:~# **zfs clone tpool/data1@snap tpool/data2** ⏎ root@opensolaris:~#
6	promote　特定のクローンシステムへの移行を促し、そのファイルシステムが「元」のスナップショットに依存しないようにします。
	zfs promote *clone-filesystem*
	このコマンドにより、クローンの作成元のファイルシステムを破棄できるようになります。クローンの親子依存関係が逆転し、「元」のファイルシステムが、指定されたファイルシステムのクローンになります。 クローンされたスナップショットと、そのスナップショットより前のすべてのスナップショットは、移行を促されたクローンによって所有されるようになります。それらが使用する領域は、「元」のファイルシステムから移行を促されたクローンに移されます。これは、それらのスナップショットを収容するための領域が不足しないようにするためです。この操作を行っても新しい領域は消費されませんが、領域のアカウンティングは調整されます。移行を促されたクローンは、自身の衝突するスナップショット名を持ってはいけません。“rename”サブコマンドを使えば、衝突するスナップショット名を変更できます。
	使用例 1：zfs promote コマンドの例を示します。 root@opensolaris:~# **zfs promote tpool/data2** ⏎ root@opensolaris:~#
7	rename　指定したデータセットの名前を変更します。
	zfs rename *filesystem*|*volume*|*snapshot filesystem*|*volume*|*snapshot*
	スナップショットを除き、新規ターゲットは ZFS 階層内の任意の場所に配置できます。スナップショットの名前を変更できるのは、親のファイルシステムまたはボリューム内だけです。スナップショットの名前を変更する場合、スナップショットの親ファイルシステムを 2 番目の引数として指定する必要はありません。名前の変更されたファイルシステムは、新しいマウントポイントを継承できます。この場合、ファイルシステムは、マウント解除されてから新しいマウントポイントで再

マウントされます。

使用例 1：zfs rename コマンドの例を示します。
root@opensolaris:~# **zfs rename tpool/data1 tpool/data3** ⏎
root@opensolaris:~#

8 | snapshot | 指定した名前のスナップショットを作成します。

zfs snapshot [-r] [-o *property=value*] ... *filesystem@snapname|volume@snapname*

-r　すべての子孫データセットのスナップショットを再帰的に作成します。スナップショットは原子的に取得されるため、再帰的スナップショットはすべて同じ時点のものになります。
-o property　指定したプロパティをセットします。

使用例 1：zfs snapshot コマンドの例を示します。
root@opensolaris:~# **zfs snapshot tpool/data1@snap3** ⏎
root@opensolaris:~#

9 | rollback | 指定したデータセットを以前のスナップショットにロールバックします。

zfs rollback [-rRf] *snapshot*

データセットをロールバックすると、スナップショット作成時から変更されたすべてのデータは破棄され、データセットがスナップショット作成時の状態に戻ります。デフォルトでは、このコマンドを使って、最新のスナップショット以外のスナップショットにロールバックすることはできません。最新でないスナップショットにロールバックする場合は、“-r”オプションを使って中間スナップショットをすべて破棄する必要があります。このファイルシステムは、必要に応じて、マウント解除および再マウントされます。

-r　指定したスナップショット以降のスナップショットをすべて再帰的に破棄します。
-R　指定したスナップショット以降のスナップショットを再帰的に破棄すると共に、これらのスナップショットのクローンもすべて破棄します。
-f　“unmount -t”コマンドを使用してすべてのファイルシステムを強制的にマウント解除します。

使用例 1：zfs rollback コマンドの例を示します。
root@opensolaris:~# **zfs rollback tpool/data1@snap** ⏎
root@opensolaris:~#

10 | list | 指定したデータセットのプロパティ情報を、表形式で表示します。

zfs list [-rH] [-o *property*[,...]] [-t *type*[,...]] [-s *property*] ... [-S *property*] ... [*filesystem|volume|snapshot*] ...

指定した場合、プロパティ情報を絶対パス名または相対パス名で表示できます。デフォルトでは、すべてのデータセットが表示されます。次のフィールドが含まれます。

- name
- used
- available
- referenced
- mountpoint

-H　スクリプティングモードで使用します。ヘッダーを出力せず、フィールドを任意の空白文字ではなく単一のタブで区切ります。
-r　データセットのすべての子をコマンド行に再帰的に表示します。
-o *property*　表示するプロパティのコンマ区切りのリスト。プロパティは、ZFS プロパティセットのいずれか、またはデータセット名を表示する特殊な値“name”を使用する必要があります。
-s *property*　プロパティの値に基づいて、出力を列で昇順にソートする場合に使用するプロパティ。プロパティは、ZFS プロパティセットのいずれか、またはデータセット名をソートする特殊な値“name”にする必要があります。複数の“-s”プロパティオプションを使用することで、一度に複数のプロパティを指定できます。複数の“-s”オプションは、左から右の順で、重要度の高さが判断されます。

以下にソート基準の一覧を示します。

- 数値型は、数値順にソートされる。
- 文字列型は、アルファベット順にソートされる。
- 行に適さない型は、指定したソート順に関係なく、その行がリテラルソート順の一番下になる。
- ソートオプションを指定しない場合は、“zfs list”の既存の動作が維持される。

-S *property*　-s オプションと同じですが、プロパティで降順にソートされる点が異なります。
-t *type*　表示するタイプのコンマ区切りのリスト。ここで、“type”は、“filesystem”、

	“snapshot”、“volume”のいずれかになります。例えば、“-t snapshot”を指定すると、スナップショットだけが表示されます。
	使用例 1：zfs list コマンドの例を示します。 root@opensolaris:~# **zfs list -t snapshot** ⏎ NAME USED AVAILREFER MOUNTPOINT rpool/ROOT/opensolaris-1@install 148M - 2.82G- rpool/ROOT/opensolaris-1@2009-10-19-07:32:50 20.0M - 2.87G- tpool/data2@snap 0 - 1.07M - root@opensolaris:~#
11	set　プロパティを、各データセット用に指定した値に設定します。
	zfs set *property=value filesystem|volume|snapshot* ...
	一部のプロパティのみ、編集可能です。数値は、正確な値を指定することも、サフィックス「B」、「K」、「M」、「G」、「T」、「P」、「E」、「Z」(それぞれ、バイト、キロバイト、メガバイト、ギガバイト、テラバイト、ペタバイト、エクサバイト、ゼタバイトを表します)を使って人間の読み取り可能な形式で指定することもできます。スナップショットにプロパティを設定することはできません。
	使用例 1：zfs set コマンドの例を示します。 root@opensolaris:~# **zfs set mountpoint=legacy tpool/data2** ⏎ root@opensolaris:~#
12	get　指定したデータセットのプロパティを表示します。
	zfs get [-rHp] [-o *field*[,*field*]...] [-s *source*[,*source*]...] *all* | *property*[,*property*]... *filesystem|volume|snapshot* ...
	データセットが指定されていない場合、このコマンドはシステムのすべてのデータセットのプロパティを表示します。プロパティごとに、次の列が表示されます。 name Dateset name Property Property name value Property value source Property source. Can either be local, default, temporary, inherited, or none (-). デフォルトではすべての列が表示されます。これは、“-o”オプションを使って制御できます。このコマンドには、コンマ区切りのプロパティリストを指定できます。 特殊な値“all”を使って、指定したデータセットのプロパティをすべて表示できます。 -r 任意の子のプロパティを再帰的に表示します。 -H スクリプトによる解析がより容易な形式で、出力を表示します。ヘッダーがすべて省略され、フィールドが任意の数の空白ではなく、タブ 1 つで明示的に区切られます。 -o *field* 表示する列のコンマ区切りのリスト。デフォルト値は、「name,property,value,source」です。 -s *source* 表示するソースのコンマ区切りのリスト。このリストにないソースからのプロパティは、無視されます。各ソースは、「local、default、inherited、temporary、none」のいずれかでなければなりません。デフォルト値はすべてのソースです。 -p 解析可能な(絶対)値で数を表示します。
	使用例 1：zfs get コマンドの例を示します。 root@opensolaris:~# **zfs get compression tpool/data1** ⏎ NAME PROPERTY VALUE SOURCE tpool/data1 compression on local root@opensolaris:~#
13	inherit　指定したプロパティをクリアして、そのプロパティが祖先から継承されるようにします。
	inherit [-r] *property filesystem|volume|snapshot* ...
	祖先にプロパティが設定されていない場合は、デフォルト値が使用されます。 -r すべての子で指定したプロパティを再帰的に継承します。
	使用例 1：zfs inherit コマンドの例を示します。 root@opensolaris:~# **zfs inherit compression rpool/export/home** ⏎ root@opensolaris:~#
14	mount　マウントされている ZFS ファイルシステムを表示します。
	zfs mount
	現在マウントされている ZFS ファイルシステムを表示します。

使用例 1：zfs mount コマンドの例を示します。
```
root@opensolaris:~# zfs mount ⏎
rpool/ROOT/opensolaris-1          /
rpool/export                      /export
rpool/export/home                 /export/home
rpool/export/home/taro            /export/home/taro
rpool                             /rpool
tpool                             /tpool
tpool/data2                       /tpool/data2
tpool/data3                       /tpool/data3
root@opensolaris:~#
```

15 | mount | 使用可能な ZFS ファイルシステムをマウントします。

zfs mount [-vO] [-o *options*] -a | *filesystem*

-o *options*	マウント時に一時的に使用する、コンマ区切りのマウントオプションリスト(省略可能)。
-O	オーバーレイマウントを実行します。
-v	マウントの進捗を示します。
-a	使用可能なすべての ZFS ファイルシステムをマウントします。 これは、ブートシステムの一部として自動的に呼び出されます。
filesystem	マウントするファイルシステムを指定します。

使用例 1：zfs mount コマンドの例を示します。
```
root@opensolaris:~# zfs mount -v -a ⏎
Reading ZFS config: done.
Mounting ZFS filesystems: (16/16)
root@opensolaris:~#
```

16 | unmount or umount | マウントされているファイルシステムをマウント解除します。

zfs unmount [-f] -a | *filesystem*|*mountpoint*

-f	ファイルシステムを、使用中であっても強制的にマウント解除します。
-a	現在マウントされているすべての ZFS ファイルシステムをマウント解除します。 これは、シャットダウンプロセスの一部として自動的に呼び出されます。
filesystem|*mount point*	指定したファイルシステムをマウント解除します。また、マウントポイントを指定することでマウント解除することもできます。

使用例 1：zfs unmount コマンドの例を示します。
```
root@opensolaris:~# zfs unmount -a ⏎
root@opensolaris:~#
```

17 | share | 使用可能な ZFS ファイルシステムを共有します。

zfs share -a | *filesystem*

-a	使用可能なすべての ZFS ファイルシステムを共有します。 これは、ブートプロセスの一部として自動的に呼び出されます。
filesystem	ファイルシステムは、“sharenfs”と“sharesmb”プロパティの設定時に共有されます。

使用例 1：zfs share コマンドの例を示します。
```
root@opensolaris:~# zfs share tpool/data1 ⏎
root@opensolaris:~#
```

18 | unshare | 現在共有されている ZFS ファイルシステムを共有解除します。

zfs unshare -a *filesystem*|*mountpoint*

-a	使用可能なすべての ZFS ファイルシステムを共有解除します。 これは、シャットダウンプロセスの一部として自動的に呼び出されます。
filesystem|*mount point*	指定したファイルシステムを共有解除します。 システム上で共有される ZFS ファイルシステムのパスを指定することもできます。

使用例 1：zfs unshare コマンドの例を示します。
```
root@opensolaris:~# zfs unshare -a ⏎
```

root@opensolaris:~#

19 | send | snapshot2 のストリーム表現を作成します。

zfs send [-vR] [-[iI] *snapshot*] *snapshot*

これは、標準出力に書き込まれます。出力は、ssh(1)などを使用して、ファイルまたは別のマシンにリダイレクトできます。デフォルトでは、完全なストリームが生成されます。

-i *snapshot*	*snapshot1* から *snapshot2* への増分ストリームを生成します。増分ソース *snapshot1* は、スナップショット名の最後の構成要素(例えば、"@"の後の部分)として指定できます。これは、*snapshot2* と同じファイルシステムに由来すると見なされます。 もしも出力先がクローンであれば、ソースは、元のスナップショットの場合があります。そのため、"@origin"ではなく、"pool/fs@origin"のように明確に指定しなければなりません。
-I *snapshot*	生成されるストリームパッケージに、snapshot1 から snapshot2 までの全ての中間スナップショットが送られます。例えば、"-I @a fs@d"と、"-i @a fs@b、-i @b fs@c、-i @c fs@d"は似た意味になります。増分ソーススナップショットは"-i"オプションのように指定したものです。
-R	指定したファイルシステム、および全ての親ファイルシステムの複製ストリームパッケージを、名前付けられたスナップショットに生成します。 "-R"オプションを、"-i"または"-I"オプションと組み合わせて使用した場合、増分複製ストリームが生成されます。プロパティの現在の値、および現在のファイルシステム名は、ストリームが受信された時にセットされます。"-F"オプションが指定され、このストリームを受信したとき、このストリームと、送信側に存在しなかったファイルシステムは破棄されます。
-v	生成されたストリームパッケージに関する詳細な情報を表示します。

ストリームの形式は、発展途上にあります。下位互換性は保証されていません。ZFS の将来のバージョンでは、使用しているストリームを受信できなくなる可能性があります。

使用例 1：zfs send コマンドの例を示します。

```
root@opensolaris:~# zfs send tpool/data1@snap | ssh taro@opensolaris dd of=snap ⏎
The authenticity of host 'opensolaris (::1)' can't be established.
RSA key fingerprint is d8:33:7a:71:81:61:55:82:46:7d:ed:a9:70:e4:f5:f
Are you sure you want to continue connecting (yes/no)? yes ⏎
Warning: Permanently added 'opensolaris' (RSA) to the list of known hosts.
Password: ○○○○○○○○ ⏎
501+2 records in
501+1 records out
root@opensolaris:~#
```

初回起動時のみ表示されます。

20 | receive | スナップショットを作成します。内容は、標準入力から受信したストリームの指定に基づきます。

receive [-vnF] *filesystem*|*volume*|*snapshot*
receive [-vnF] -d *filesystem*

ストリーム全体を受信する場合、新しいファイルシステムも作成されます。ストリームは"zfs send"サブコマンドによって作成されます。このサブコマンドはデフォルトで、完全なストリームを作成します。"zfs receive"の別名として、"zfs recv"を使用できます。

増分ストリームが受信された場合、対象となるファイルシステムがすでに存在しており、かつその最新のスナップショットがその増分ストリームのソースに一致する必要があります。対象となるファイルシステムおよびその子ファイルシステムがすべてマウント解除されるため、受信操作中にアクセスすることはできません。

このサブコマンドが作成するスナップショット(ストリーム全体を受信する場合はファイルシステムも)の名前は、引数タイプおよび"-d"オプションによって決まります。

引数がスナップショット名の場合には、指定した *snapshot* が作成されます。引数がファイルシステム名またはボリューム名の場合には、送信されたスナップショットと同名のスナップショットが、指定した *filesystem* または *volume* 内部で作成されます。"-d"オプションを指定すると、指定した *filesystem* に送信したスナップショットの名前を追加してスナップショット名が決定されます。また、指定したファイルシステム内のすべての必須ファイルシステムが作成されます。

-d	前の段階で説明したように、送信したスナップショットの名前を使って、新規スナップショットの名前が決定されます。
-v	ストリームおよび受信操作の所要時間に関する詳細な情報を出力します。
-n	ストリームを実際には受信しません。これを"-v"オプションと組み合わせて使用すると、受信操作で使用する名前を決定するのに役に立ちます。
-F	受信操作を実行する前に、*filesystem* のロールバックを最新のスナップショットに強制します。

使用例 1：zfs receive コマンドの例を示します。
root@opensolaris:~# **zfs receive tpool/data3@snap3 < fromsnap2** ⏎
root@opensolaris:~#

21 | upgrade | 最新のバージョンではないファイルシステムを表示します。

zfs upgrade [-v]

最新のバージョンではない ZFS ファイルシステムのリストを表示します。
-v 現在ソフトウェアでサポートされている ZFS バージョンを表示します。

使用例 1：zfs upgrade コマンドの例を示します。
root@opensolaris:~# **zfs upgrade** ⏎
root@opensolaris:~# zfs upgrade
This system is currently running ZFS filesystem version 3.

The following filesystems are out of date, and can be upgraded. After being upgraded, these filesystems (and any 'zfs send' streams generated from subsequent snapshots) will no longer be accessible by older software versions.

VERFILESYSTEM
------ ----------------------
2 tpool
root@opensolaris:~#
使用例 2：zfs upgrade -v コマンドの例を示します。
root@opensolaris:~# **zfs upgrade -v** ⏎
The following filesystem versions are supported:

VERDESCRIPTION
--- --
1 Initial ZFS filesystem version
2 Enhanced directory entries
3 Case insensitive and File system unique identifer (FUID)

For more information on a particular version, including supported releases, see:

http://www.opensolaris.org/os/community/zfs/version/zpl/N

Where 'N' is the version number.
root@opensolaris:~#

22 | upgrade | すべてのプール、または指定したプールを最新のオンディスクバージョンにアップグレードします。

zfs upgrade [-r] [-V *version*] -a | *filesystem*

一度アップグレードを行うと、以前のバージョンのソフトウェアを実行するシステム上で、ファイルシステムにアクセスすることはできなくなります。アップグレードされたファイルシステム、およびこれらのアップグレードされたファイルシステムから“zfs send”コマンドによって作成されたストリームには、古いソフトウェアリリースを実行しているシステムからはアクセスできません。
プールまたはファイルシステムのどちらかのバージョンが更新されている場合、upgrade を実行する必要はありません。
-a すべてのインポートされているプールのファイルシステムをアップグレードします。
filesystem 指定したファイルシステムをアップグレードします。
-r 指定したファイルシステムと、すべての親ファイルシステムをアップグレードします。
-V *version* 指定したバージョンにアップグレードします。“-V”オプションを指定しなかった場合、一番最近のバージョンにアップグレードされます。

使用例 1：zfs upgrade コマンドの例を示します。
root@opensolaris:~# **zfs upgrade -a** ⏎
1 filesystems upgraded
7 filesystems already at this version
root@opensolaris:~#

23 | allow | ファイルシステムの ZFS 管理アクセス権を別のアクセス権を持っていないユーザーに付与します。

zfs allow [-ldug] "*everyone*"|*user*|*group*[,...] *perm*|*@setname*[,...] *filesystem*|*volume*

zfs allow [-ld] -e *perm*|*@setname*[,...] *filesystem*|*volume*

[-ug] "*everyone*"|*user*|*group*[,...]	アクセス権を付与するユーザーを指定します。複数のアクセス権をコンマ区切りのリストとして指定できます。“-ug”オプションを指定しなかった場合、引数は優先的に“everyone”、次にユーザー名、最後にグループ名として解釈されます。“everyone”という名前のユーザーまたはグループを指定するには、“-u”または“-g”オプションを使用します。ユーザーと同じ名前のグループを指定するには、“-g”オプションを使用します。
[-e] *perm*|*@setname*[,...]	アクセス権がすべてのユーザーに付与されます。複数のアクセス権をコンマ区切りのリストとして指定できます。アクセス権の名前は、ZFS サブコマンドおよびプロパティと同じです。以下のプロパティリストを参照してください。プロパティセット名は、アットマーク記号(@)で始まるもので指定します。
[-ld] *filesystem*|*volume*	“-l”オプションは、アクセス権が指定のデータセットだけに許可されることを示します。“-d”オプションも指定されている場合を除き、子孫には許可されません。“-d”オプションは、アクセス権が子孫のデータセットだけに許可されることを示します。“-l”オプションも指定されている場合を除き、このデータセットには許可されません。“-ld”オプションのどちらも指定されていない場合は、ファイルシステムまたはボリュームおよびそのすべての子孫にアクセス権が許可されます。

“zfs allow”コマンドを使用して、ZFS データセットに対するアクセス権を root ユーザー以外のユーザーに次の方法で付与できます。
個別のアクセス権をユーザー、グループ、または全員に付与することができます。
個別のアクセス権の集まりを「アクセス権セット」としてユーザー、グループ、または全員に付与することができます。
アクセス権は、現在のデータセットだけにローカルで付与するか、現在のデータセットのすべての子孫に付与することができます。
次の表では、委任できる操作と、委任された操作の実行に必要な依存するアクセス権について説明します。

アクセス権 (サブコマンド)	説明	依存関係
allow	所有しているアクセス権を別のユーザーに付与する機能	許可しようとしているアクセス権自体を持っていることも必要です。
clone	データセットのスナップショットのいずれかを複製する機能	元のファイルシステムで create 機能と mount 機能を持っていることも必要です。
create	子孫のデータセットを作成する機能	mount 機能を持っていることも必要です。
destroy	データセットを破棄する機能	mount 機能を持っていることも必要です。
mount	データセットのマウントとマウント解除、およびボリュームのデバイスリンクの作成と破棄を行う機能	
promote	クローンをデータセットに昇格させる機能	元のファイルシステムで mount 機能と promote 機能を持っていることも必要です。
receive	“zfs receive”コマンドで子孫のファイルシステムを作成する機能	mount 機能と create 機能を持っていることも必要です。
rename	データセットの名前を変更する機能	新しい親で create 機能と mount 機能を持っていることも必要です。
rollback	スナップショットをロールバックする機能	mount 機能を持っていることも必要です。
send	スナップショットストリームを送信する機能	
share	データセットを共有および共有解除する機能	
snapshot	データセットのスナップショットを作成する機能	mount 機能を持っていることも必要です。

次の一連のアクセス権を委任できますが、アクセス、読み取り、および変更のアクセス権に限定されることがあります。
groupquota
groupused
userprop
userquota
userused
また、次の ZFS プロパティをルート以外のユーザーに委任できます。
aclinherit
aclmode
atime
canmount
checksum
compression
copies

	devices exec mountpoint primarycache quota readonly recordsize reservation secondarycache setuid sharenfs snapdir version volsize xattr zoned
	使用例 1：zfs allow コマンドの例を示します。 root@opensolaris:~# **zfs allow taro create,destroy,mount,snapshot tpool** ⏎ root@opensolaris:~# **使用例 2**：zfs allow -e コマンドの例を示します。 root@opensolaris:~# **zfs allow -e @eng tpool** ⏎ root@opensolaris:~#
24	allow　作成時のアクセス権を付与します。
	zfs allow -c *perm*|*@setname*[,...] *filesystem*|*volume*
	新たに親ファイルシステムを作成したユーザーにアクセス権を付与します。 -c　作成時のアクセス権を付与します。
	使用例 1：zfs allow -c コマンドの例を示します。 root@opensolaris:~# **zfs allow -c @eng,mountpoint tpool** ⏎ root@opensolaris:~#
25	allow　アクセス権を「アクセス権セット」にまとめて、指定します。
	zfs allow -s *@setname perm*|*@setname*[,...] *filesystem*|*volume*
	アクセス権を「アクセス権セット」にまとめて、"-s"オプションで指定できます。アクセス権セットは、指定のファイルシステムとその子孫に対してほかの"zfs allow"コマンドで使用できます。アクセス権セットは動的に評価されるため、セットに加えられた変更はすぐに更新されます。アクセス権セットは ZFS ファイルシステムと同じ命名規則に従いますが、名前はアットマーク記号(@)で始まり、64 文字以下の長さでなければなりません。
	使用例 1：zfs allow -s コマンドの例を示します。 root@opensolaris:~# **zfs allow -s @simple create,mount tpool** ⏎ root@opensolaris:~#
26	unallow　委任したアクセス権を削除します。
	unallow [-rldug] "*everyone*"|*user*|*group*[,...] [*perm*|*@setname*[,...]] *filesystem*|*volume* unallow [-rld] -e [*perm*|*@setname*[,...]] *filesystem*|*volume* unallow [-r] -c [*perm*|*@setname*[...]] *filesystem*|*volume*
	"zfs allow"コマンドによって委任されたアクセス権を削除します。アクセス権は明確に記述されなければ削除することはできません。いかなるアクセス権も、明確に指定されなかった場合、すべてのユーザー、グループまたはすべてのユーザーのためのアクセス権は削除されません。"everyone"によって委任されたユーザー、グループまたはすべてのユーザーのアクセス権は、"everyone"で削除してください。 -r　ファイルシステムとすべての親ファイルシステムのアクセス権を再帰的に削除します。
	使用例 1：zfs unallow コマンドの例を示します。 root@opensolaris:~# **zfs unallow taro tpool** ⏎ root@opensolaris:~# **使用例 2**：zfs unallow -e コマンドの例を示します。 root@opensolaris:~# **zfs unallow -r -e @create tpool** ⏎ root@opensolaris:~# **使用例 3**：zfs unallow -c コマンドの例を示します。 root@opensolaris:~# **zfs unallow -c tpool** ⏎ root@opensolaris:~#

27	unallow	アクセス権セットのアクセス権を削除します。
	unallow [-r] -s *@setname* [*perm*|*@setname*[,...]] *filesystem*|*volume*	
	いかなるアクセス権も、明確に指定されなかった場合、すべてのアクセス権、およびアクセス権セットは削除されません。	
	使用例 1：zfs unallow -s コマンドの例を示します。 root@opensolaris:~# **zfs unallow -s @destroy tpool** ⏎ root@opensolaris:~#	

A-3. beadm コマンド

OpenSolaris のルートファイルシステムの世代管理に“beadm”コマンドを使用します。

【備考】 Solaris 環境では“luxxx”コマンドに相当します。

次に beadm コマンドの概要を示します。No 欄の番号は、表 A-3-2 の参照番号です。

表 A-3-1. beadm コマンド概要

No	コマンド	概要
1	beadm	ヘルプメッセージを表示します。
2	activate *beName*	次回のリブート時に *beName* をアクティブなブート環境にします。
3	beadm create [-a] [-d *description*] [-e *non-activeBeName* | *beName@snapshot*] [-o property=value]... [-p zpool] *beName*	新しいブート環境名 *beName* を作成します。
4	beadm create *beName@snapshot*	*beName* という既存のブート環境のスナップショットを作成します。
5	beadm destroy [-fF] *beName* | *beName@snapshot*	*beName* というブート環境を破棄するか、ブート環境の既存のスナップショット *beName@snapshot* を破棄します。
6	beadm list [[-a] | [-d] [-s]] [-H] [*beName*]	*beName* という既存のブート環境に関する情報を表示します。
7	beadm mount *beName mountpoint*	マウントポイントで *beName* というブート環境をマウントします。
8	beadm rename *beName newBeName*	ブート環境の名前を変更します。
9	beadm unmount [-f] *beName*	*beName* というブート環境をマウント解除します。

表 A-3-2. beadm コマンドリファレンス

No	コマンド名	概要
	コマンド形式	
	解説	
	使用例	
1	beadm	ヘルプメッセージを表示します。
	beadm	
	beadm コマンドのキーワード、および使用方法について、簡潔な説明を表示します。	
	使用例 1：beadm コマンドの例を示します。 root@opensolaris:~# **beadm** ⏎ Usage: beadm subcommand cmd_options subcommands: beadm activate beName beadm create [-a] [-d description] [-e non-activeBeName | beName@snapshot] [-o property=value] ... [-p zpool] beName beadm create beName@snapshot beadm destroy [-fF] beName | beName@snapshot beadm list [[-a] | [-d] [-s]] [-H] [beName] beadm mount beName mountpoint beadm rename beName newBeName beadm unmount [-f] beName root@opensolaris:~#	
2	activate	次回リブート時にアクティブなブート環境にします。
	activate *beName*	
	次回のリブート時に *beName* をアクティブなブート環境にします。	
	使用例 1：beadm activate コマンドの例を示します。 root@opensolaris:~# **beadm activate BE1** ⏎ root@opensolaris:~#	
3	create	新しいブート環境名 *beName* を作成します。
	beadm create [-a] [-d *description*] [-e *non-activeBeName* | *beName@snapshot*] [-o property=value] ... [-p zpool] *beName*	
	“-e”オプションを指定しないかぎり、新しいブート環境は現在実行中のブート環境のクローンとして作成されます。 -a　作成と同時に、新しく作成されたブート環境をアクティブにします。デフォルトでは、新しく作成されたブート環境をアクティブにしません。 -d *description*　この内容は、x86 用 GRUB メニューまたは SPARC 用ブートメニューに表示する、この新しいブート環境のタイトルに使用します。このオプションが指定されていない場合は、*beName* がタイトルとして使用されます。 -e *non-activateBeName*　既存のアクティブでないブート環境から新しいブート環境を作成します。デフォルトでは、アクティブなブート環境からブート環境を作成します。 -e *beName@snapshot*　*beName* というブート環境の既存のスナップショットから新しいブート環境を作成します。 -o property=value　特定の ZFS プロパティを使って新しいブート環境のデータセットを作成します。複数の“-o”オプションを指定できます。 *beName*　作成するブート環境の名前。	
	使用例 1：beadm create コマンドの例を示します。 root@opensolaris:~# **beadm create -a BE1** ⏎ root@opensolaris:~#	

4	create	*beName* という既存のブート環境のスナップショットを作成します。

beadm create *beName@snapshot*

beName@snapshot	スナップショット名には、*beName@snapshotdescription* の形式を使用する必要があります。この場合、*beName* はスナップショットの作成対象となる既存のブート環境の名前です。カスタム *snapshotdescription* を入力して、スナップショットの日付または目的を特定します。

使用例 1：beadm create コマンドの例を示します。

```
root@opensolaris:~# beadm create BE1@snap ⏎
root@opensolaris:~#
```

5	destroy	*beName* というブート環境を破棄するか、 ブート環境の既存のスナップショット *beName@snapshot* を破棄します。

beadm destroy [-fF] *beName* | *beName@snapshot*

-f	ブート環境がマウントされている場合でも強制的に破棄します。
-F	確認を求めずにブート環境を強制的に破棄します。

使用例 1：beadm destroy コマンドの例を示します。

```
root@opensolaris:~# beadm destroy BE1 ⏎
Are you sure you want to destroy BE1? This action cannot be undone(y/[n]): y ⏎
root@opensolaris:~#
```

6	list	*beName* という既存のブート環境に関する情報を表示します。

beadm list [[-a] | [-d] [-s]] [-H] [*beName*]

-a	ブート環境に関する利用可能なすべての情報を表示します。この情報には、従属データセットおよびスナップショットが含まれます。
-d	ブート環境に属しているすべての従属データに関する情報を表示します。
-s	ブート環境のスナップショットに関する情報を表示します。
-H	ヘッダー情報を表示しないようにします。出力の各フィールドは、セミコロンで区切られます。 ヘッダー情報のない表示例を次に示します。 BE2:no:yes:mounted:/pool1/BE/BE2:6.2G;;; ヘッダー情報がない場合は、次の区切り文字で表示情報が識別されます。 ;ブート環境、データセット、ゾーン、およびスナップショットを区切ります。 :ブート環境、データセット、ゾーン、およびスナップショットの属性を区切ります。 ,複数のデータセット、ゾーン、およびスナップショットを区切ります。 複数のブート環境は、復帰改行(空行)で区切ります。
beName	*beName* を指定しないでこのコマンドを実行すると、すべてのブート環境の情報が表示されます。

【備考】“-p”オプションは他のオプションと組み合わせることができます。

使用例 1：beadm list コマンドの例を示します。

```
root@opensolaris:~# beadm list -d ⏎
BE/Dataset    ActiveMountpointSpacePolicy Created
--------------    -------- -------------- -------- -------- -----------
BE1
    rpool/ROOT/BE1   --  42.0Kstatic 2009-10-23 11:38
opensolaris
    rpool/ROOT/opensolaris --  8.80M  static 2009-10-13 10:34
opensolaris-1
    rpool/ROOT/opensolaris-1 NR/  3.05Gstatic 2009-10-19 16:32
root@opensolaris:~#
```

7	mount	マウントポイントで *beName* というブート環境をマウントします。

beadm mount *beName mountpoint*

マウントポイントは、既存の空のディレクトリにする必要があります。
【注意】リブートする前にブート環境をマウント解除します。

使用例 1：beadm mount コマンドの例を示します。

```
root@opensolaris:~# beadm mount BE1 /be1 ⏎
root@opensolaris:~#
```

8	rename	ブート環境の名前を変更します。

	beadm rename *beName newBeName*	
	beName というブート環境の名前を *newBeName* に変更します。	
	使用例 1：beadm rename コマンドの例を示します。 root@opensolaris:~# **beadm rename BE1 NBE1** ⏎ root@opensolaris:~#	
9	unmount or umount	*beName* というブート環境をマウント解除します。
	beadm unmount [-f] *beName*	
	-f　　ブート環境が現在ビジー状態であっても強制的にマウント解除します。	
	使用例 1：beadm unmount コマンドの例を示します。 root@opensolaris:~# **beadm unmount -f BE1** ⏎ root@opensolaris:~#	

A-4. ZFS プロパティセット

最初のプロパティセットはデータセットに関する統計データを表示するものです。これらのプロパティは、設定不可であり、継承も行われません。ネイティブのプロパティは、特に注記がない限り、全てのデータセットタイプに適用されます。

表 A-4-1. プロパティセット概要

No	プロパティセット	概要
1	type	データセットのタイプ。
2	creation	このデータセットが作成された時刻。
3	used	このデータセットおよびそのすべての子孫が使用する容量を調べます。
4	available	データセットおよびその子すべてが使用可能な容量。
5	referenced	このデータセットでアクセス可能なデータ量。
6	compressratio	このデータセットに対して達成された圧縮比。
7	mounted	ファイルシステムの場合は、ファイルシステムが現在マウントされているかどうかを示します。
8	origin	ファイルシステムまたはボリュームのクローンを作成した場合は、クローンの作成元のスナップショット。
9	quota=*size* | *none*	データセットおよびその子孫が使用できる容量を制限します。
10	reservation=*size* | *none*	データセットおよびその子孫に保証される最小容量。
11	volsize=*size*	ボリュームの場合に、ボリュームの論理サイズを指定します。
12	volblocksize=*blocksize*	ボリュームの場合に、ボリュームのブロックサイズを指定します。
13	recordsize=*size*	ファイルシステムに格納するファイルの推奨ブロックサイズを指定します。
14	mountpoint=*path* | *none* | *legacy*	このファイルシステムで使用されるマウントポイントを制御します。
15	sharenfs=*on* | *off* | *opts*	ファイルシステムをＮＦＳ経由で共有するかどうか、および使用するオプションを制御します。
16	shareiscsi=*on* | *off*	“shareiscsi”は、“sharenfs”プロパティと同様に、ZFS ボリュームが iSCSI ターゲットとしてエクスポートされるかどうかを示します。
17	checksum=*on* | *off* | *fletcher2*, | *fletcher4* | *sha256*	データの完全性を検証するために使用するチェックサムを制御します。
18	compression=*on* | *off* | *lzjb*	このデータセットで使用される圧縮アルゴリズムを制御します。
19	atime=*on* | *off*	ファイルを読み取るときにファイルのアクセス時刻を更新するかどうかを制御します。
20	devices=*on* | *off*	このファイルシステムでデバイスノードを開くことができるかどうかを制御します。
21	exec=*on* | *off*	このファイルシステム内部から、プロセスが実行可能かどうかを制御します。
22	setuid=*on* | *off*	設定された UID ビットが、このシステムで順守されるかどうかを制御します。
23	readonly=*on* | *off*	このデータを変更できるかどうかを制御します。
24	zoned=*on* | *off*	データセットを非大域ゾーンから管理するかどうかを制御します。
25	snapdir=*hidden* | *visible*	ファイルシステムのルートで、“.zfs”ディレクトリを非表示にするか、表示するかを制御します。
26	aclmode=discard | groupmask | passthrough	chmod(2)の実行時の ACL の変更方法を制御します。
27	aclinherit=discard | noallow | secure | passthrough	ファイルとディレクトリが作成されるときに ACL エントリをどのように継承するかを制御します。
28	canmount=on | off	このプロパティが“off”に設定されている場合、ファイルシステムはマウ

		ントできず、“zfs mount -a”を実行しても無視されます。
29	xattr=on | off	このファイルシステムで拡張属性が有効かどうかを制御します。
30	iscsioptions	iSCSI ターゲットデーモンにより IQN などの持続的情報の格納に使用されます。

表 A-4-2. プロパティセット

No	プロパティセット	説明
1	type	データセットのタイプ。“filesystem”、“volume”、“snapshot”、または“clone”です。
2	creation	このデータセットが作成された時刻。
3	used	このデータセットおよびそのすべての子孫が使用する容量を調べます。この値は、このデータセットの割り当ておよび予約に基づいて計算されます。使用される領域にこのデータセットの予約は含まれませんが、子孫のデータセットがある場合はそれらの予約も考慮されます。データセットがその親から継承して使用する容量、およびこのデータセットが再帰的に破棄されるときに解放される容量は、使用済み領域および予約の中で大きな割合を占めます。 スナップショットを最初に作成したとき、それらの領域はスナップショットとファイルシステムの間で共有されます。それまでに作成したスナップショットと領域が共有されることもあります。ファイルシステムが変化していくにつれて、それまで共有されていた領域がスナップショット固有になり、スナップショットが使用する領域に計上されます。また、スナップショットを削除すると、他のスナップショットに固有の(および使用される)容量を増やすことができます。 使用している容量、使用できる容量、または参照する容量では、保留状態の変更は考慮されません。保留状態の変更は通常、数秒以内に計上されます。“fsync(3C)”や“0_SYNC”を使用してディスクへの変更をコミットしても、領域の使用状況の情報がすぐに更新されることが保証されているわけではありません。
4	available	データセットおよびその子すべてが使用可能な容量。プール内にその他のアクティビティーが存在しないものとして計算されます。容量はプール内で共有されるため、プールの物理サイズ、割り当て、予約、プール内の他のデータセットなどの様々な要因によって、利用できる容量が制限されることがあります。 このプロパティは、列名の短縮形“avail”を使用しても参照できます。
5	referenced	このデータセットでアクセス可能なデータ量。これは、プール内の他のデータセットと共有される場合も、共有されない場合もあります。スナップショットまたはクローンを作成したときには、それらの作成元のファイルシステムまたはスナップショットと同じ領域を最初は参照しています。内容が同じであるためです。 このプロパティは、列名の短縮形“refer”を使用しても参照できます。
6	compressratio	このデータセットに対して達成された圧縮比。乗数で表記されます。“zfs set compression=on *dataset*”を実行すると、圧縮を有効にできます。デフォルト値は“off”です。
7	mounted	ファイルシステムの場合は、ファイルシステムが現在マウントされているかどうかを示します。このプロパティの値は、“yes”または“no”になります。
8	origin	ファイルシステムまたはボリュームのクローンを作成した場合は、クローンの作成元のスナップショット。クローンが存在するかぎり、“-r”や“-f”オプションを使用しても、作成元は破棄できません。
以降のプロパティセットは、データセット間で容量を割り当てる方法を制御するものです。これらのプロパティは継承されませんが、子孫に影響を及ぼします。		
9	quota=*size* | *none*	データセットおよびその子孫が使用できる容量を制限します。このプロパティにより、使用される容量に対して強い制限値が設定されます。これには、子孫の消費する容量すべて(ファイルシステムとスナップショットを含む)が含まれます。割り当てが既に設定されているデータセットの子孫に割り当てを設定した場合は、祖先の割り当ては上書きされずに、制限が追加されます。 ボリュームには割り当てを設定できません。“volsize”プロパティが暗黙的な割り当てとして機能します。
10	reservation=*size* | *none*	データセットおよびその子孫に保証される最小容量。使用している容量がこの値を下回っているデータセットは、予約に指定された容量を使用していると見なされます。予約は、親データセットが使用する容量に計上されるので、親データセットの割り当てと予約を減らすことになります。 このプロパティは、列名の短縮形“reserve”を使用しても参照できます。

11	volsize=*size*	ボリュームの場合に、ボリュームの論理サイズを指定します。デフォルトでは、ボリュームを作成するときに、同じ容量の予約が設定されます。“volsize”への変更があった場合には、予約にも対応する変更が反映されます。“volsize”に設定可能な値は、“volblocksize”の倍数だけです。この値をゼロにすることはできません。 コンシューマの予期しない動作を防ぐため、予約はボリュームの論理サイズと等価に保たれます。予約を使用しない場合、ボリュームの使用方法によっては、ボリュームの容量が不足して未定義の動作またはデータ破壊が発生する可能性があります。このような影響は、ボリュームの使用中にボリュームサイズを変更した場合にも発生することがあります。特に、サイズを縮小した場合にはその可能性が高くなります。ボリュームサイズを調整するときは、特に注意するようにしてください。 推奨されてはいませんが、“zfs create -V”コマンドに“-s”オプションを指定するか、ボリュームの作成後に予約を変更することにより、「疎ボリューム」(「シンプロビジョニング」とも呼ばれる)を作成できます。「疎ボリューム」とは、予約がボリュームサイズよりも小さいボリュームのことです。このため、領域上のプールが小さい場合、疎ボリュームへの書き込みが ENOSPC で失敗する可能性があります。疎ボリュームの場合、“volsize”を変更しても予約には反映されません。
12	volblocksize=*blocksize*	ボリュームの場合に、ボリュームのブロックサイズを指定します。ボリュームが書き込まれたあとで、“blocksize”を変更することはできません。このため、このプロパティはボリュームの作成時に設定してください。ボリュームのデフォルト“blocksize”は、8KB です。512B～128KB の範囲で、任意の 2 の累乗を指定できます。 このプロパティは、列名の短縮形“volblock”を使用しても参照できます。
13	recordsize=*size*	ファイルシステムに格納するファイルの推奨ブロックサイズを指定します。このプロパティは、レコードサイズが固定されているファイルにアクセスするデータベースワークロードだけで使用するように設計されています。ZFS では、標準的なアクセスパターンに最適化された内部アルゴリズムに従って、ブロックサイズが自動的に調整されます。 作成されるファイルのサイズが非常に大きく、それらのファイルに様々なパターンの小さなブロック単位でアクセスするデータベースの場合には、このようなアルゴリズムが最適でないことがあります。 “recordsize”にデータベースレコードサイズ以上の値を設定すると、パフォーマンスが大きく向上することがあります。このプロパティを汎用目的のファイルシステムに使用することは、パフォーマンスが低下する可能性があるため、できるだけ避けてください。 指定するサイズは、512B～128KB の 2 の累乗にしてください。 ファイルシステムの“recordsize”を変更すると、その後に作成されたファイルだけが影響を受けます。既存のファイルに影響はありません。 このプロパティは、列名の短縮形“recsize”を使用しても参照できます。
14	mountpoint=*path* | *none* | *legacy*	このファイルシステムで使用されるマウントポイントを制御します。 ファイルシステムの“mountpoint”プロパティを変更すると、そのマウントポイントを継承するファイルシステムおよびそのすべての子がマウント解除されます。新しい値が“legacy”である場合、マウント解除された状態が継続します。それ以外のときは、プロパティの古い値が“legacy”または“none”だった場合、またはプロパティが変更される前にマウントされていた場合は、新しい場所で自動的に再マウントされます。また、共有されていたすべてのファイルシステムは、共有が解除されてから新しい場所で共有されます。
15	sharenfs=*on* | *off* | *opts*	ファイルシステムを NFS 経由で共有するかどうか、および使用するオプションを制御します。“sharenfs”プロパティが“off”であるファイルシステムは、“share(1M)”、“unshared(1M)”、“dfstab(4)”などの従来のツールを使って管理します。これらのツールを使って管理しない場合、ファイルシステムは“zfs share”および“zfs unshare”コマンドにより自動的に共有および共有解除されます。このプロパティを“on”に設定すると、“share(1M)”コマンドがオプションなしで呼び出されます。“on”に設定しない場合、“share(1M)”コマンドの呼び出し時に、このプロパティの内容と等価なオプションが使用されます。 データセットの“sharenfs”プロパティが変更されると、プロパティが以前に“off”に設定されていたか、変更前に共有されていた場合にのみ、そのデータセットお

		よびそのプロパティを継承するすべての子により新しいオプションが再度共有されます。新規プロパティが“off”の場合、ファイルシステムは共有を解除されます。
16	shareiscsi=*on* | *off*	“shareiscsi”は、“sharenfs”プロパティと同様に、ZFS ボリュームが iSCSI ターゲットとしてエクスポートされるかどうかを示します。このプロパティで使用可能な値は、“on”、“off”、および“type=disk”です。デフォルト値は“off”です。将来、“tape”など、その他のターゲットタイプもサポートされる可能性があります。 ファイルシステムに“shareiscsi=on”を設定して、そのファイルシステム内のすべての ZFS ボリュームがデフォルトで共有されるようにしたい場合があるかもしれません。しかし、ファイルシステム上でこのプロパティを設定しても、直接的な効果はありません。
17	checksum=*on* | *off* | *fletcher2*, | *fletcher4* | *sha256*	データの完全性を検証するために使用するチェックサムを制御します。デフォルト値“on”では、適切なアルゴリズムが自動的に選択されます。現在、アルゴリズムは *fletcher2* ですが、将来のリリースで変更される可能性があります。値が“off”の場合、ユーザーデータの完全性チェックが無効になります。チェックサムの無効化は、推奨されていない操作です。
18	compression=*on* | *off* | *lzjb*	このデータセットで使用される圧縮アルゴリズムを制御します。現在のところ、存在するアルゴリズムは“*lzjb*”だけですが、将来のリリースでは変更される可能性があります。デフォルト値は“off”です。 このプロパティは、列名の短縮形“compress”を使用しても参照できます。
19	atime=*on* | *off*	ファイルを読み取るときにファイルのアクセス時刻を更新するかどうかを制御します。このプロパティをオフに設定すると、ファイルを読み取るときに書き込みトラフィックが生成されなくなるため、パフォーマンスが大幅に向上する可能性があります。ただし、メールプログラムなどのユーティリティーが予期しない動作をすることがあります。デフォルト値は“on”です。
20	devices=*on* | *off*	このファイルシステムでデバイスノードを開くことができるかどうかを制御します。デフォルト値は“on”です。
21	exec=*on* | *off*	このファイルシステム内部から、プロセスが実行可能かどうかを制御します。デフォルト値は“on”です。
22	setuid=*on* | *off*	設定された UID ビットが、このシステムで順守されるかどうかを制御します。デフォルト値は“on”です。
23	readonly=*on* | *off*	このデータを変更できるかどうかを制御します。デフォルト値は“off”です。 このプロパティは、列名の短縮形“rdonly”を使用しても参照できます。
24	zoned=*on* | *off*	データセットを非大域ゾーンから管理するかどうかを制御します。デフォルトは“off”です。
25	snapdir=*hidden* | *visible*	ファイルシステムのルートで、“.zfs”ディレクトリを非表示にするか、表示するかを制御します。デフォルト値は“hidden”です。
26	aclmode=discard | groupmask | passthrough	“chmod(2)”の実行時の ACL の変更方法を制御します。“aclmode”プロパティが“discard”であるファイルシステムでは、ファイルのモードを表さない ACL エントリがすべて削除されます。“aclmode”プロパティの“groupmask”(デフォルト)は、ユーザーまたはグループのアクセス権を低下させます。アクセス権は、グループアクセス権ビットと同程度にまで低下します。ただし、アクセス権がファイルまたはディレクトリの所有者と同じ UID を持つユーザーエントリである場合を除きます。この場合、ACL アクセス権は、所有者のアクセス権ビットと同程度にまで引き下げられます。“aclmode”プロパティが“passthrough”であるファイルシステムでは、ファイルまたはディレクトリの新規モードを表す必須の ACL エントリを生成する以外、ACL に変更は加えられません。
27	aclinherit=discard | noallow | secure | passthrough	ファイルとディレクトリが作成されるときに ACL エントリをどのように継承するかを制御します。“aclinherit”プロパティが“discard”であるファイルシステムは、ACL エントリを一切継承しません。“aclinherit”プロパティが“noallow”であるファイルシステムは、“deny”アクセス権を指定する継承可能な ACL エントリだけを継承します。プロパティ値“secure”(デフォルト)は、ACL エントリ継承時に“write_acl”および“write_owner”アクセス権を削除します。“aclinherit”プロパティ値が“passthrough”であるファイルシステムは、継承時に ACL エントリに加えられた変更を除く、継承可能なすべての ACL エントリを継承します。
28	canmount=on | off	このプロパティが“off”に設定されている場合、ファイルシステムはマウントでき

		ず、“zfs mount -a”を実行しても無視されます。これは、“mountpoint”プロパティを“none”に設定することに似ていますが、継承可能な通常の“mountpoint”プロパティをデータセットが引き続き保持する点が異なります。これによってデータセットを、プロパティ継承の機構として使用できるようになります。1 つの使用例として、論理的に分けられた 2 つのデータセットが同じマウントポイントを保持するようにできます。これにより、両方のデータセットの子は、同じディレクトリ内に表示されますが、継承する特性は異なります。デフォルト値は“on”です。 このプロパティは継承されません。
29	xattr=on | off	このファイルシステムで拡張属性が有効かどうかを制御します。デフォルト値は“on”です。
30	iscsioptions	この読み取り専用の隠しプロパティは、iSCSI ターゲットデーモンにより IQN などの持続的情報の格納に使用されます。“zfs”コマンドを使って、これを表示または変更することはできません。この内容は、外部のコンシューマを対象としたものではありません。

A-4-1. 一時的なマウントポイントプロパティ

ファイルシステムのマウント時に、“mount(1M)”(従来のマウント)、または“zfs mount”コマンド(通常のファイルシステム)を使用して、プロパティに合わせたマウントオプションが設定されます。プロパティとマウントオプションは、次のような関係になっています。

PROPERTY	MOUNT OPTION
devices	devices/nodevices
exec	exec/noexec
readonly	ro/rw
setuid	setuid/nosetuid
xattr	xattr/noxattr

さらに、“-o”オプションを使って、ディスクに格納されたプロパティに影響を及ぼすことなく、これらのオプションをマウントごとに設定できます。コマンド行に指定した値により、データセットに格納された値が上書きされます。“-nosuid”オプションは、“nodevices,nosetuid”の別名です。これらのプロパティは、“zfs get”コマンドにより、“temporary”とレポートされます。データセットのマウント時にプロパティが変更されると、新規設定により一時設定がすべて上書きされます。

A-4-2. ユーザープロパティ

ZFS は、標準のネイティブプロパティに加えて、任意のユーザープロパティもサポートします。ユーザープロパティは ZFS の動作には影響を与えませんが、アプリケーションや管理者が使用して、データセットに注釈を付けることができます。

ユーザープロパティ名には、ネイティブプロパティと区別するためにコロン(“:”)文字を含める必要があります。ユーザープロパティに含めることができるのは、小文字の英字、数字、および句読文字のコロン(“:”)、ハイフン(“-”)、ピリオド(“.”)、および下線(“_”)です。想定されている規制では、プロパティ名は 2 つの部分に分割します(例:“*module:property*”)。ただし、この名前空間は ZFS によって強制されているものではありません。ユーザープロパティ名には、最大 256 文字を使用できます。名前の先頭にハイフン(“-”)を付けることはできません。

ユーザープロパティをプログラムで使用する場合、プロパティ名の *module* 要素には、逆順にした DNS ドメイン名を使用することを強くお勧めします。これは、それぞれ単独で開発された 2 つのパッケージが、異なる目的で同じプロパティ名を使用する可能性を減らすためです。“com.sun.”で始まるプロパティ名は、Sun Microsystems が使用するために予約されています。

ユーザープロパティの値は任意の文字列であり、常に継承されます。また、決して検証されることがありません。プロパティ上で動作するコマンド(“zfs list”、“zfs get”、“zfs set”など)はすべて、ネイティブプロパティとユーザープロパティの両方の操作に使用できます。ユーザープロパティをクリアするには、“zfs inherit”のコマンドを使用します。プロパティがどの親データセット内でも定義されていない場合は、そのプロパティ全体が削除されます。プロパティ値は、1024 文字以内に制限されています。

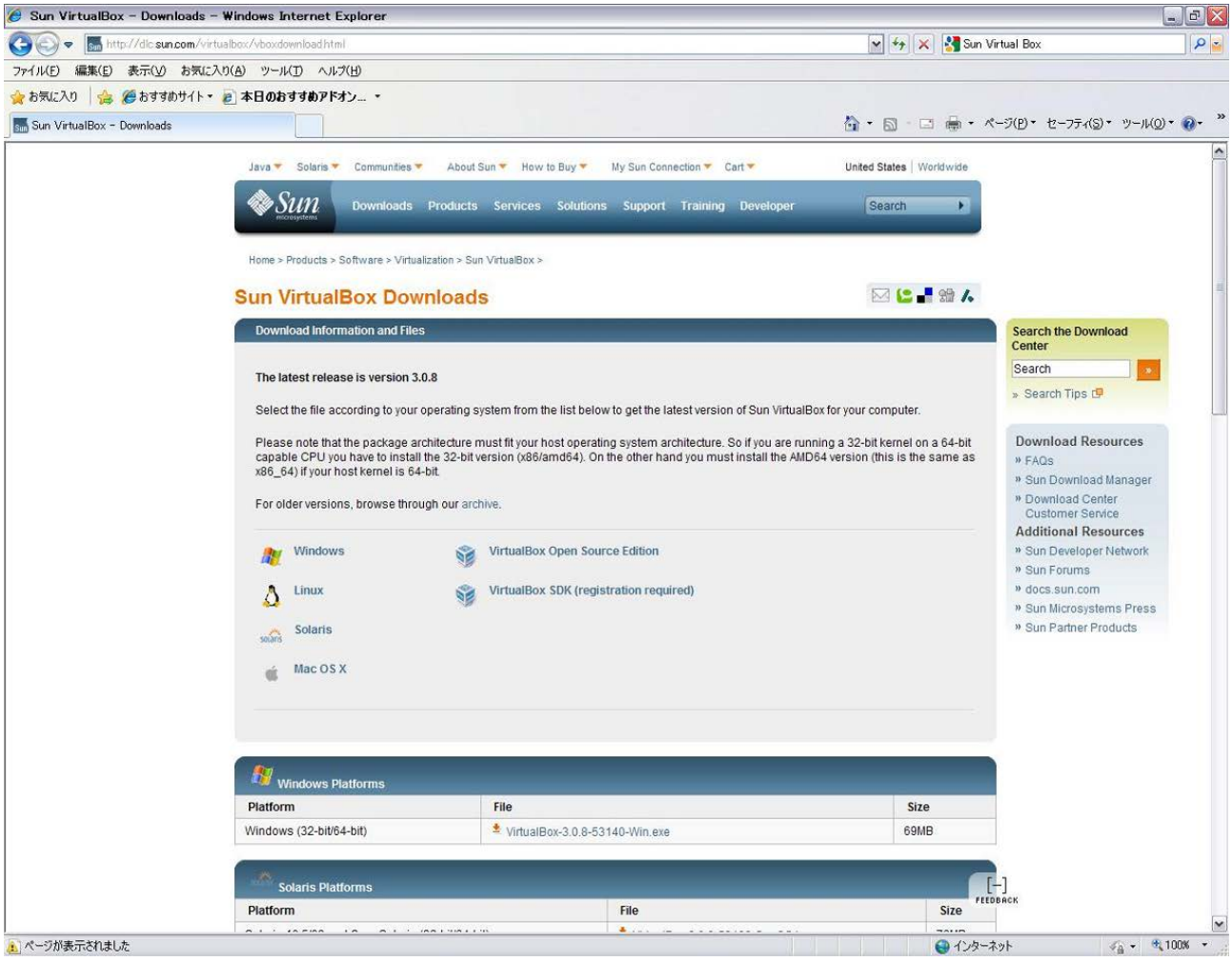
付録 B. ZFS を仮想マシンで使用する

ZFS を実際に構成する場合、サーバーやディスク装置を接続する必要があります。ここでは実ハードウェアを使用せずに仮想環境上で ZFS を操作する方法を紹介します。各種の構成テスト、ZFS のファンクションテストに有効です。

B-1. Sun xVM VirtualBox のダウンロードおよびインストール

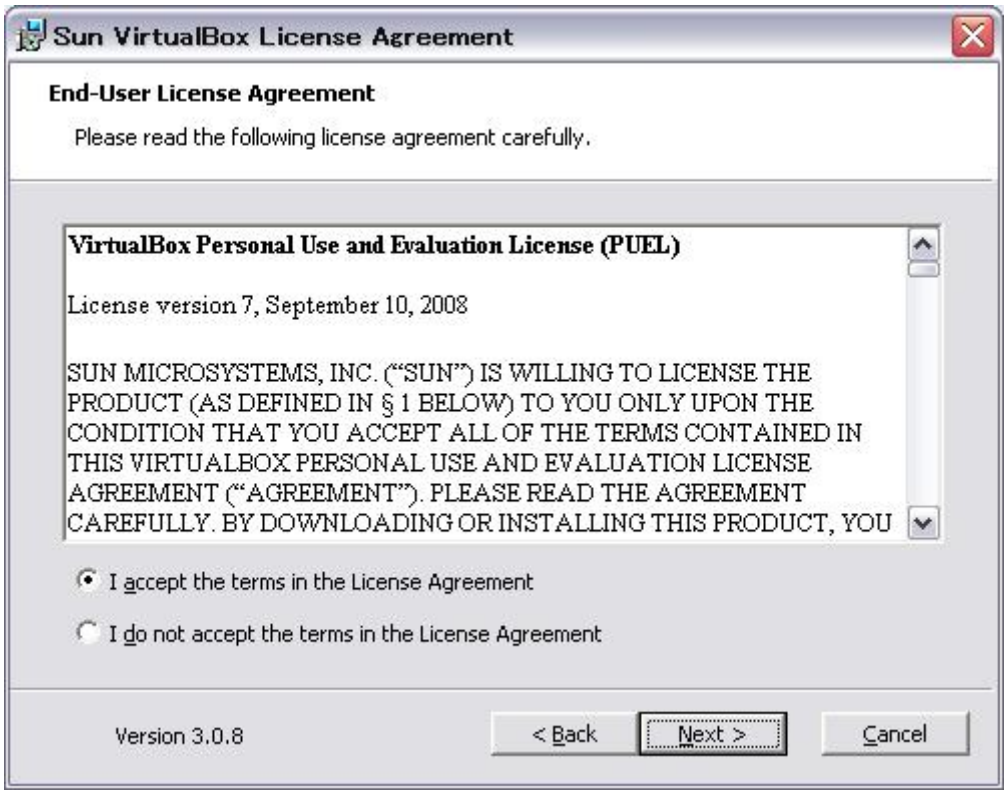
Windows に Sun xVM VirtualBox をインストールする手順を解説します。

(1) Sun Microsystems の“http://dlc.sun.com/virtualbox/vboxdownload.html#source”ページにあるファイルをダウンロードし、解凍してインストールします。

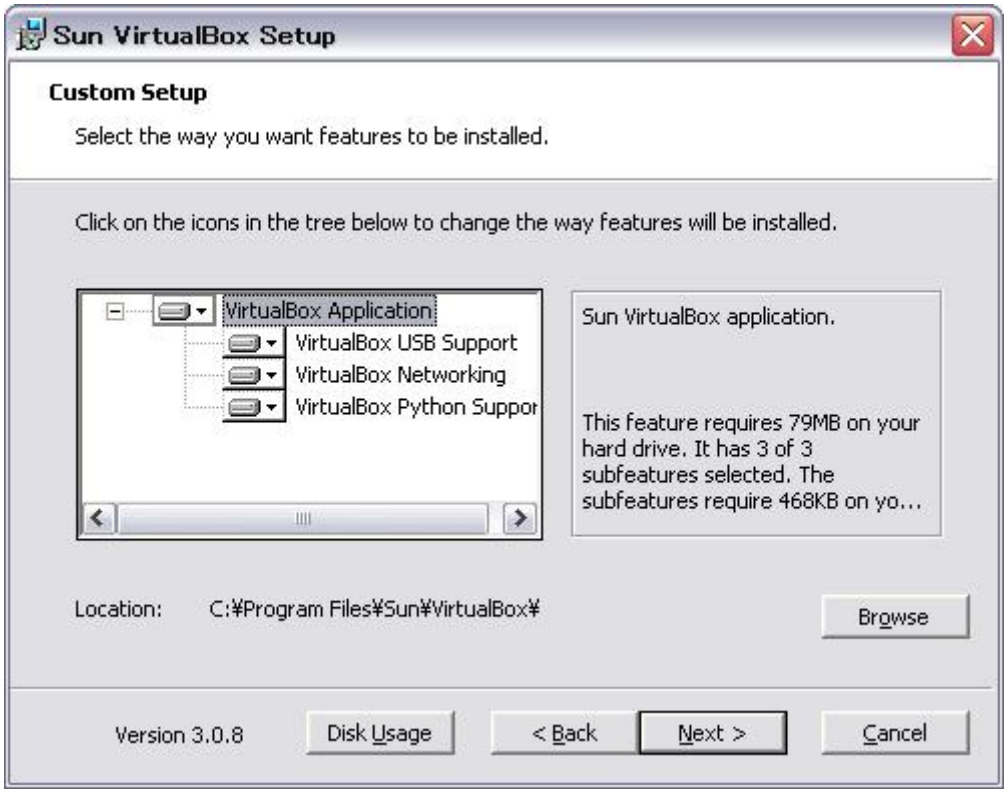


(2) Sun xVM VirtualBox Setup(インストーラー)を起動します。以下の図は起動画面です。“Next”をクリックします。

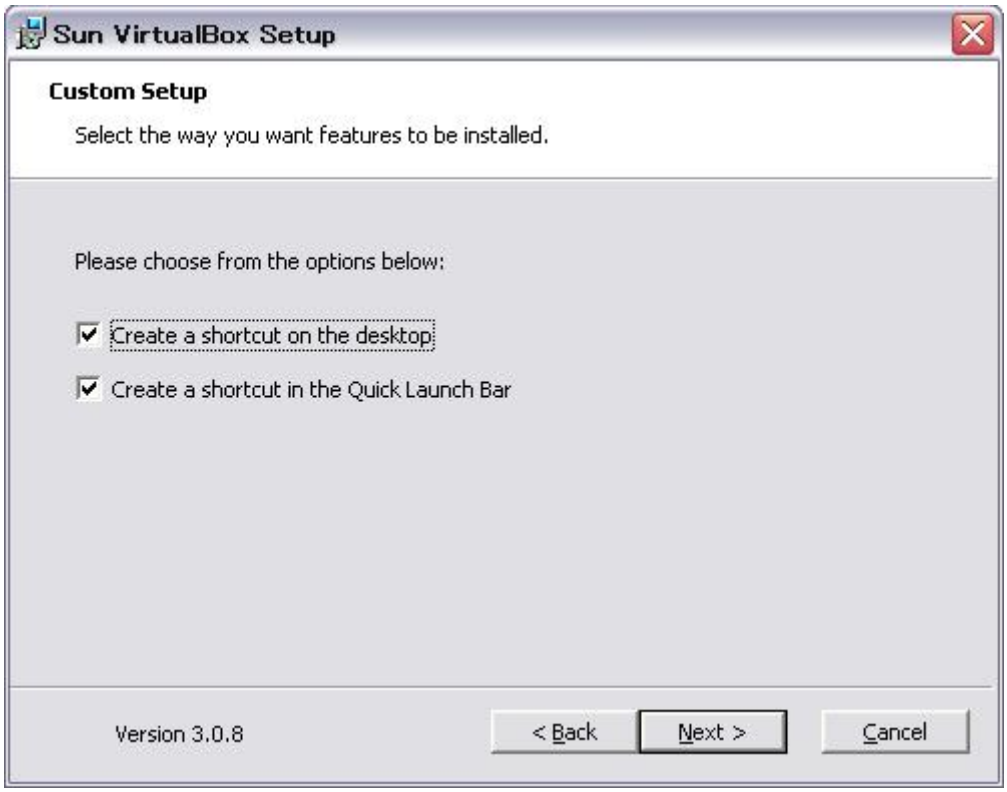
(3) ライセンスの同意画面です。“I accept the terms in the License Agreement”を選択して、“Next”をクリックします。



(4) セットアップ画面です。このまま“Next”をクリックします。

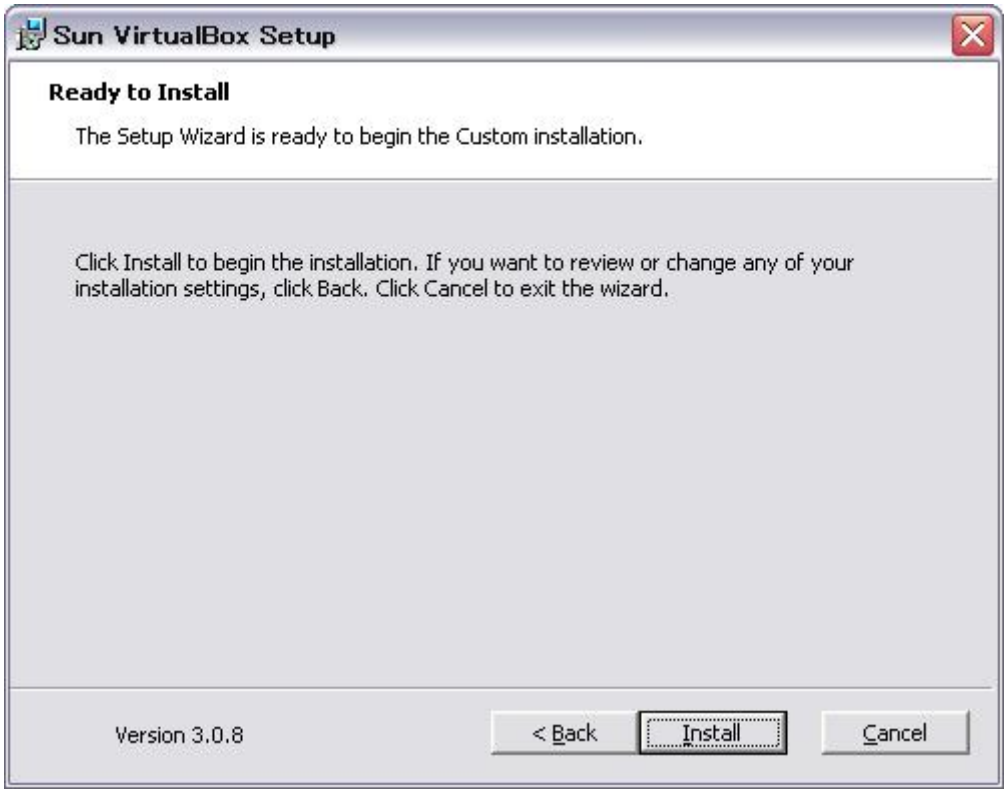


(5) セットアップ画面です。このまま“Next”をクリックします。

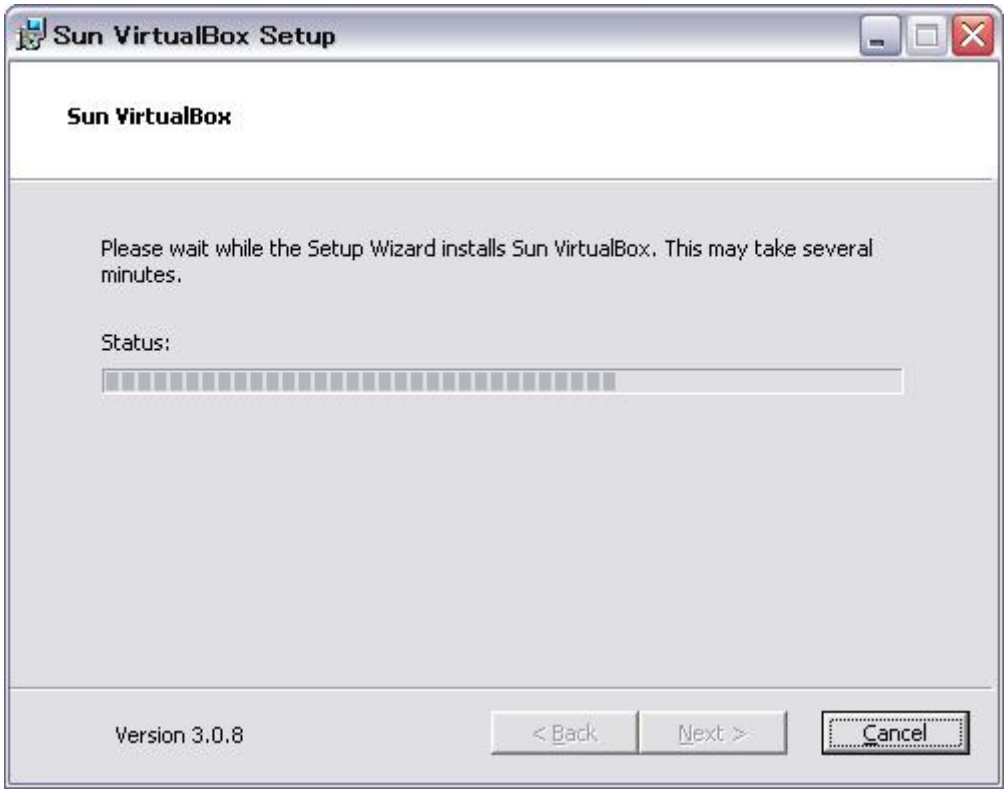


(6) 警告画面です。“Yes”をクリックします。

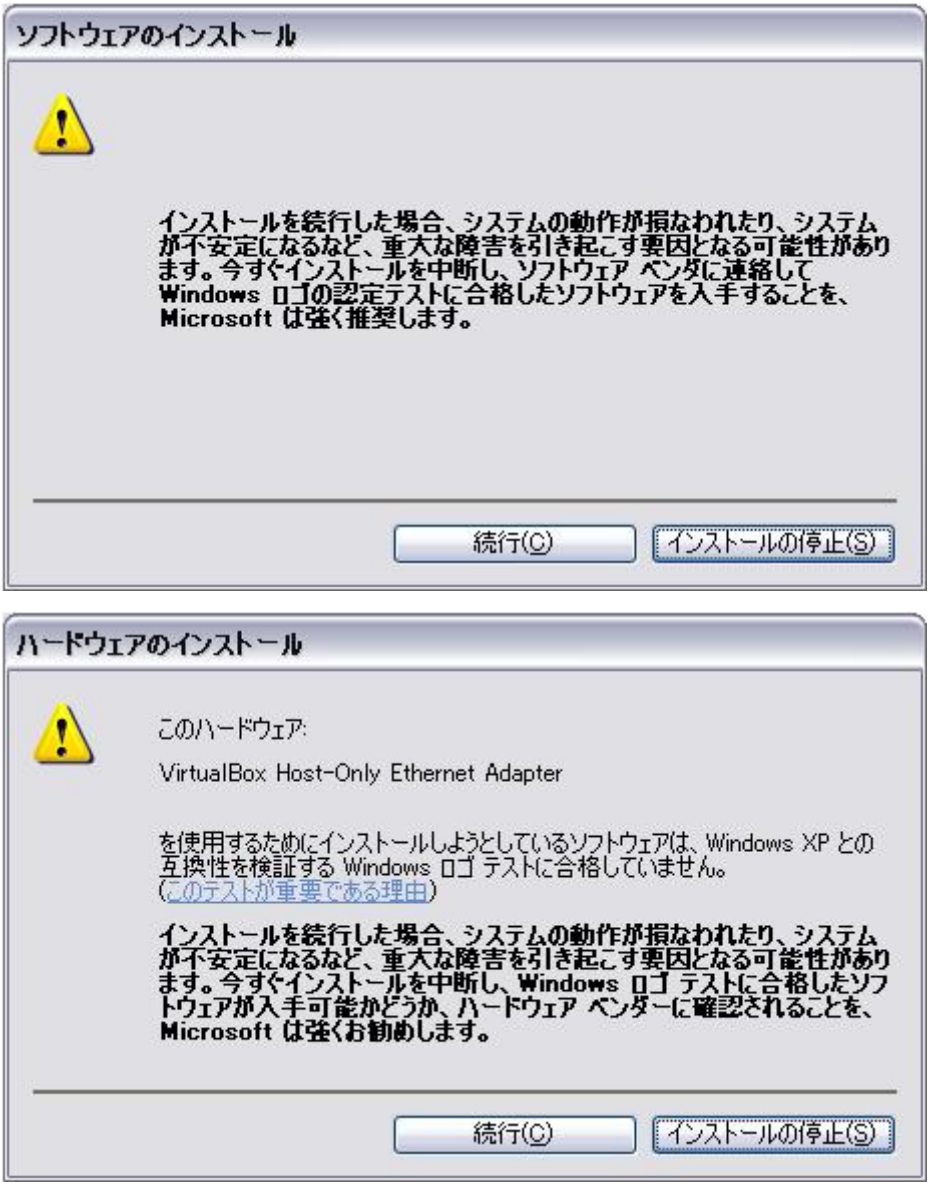
(7) インストールの開始確認画面です。“Install”をクリックします。



(8) インストールの進捗を示す画面です。



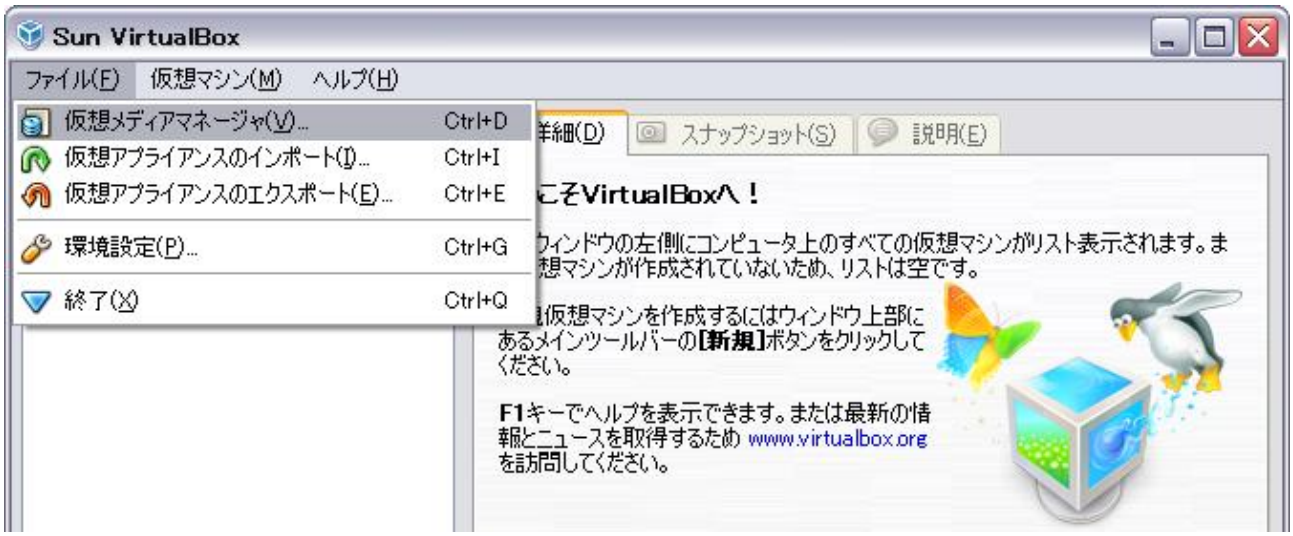
(9) 警告画面です。何度かこのような警告画面が表示されますが、“続行”をクリックします。



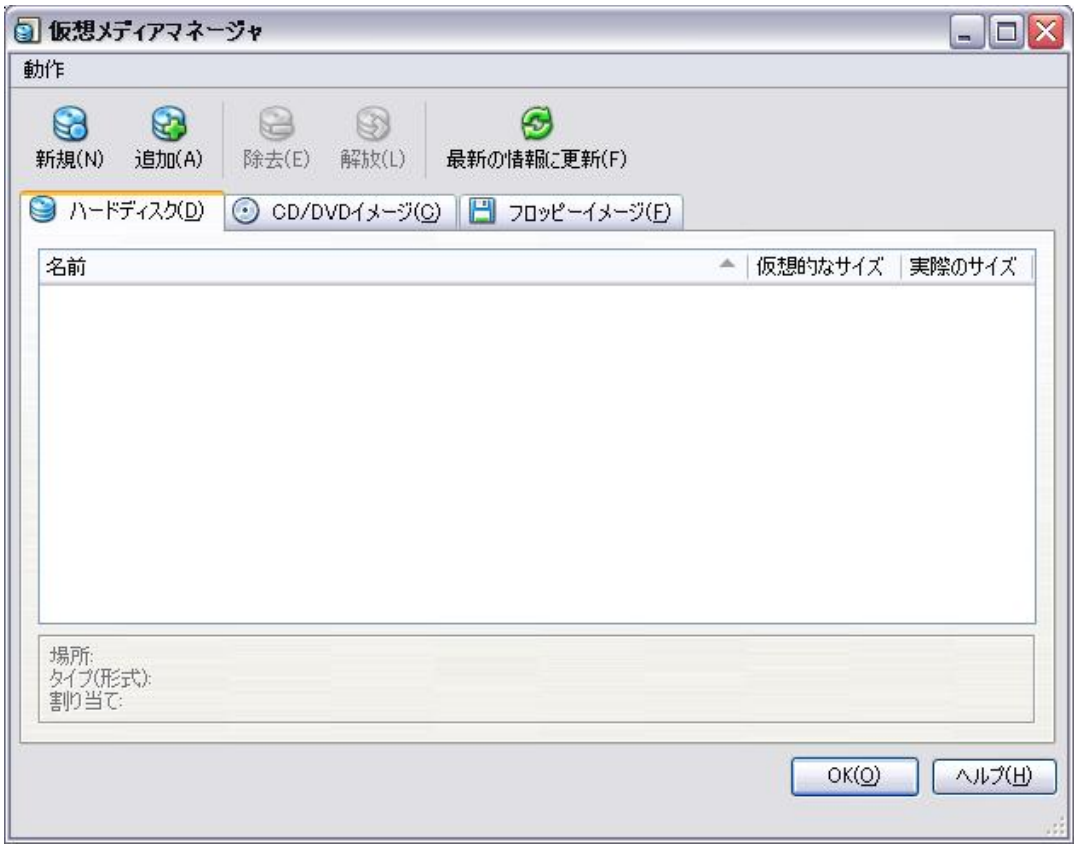
(10) インストールの完了画面です。“Finish”をクリックします。

B-2. 新規仮想ディスクの作成

(1) Sun xVM VirtualBox 起動画面です。新規仮想ディスクを作成するために、メニューから“ファイル”→“仮想メディアマネージャ”をクリックします。



(2) 仮想メディアマネージャの起動画面です。“新規”ボタンをクリックします。



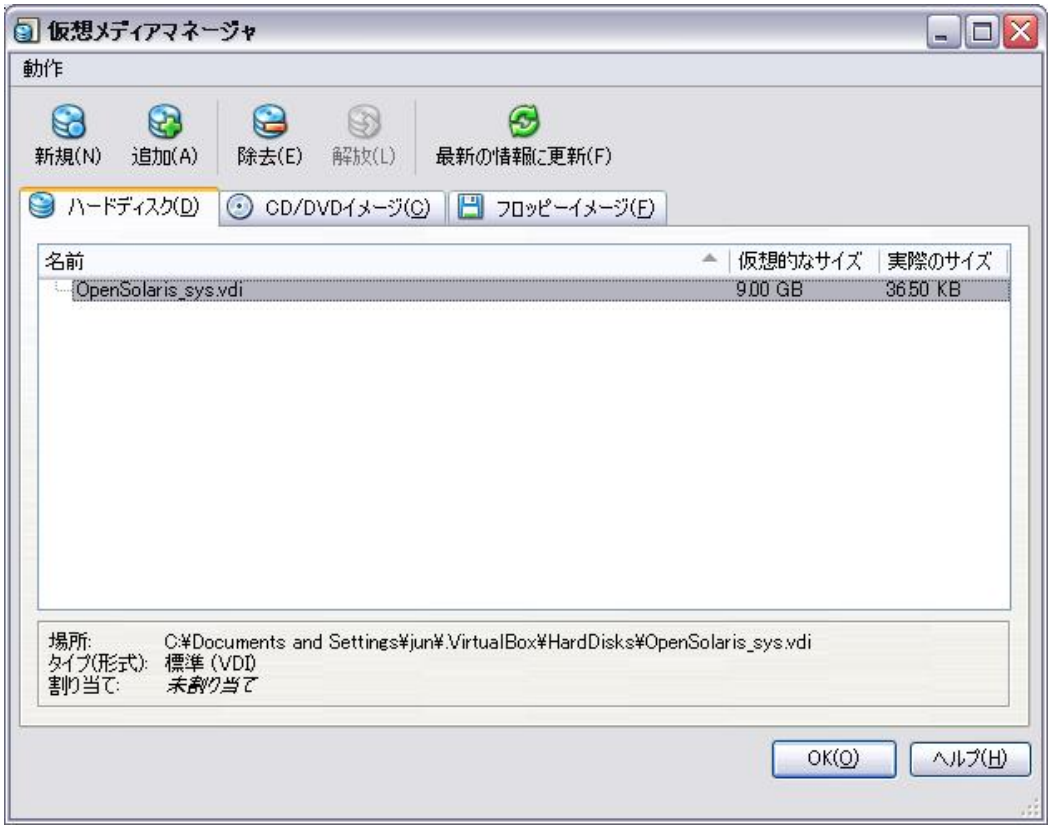
(3) 新規仮想ディスク作成ウィザード画面です。先ず、OS のインストール領域を作成します。“次へ”をクリックします。

(4) ハードディスクストレージタイプの選択画面です。“可変サイズのストレージ”を選択して、“次へ”をクリックします。

(5) 仮想ディスクの場所とサイズの設定画面です。ここでは、場所を“OpenSolaris_sys.vdi”、容量を“9.00GB”に設定しました。“次へ”をクリックします。

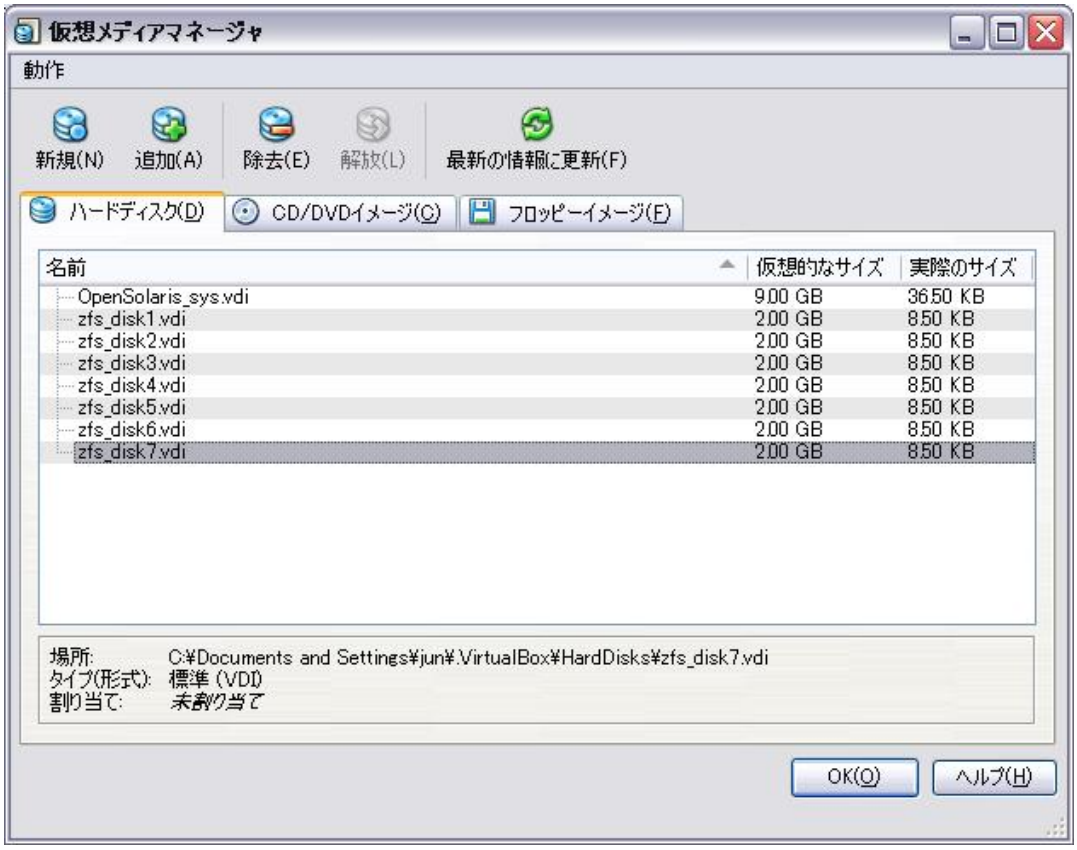
(6) ファイル名と容量の確認画面です。設定に問題がなければ、“完了”をクリックします。

(7) 仮想メディアマネージャ画面です。仮想ディスクの追加を確認します。



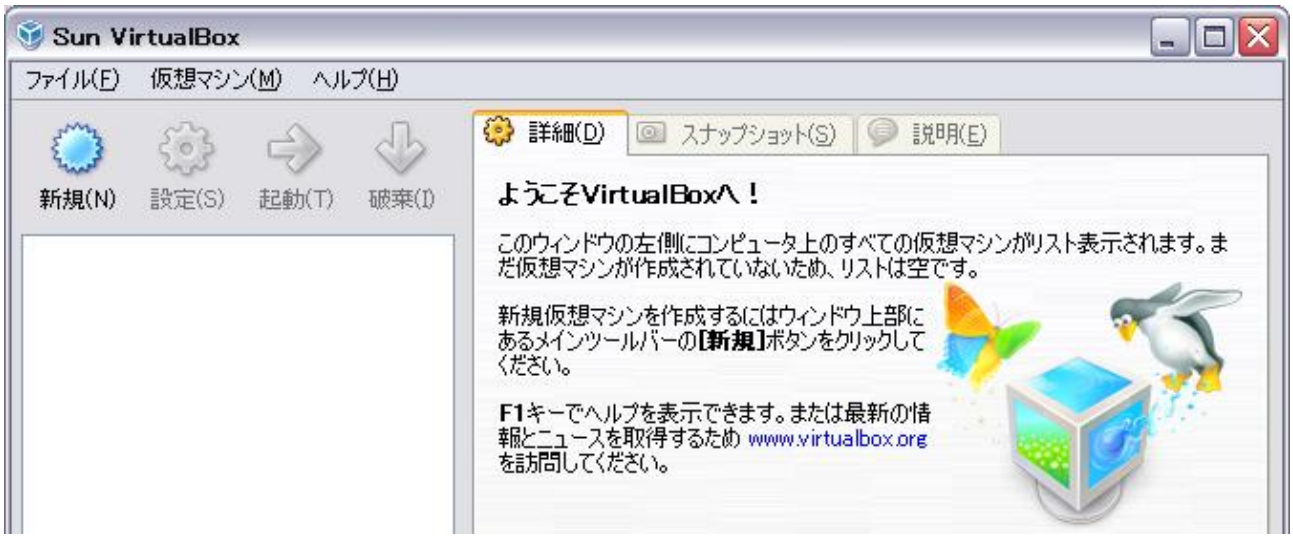
(8) 次に、ZFSテスト用の仮想ディスクを作成します。インストール用ディスクと同じ手順で作成します。ここでは、場所を"zfs_disk1.vdi"～"zfs_disk7.vdi"、容量をそれぞれ"2.00GB"に設定しました。

(9) 仮想メディアマネージャ画面です。全ての仮想ディスクが作成されたことを確認します。“OK”をクリックします。



B-3. 新規仮想マシンの作成

(1) Sun xVM VirtualBox 起動画面です。“新規”ボタンをクリックします。



(2) 新規仮想マシン作成ウィザード画面です。“次へ”をクリックします。

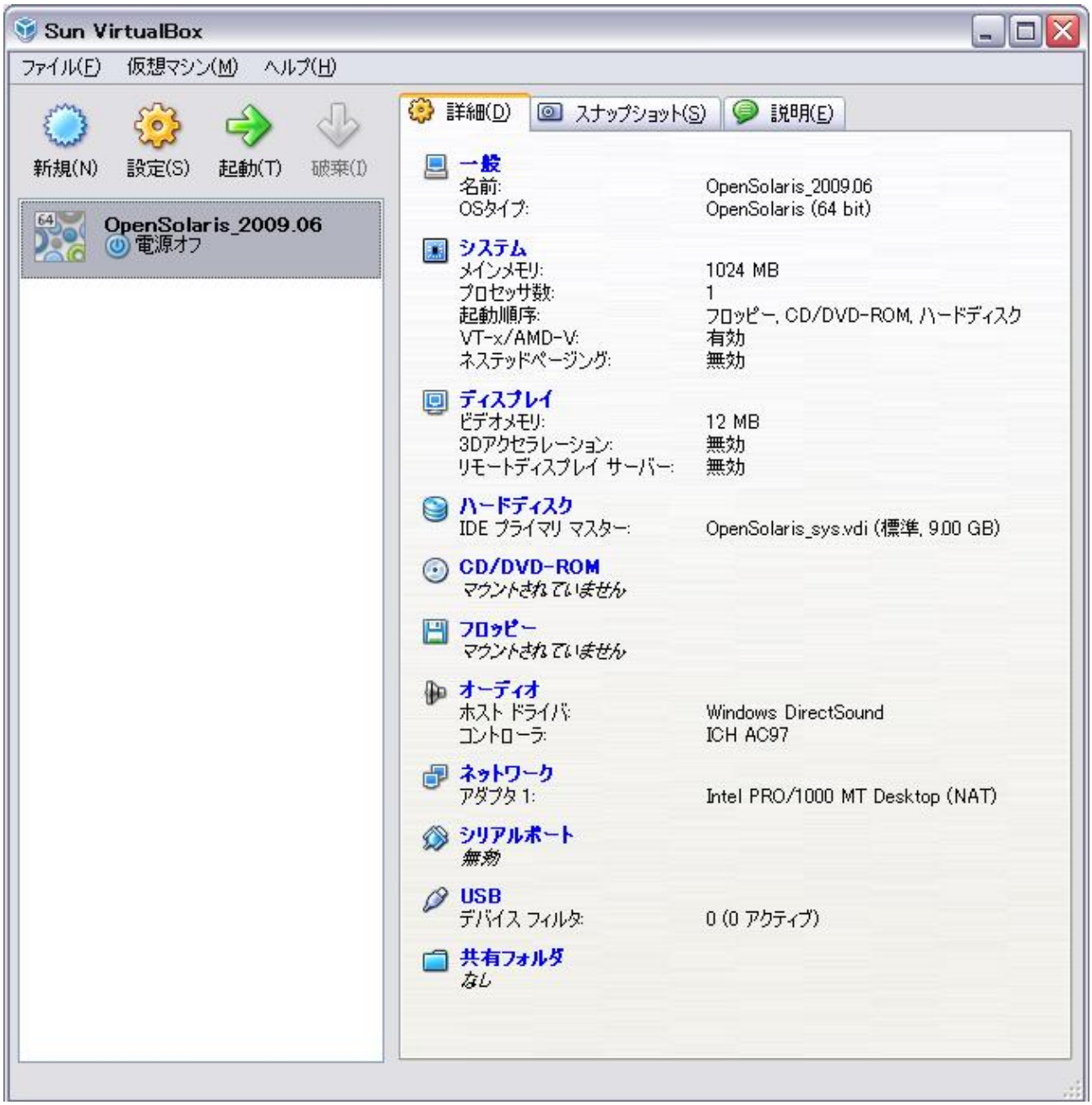
(3) 仮想マシン名と OS タイプの設定画面です。ここでは、名前を“OpenSolaris_2009.06”、OS タイプのオペレーティングシステムを“Solaris”、バージョンを“OpenSolaris (64 bit)”に設定しました。“次へ”をクリックします。

(4) メモリサイズの設定画面です。ここでは、“1024MB”を設定しました。“次へ”をクリックします。

(5) ブートディスク選択画面です。ブートディスク用に作成した、“OpenSolaris_sys.vdi”を選択して、“次へ”をクリックします。

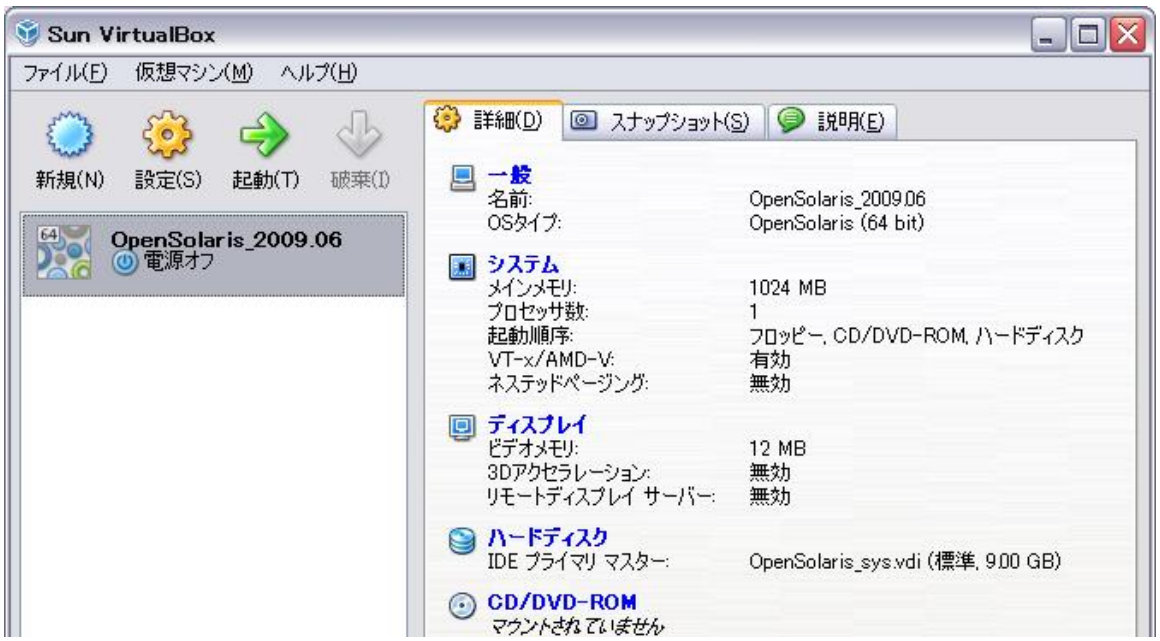
(6) 設定の確認画面です。設定に問題がなければ、“完了”をクリックします。

(7) Sun xVM VirtualBox 起動画面です。仮想マシンが追加されていることを確認します。



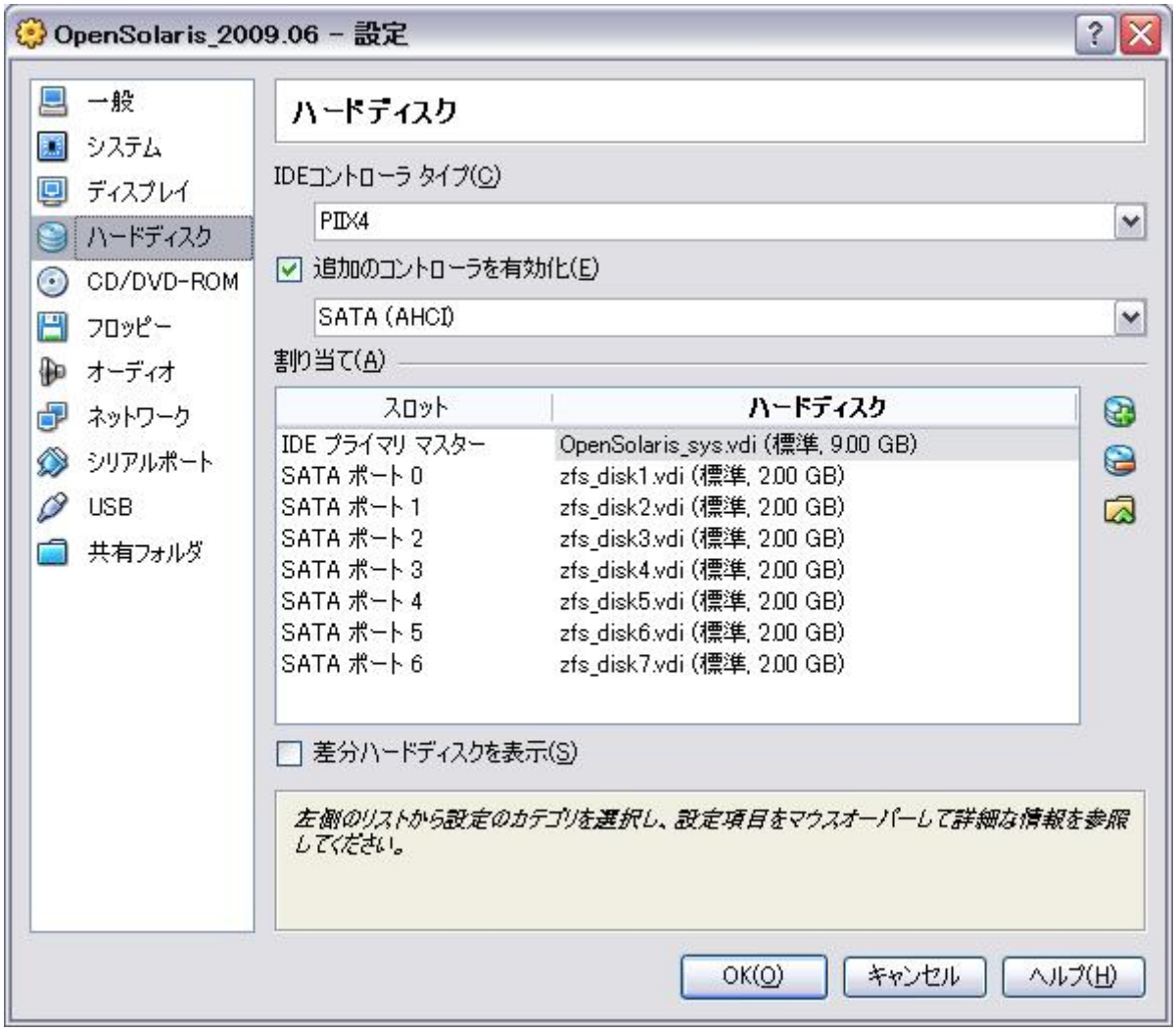
B-4. 仮想ディスクの追加とインストール CD イメージの登録

(1) Sun xVM VirtualBox 起動画面です。“設定”ボタンをクリックします。



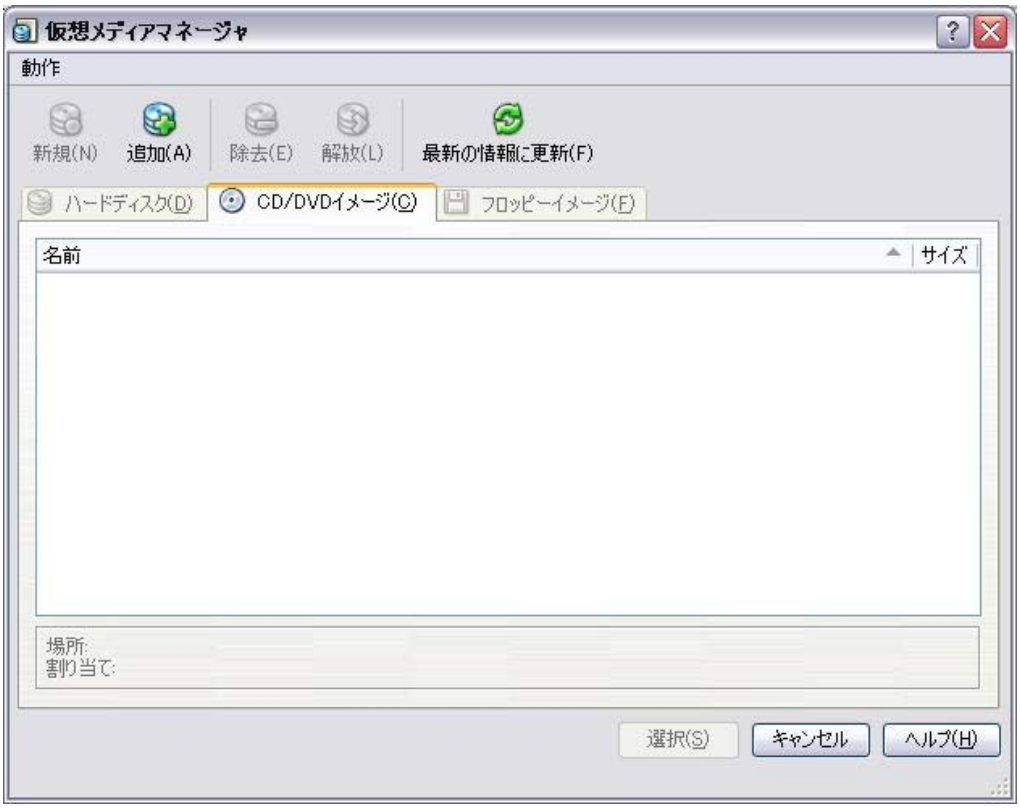
(2) ハードディスクの設定画面です。ZFS に使用するための仮想ディスクを追加します。左側メニュー欄の“ハードディスク”をクリックして設定を行います。

(3) “追加のコントローラを有効化”のチェックボタンをクリックして、有効にします。ボタンで作成した仮想ディスクを追加して、以下の画面のように設定します。

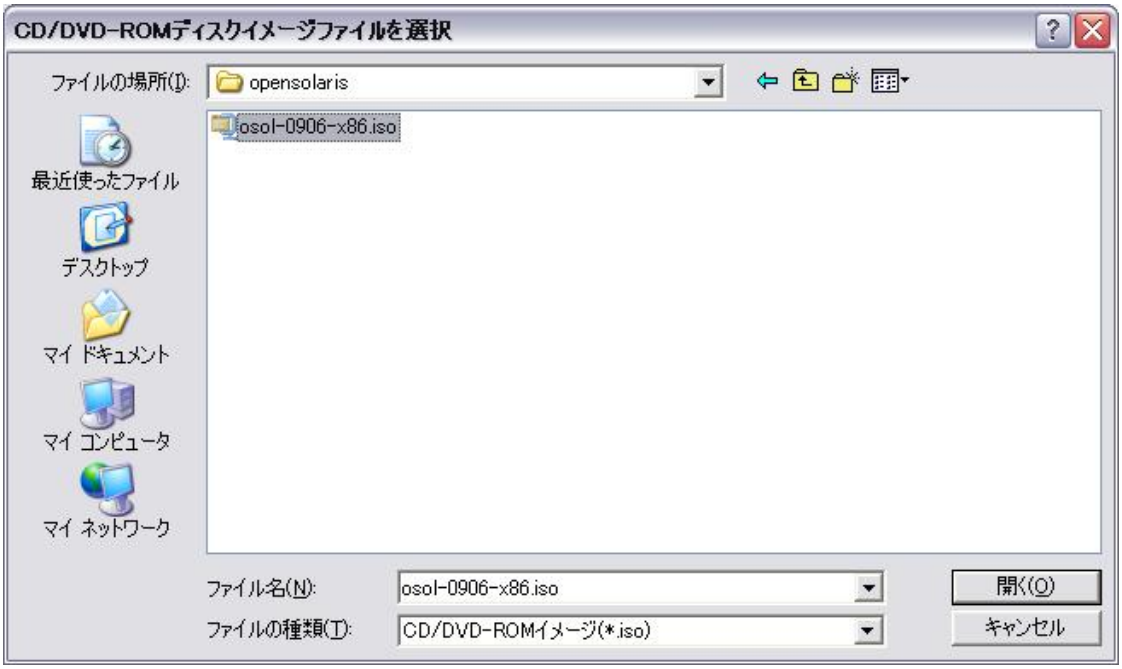


(4) CD/DVD-ROM の設定画面です。ここでは ISO イメージを利用します。opensolaris OS の ISO イメージファイルは、“http://www.opensolaris.com/get/index.jsp”ページにあるファイルをダウンロードしておきます。“CD/DVD-ROM ドライブのマウント”のチェックボタンをクリックして、“ISO イメージファイル”ボタンをクリックします。 ボタンをクリックして、仮想メディアマネージャ画面を開きます。

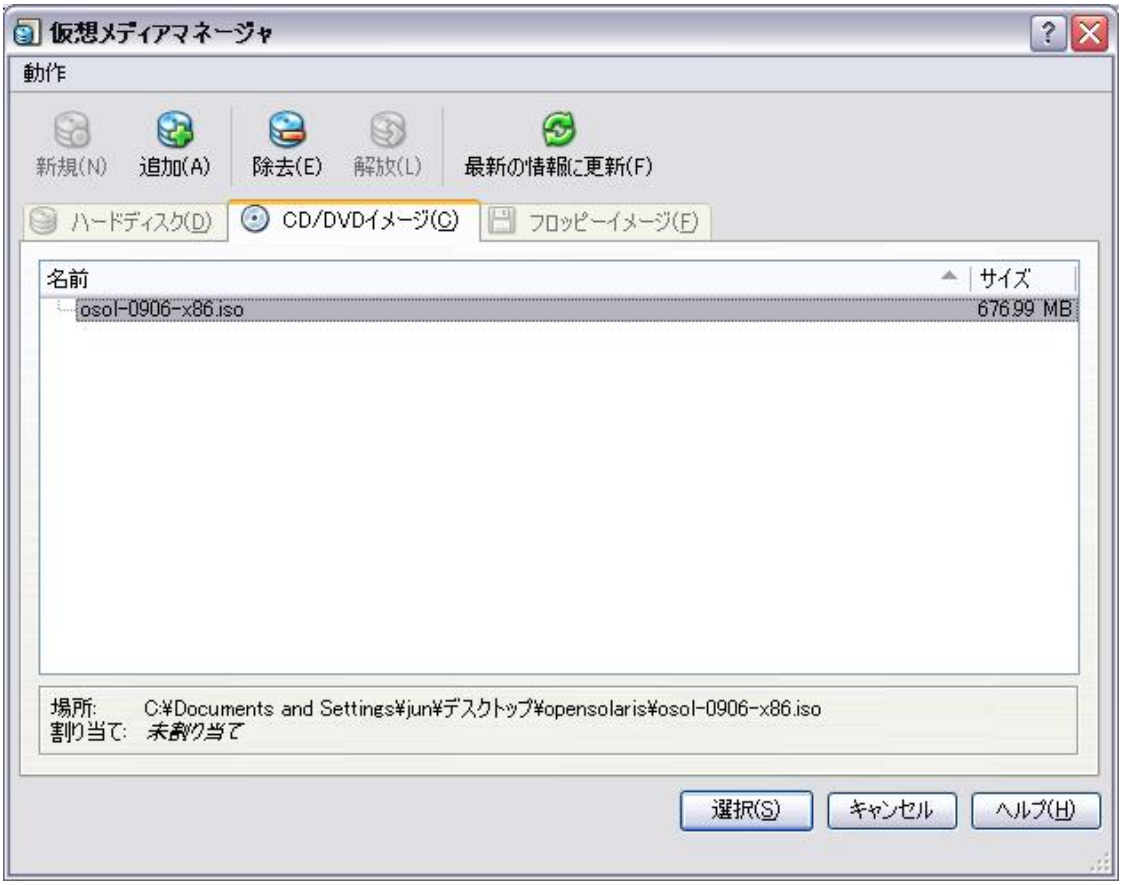
(5) 仮想メディアマネージャ画面です。“追加”ボタンをクリックします。



(6) インストールする ISO ファイルを選択して、“開く”をクリックします。



(7) 仮想メディアマネージャ画面です。先程の ISO ファイルを選択し、“選択”をクリックします。

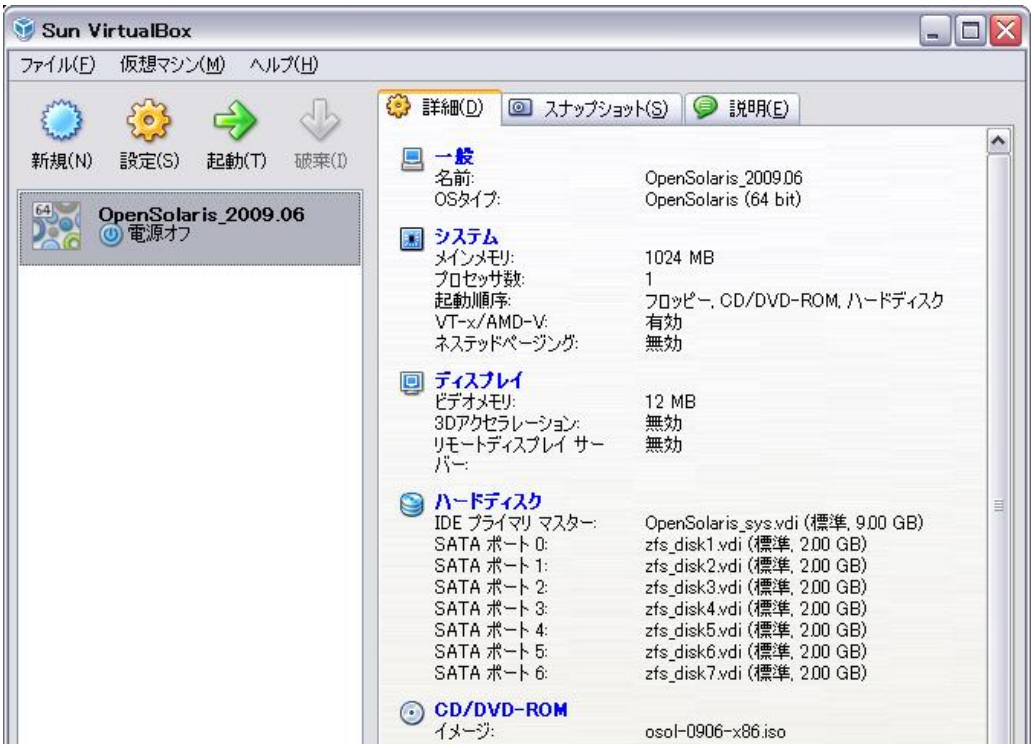


(8) CD/DVD-ROM 設定画面です。先程の ISO イメージファイルが選択されていることを確認し、“OK”をクリックします。

(9) Sun xVM VirtualBox 起動画面です。全ての設定が反映されていることを確認します。

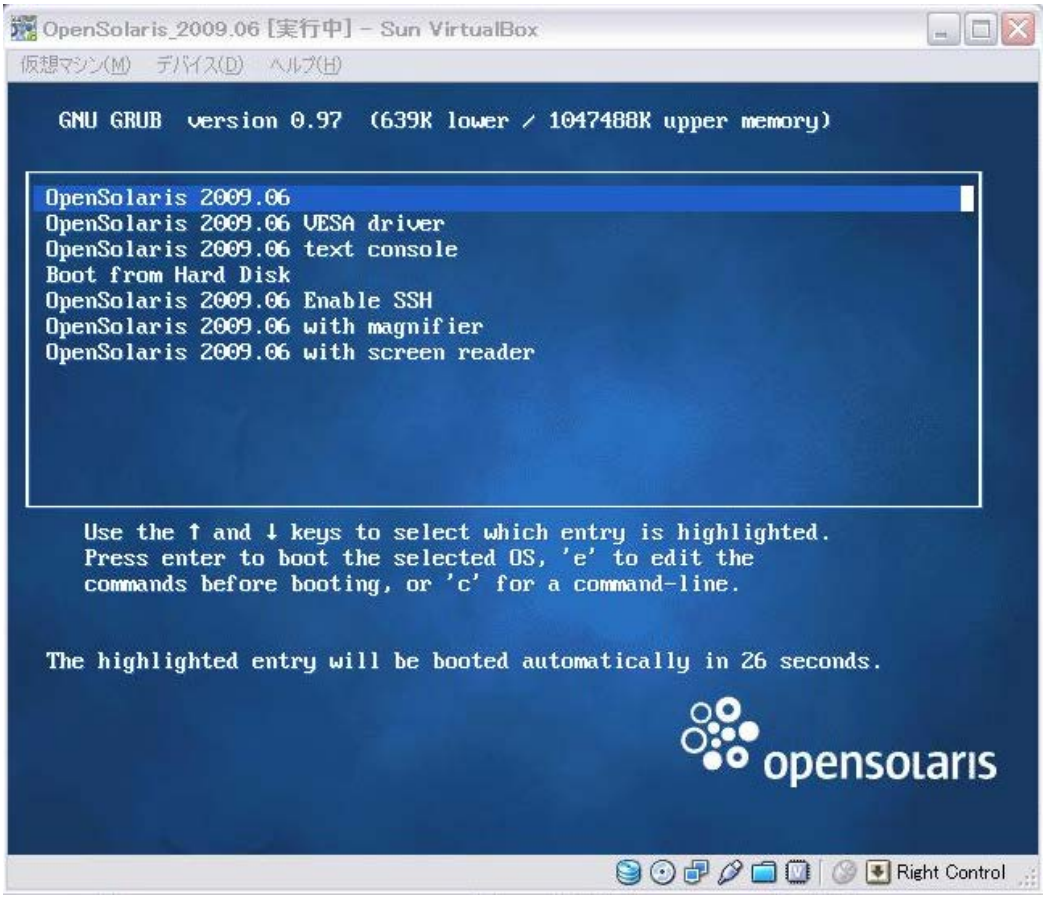
B-5. OpenSolaris OS のインストール

(1) Sun xVM VirtualBox 起動画面です。仮想マシンが選択されていることを確認して、“起動”ボタンをクリックします。

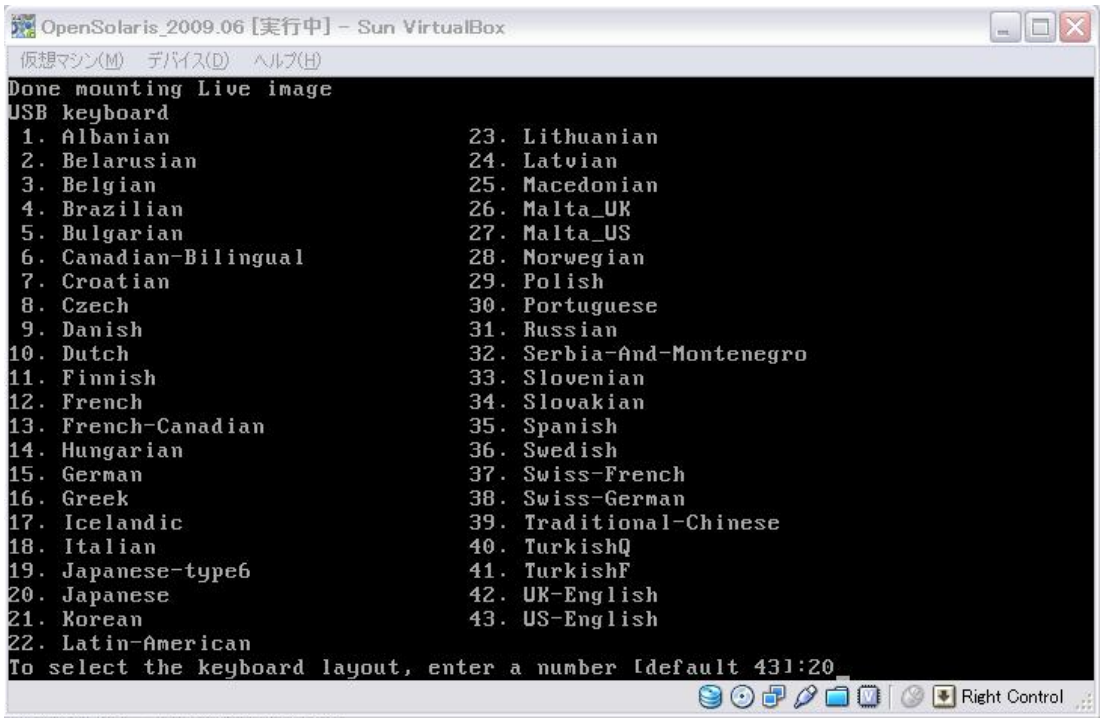


(2) BIOS 画面です。キーボード/マウスのキャプチャの終了は“右(→)＋Ctrl キー”です。

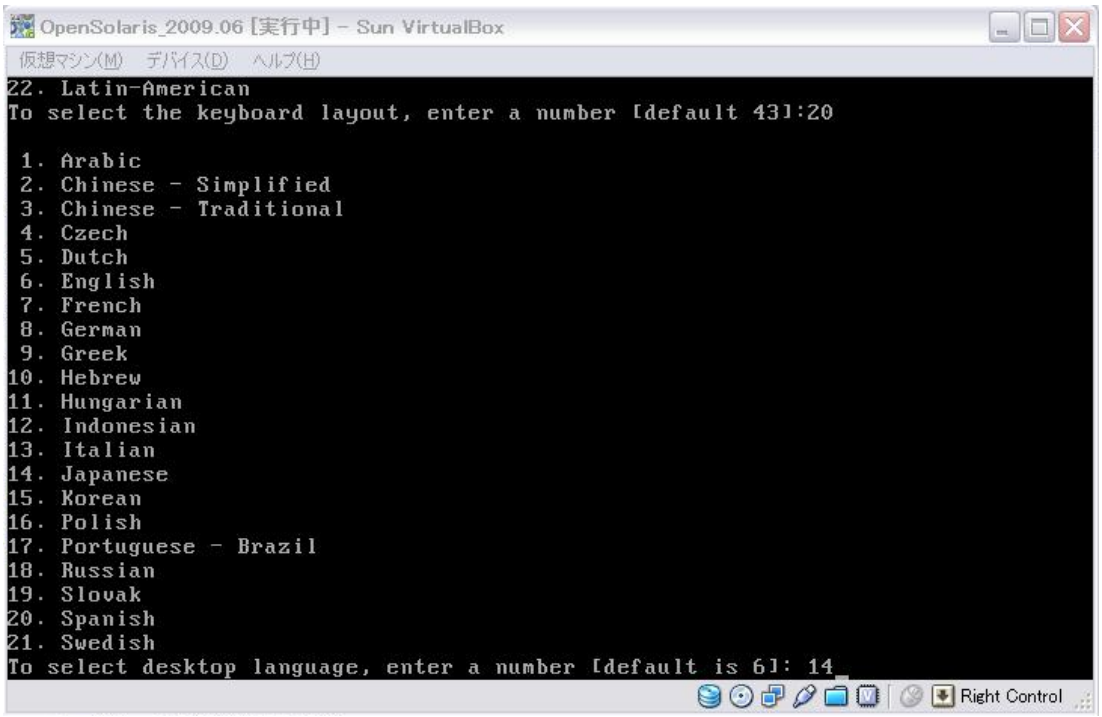
(3) GRUB 画面です。OpenSolaris 2009.06 をリターンします。



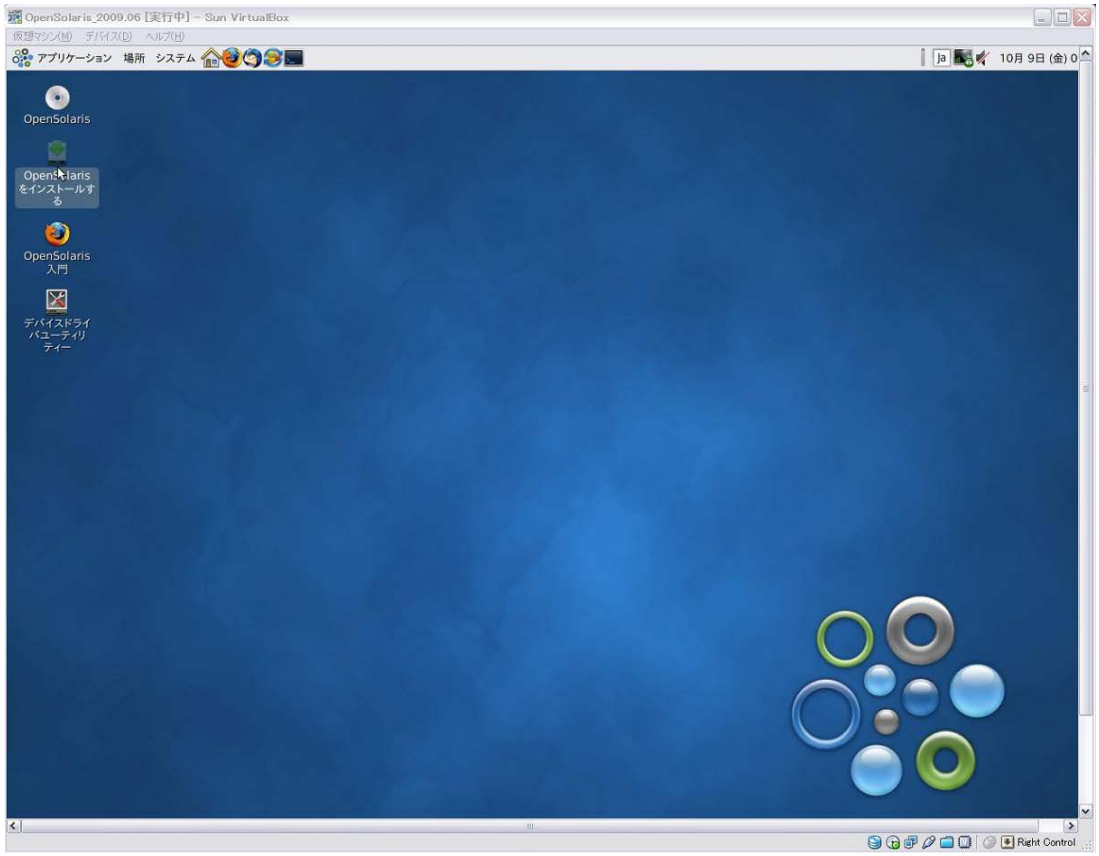
(4) キーボードタイプの選択画面です。"Japanese"の"20"を応答します。



(5) 言語の選択画面です。“Japanese”の“14”を応答します。

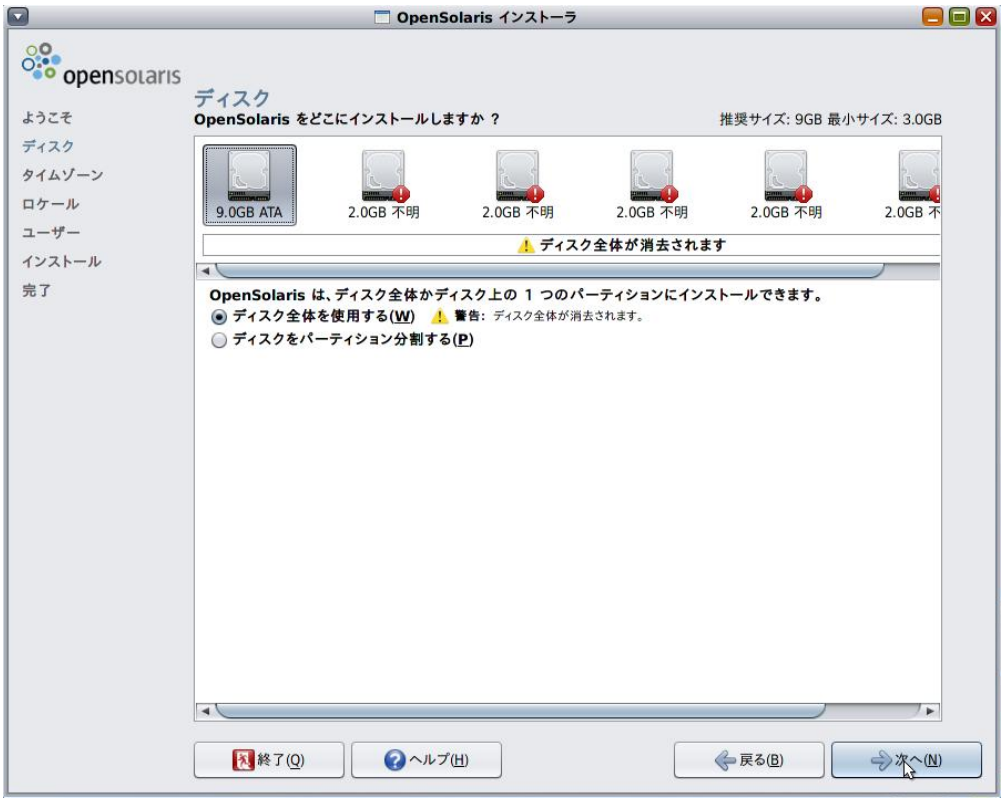


(6) OpenSolaris(LiveCD)の起動画面です。“OpenSolaris をインストールする”アイコンをクリックして、インストーラを起動します。



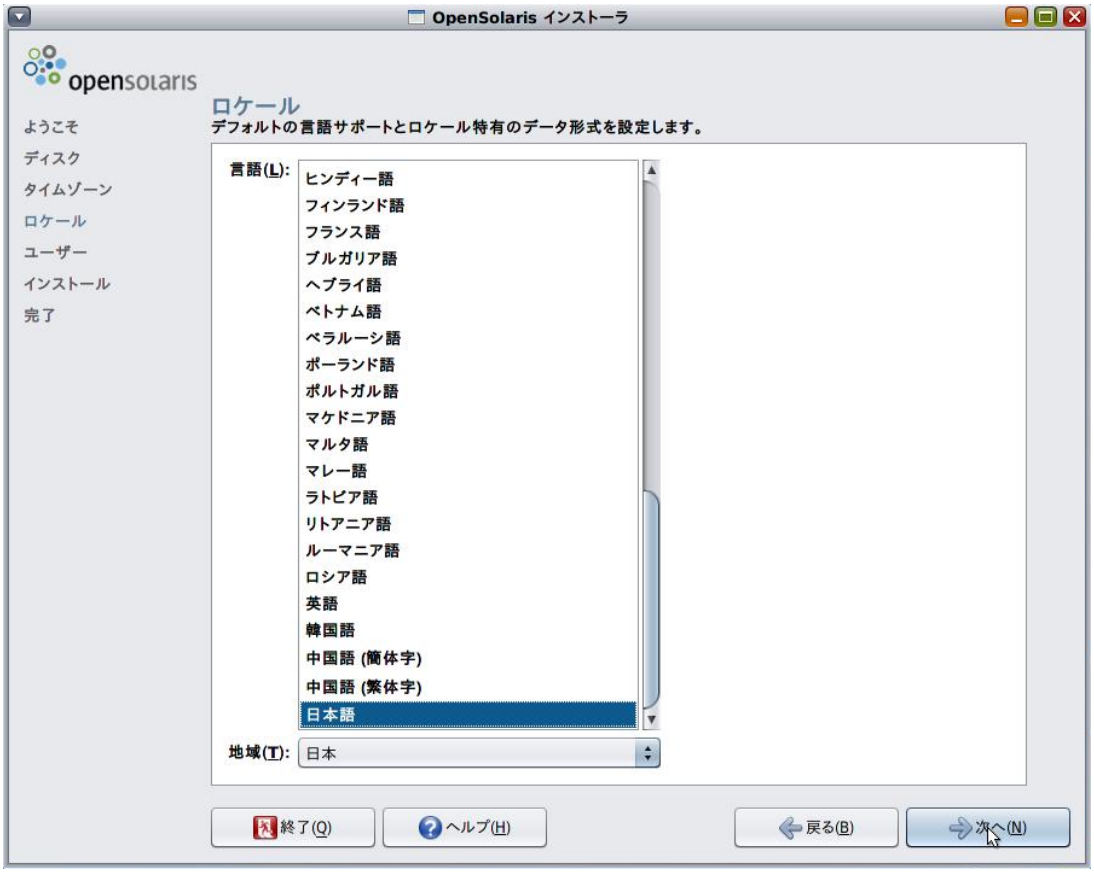
(7) OpenSolaris インストーラ画面です。“次へ”をクリックします。

(8) ディスクの選択画面です。システム用のディスクが選択されていることを確認して、“次へ”をクリックします。
ここでは“ディスク全体を使用する”を選択しています。



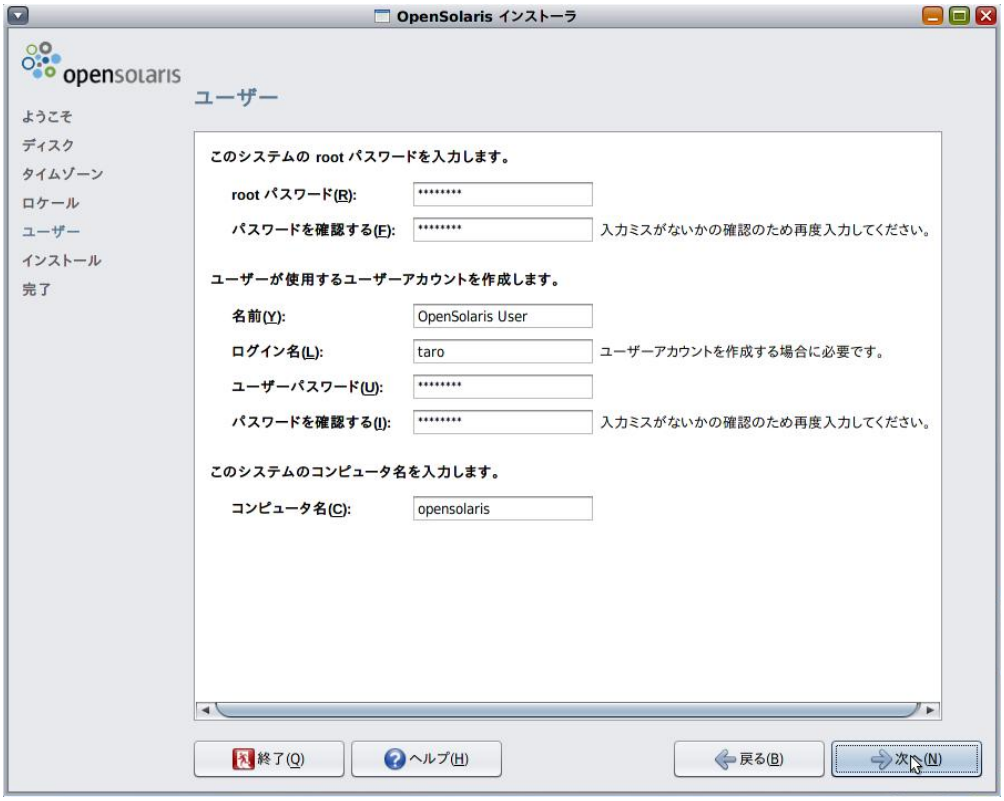
(9) タイムゾーンおよび日付と時刻の設定画面です。設定して、“次へ”をクリックします。

(10) ロケールの選択画面です。確認して、“次へ”をクリックします。



(11) ユーザーの設定画面です。OpenSolaris では root は役割（ロール)になっており、直接ログインする

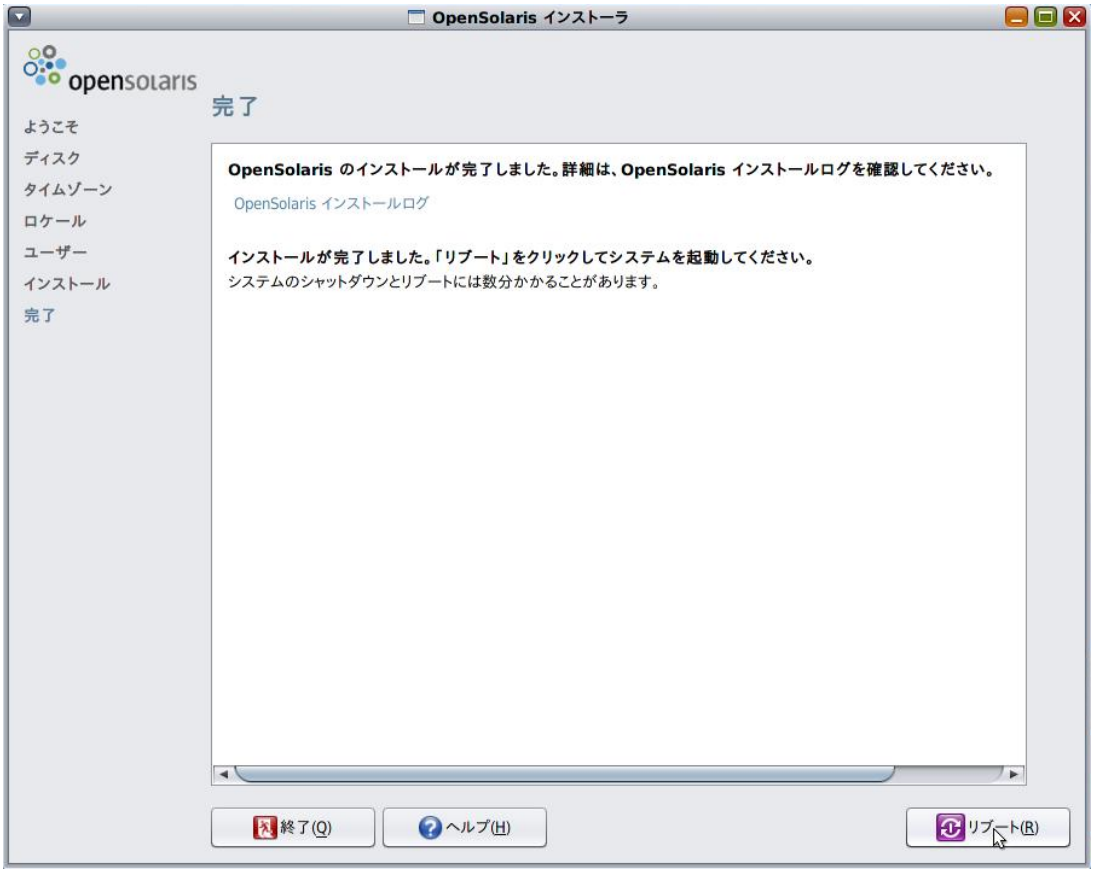
ことができません。そのため、管理操作を行うには、ここで作成したユーザーアカウントでログインした後、“su”コマンドにて root 役割になる必要があります。入力して、“次へ”をクリックします。



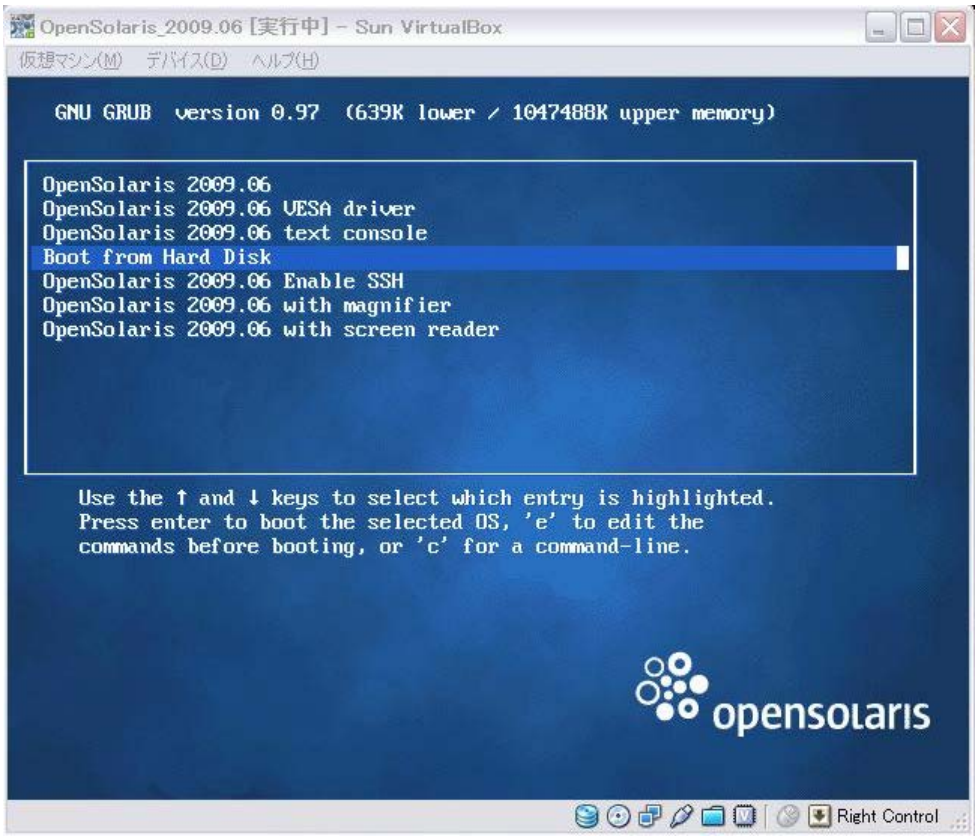
(12) 設定確認画面です。問題がなければ、“インストール”をクリックします。

(13) インストールの進捗を示す画面です。

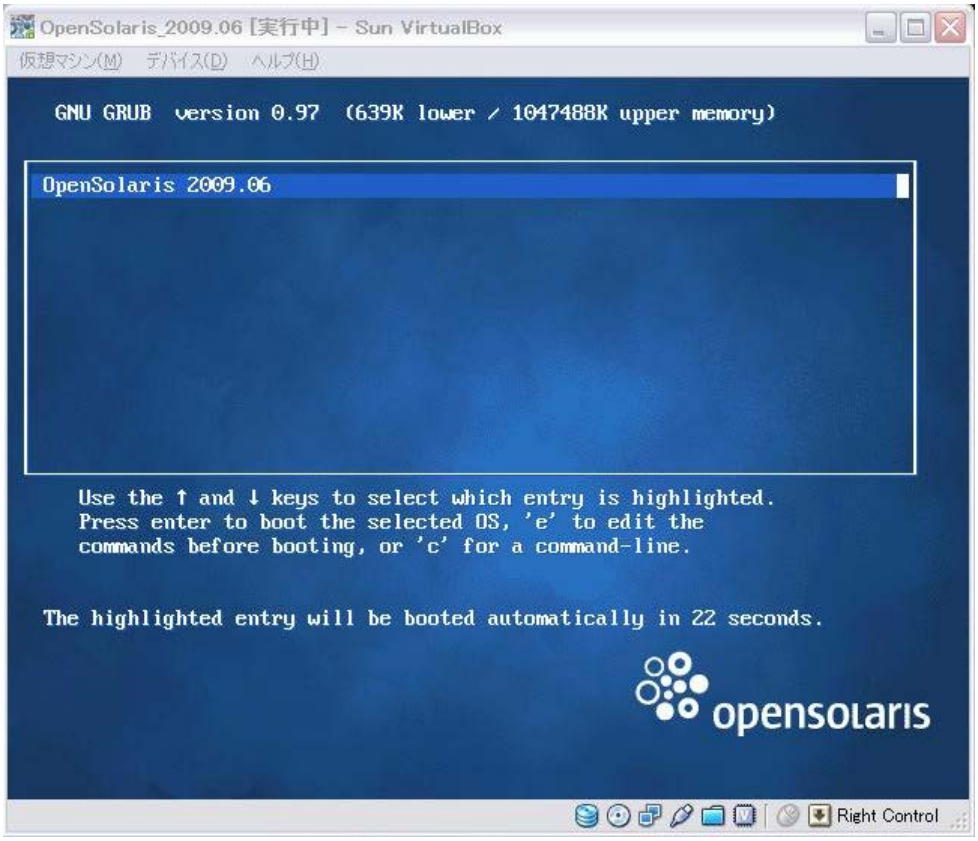
(14) インストール完了画面です。“リブート”をクリックします。



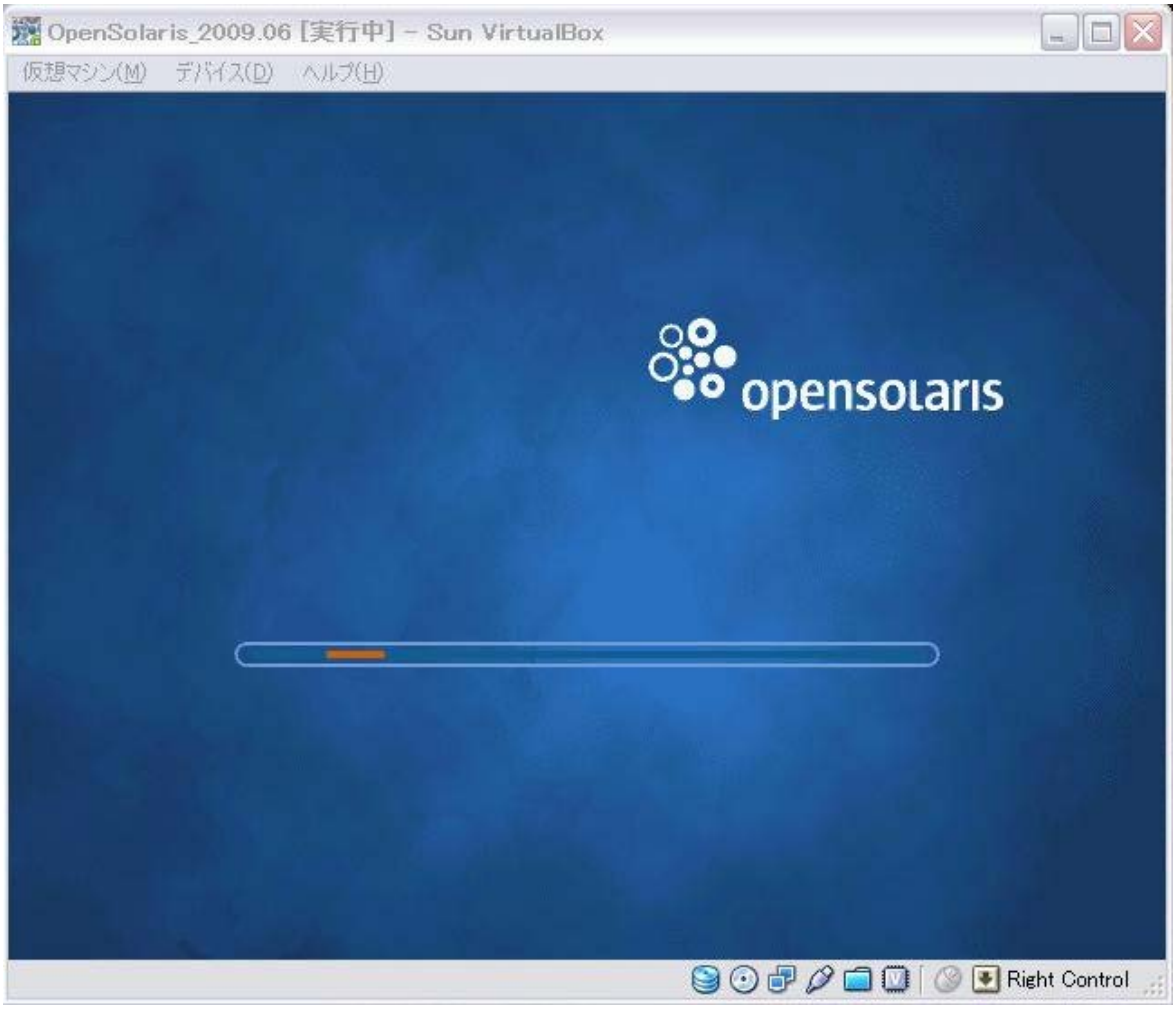
(15) GRUB 画面です。“Boot from Hard Disk”を選択して、ディスクブートします。



(16) 起動する OS を選択します。このままリターンします。



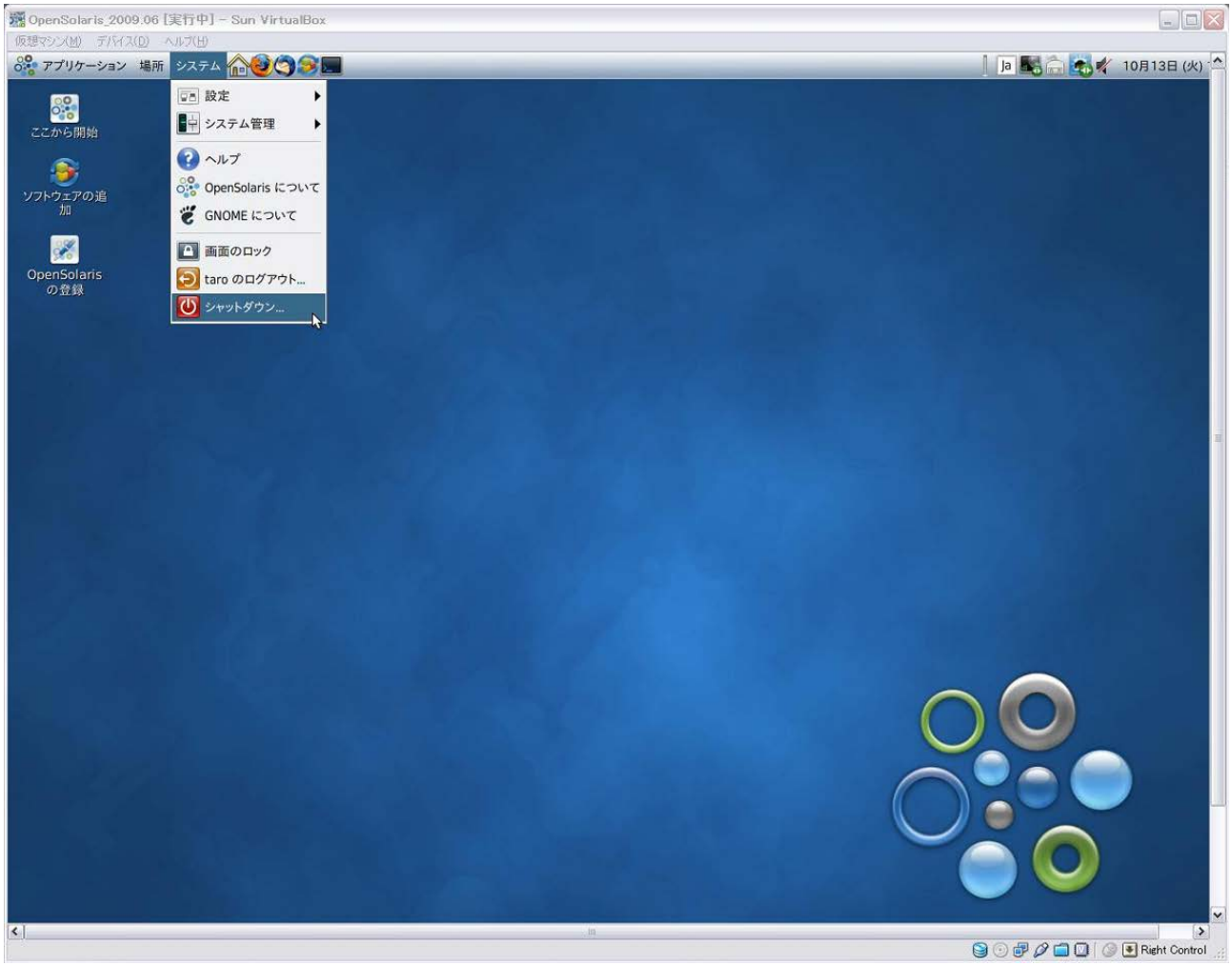
(17) OpenSolaris の起動画面です。しばらく待ちます。



(18) ユーザー名入力画面です。先程設定したユーザー名を入力して、"OK"をクリックします。

(19) パスワード入力画面です。パスワードを入力して、"OK"をクリックします。

(20) デスクトップ画面です。メニューから“システム”→“シャットダウン”を選択します。

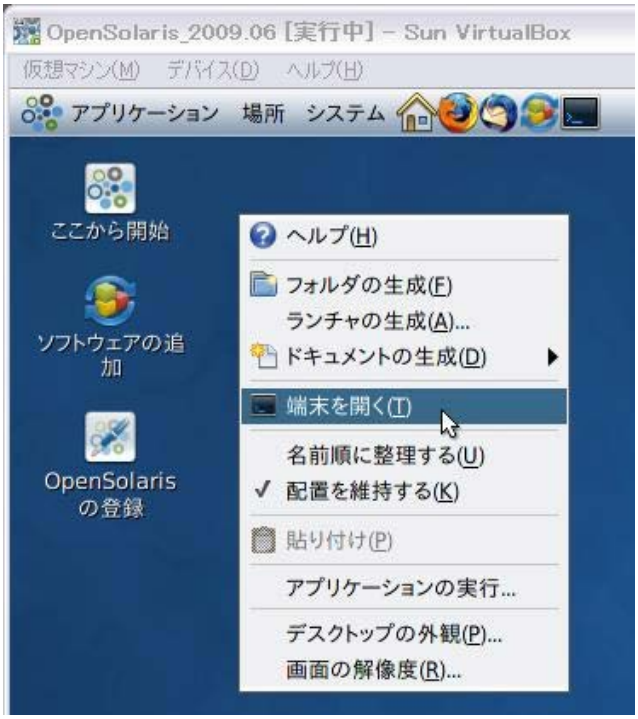


(21) シャットダウン確認メッセージです。“シャットダウン”をクリックします。

(22) Sun xVM VirtualBox 起動画面で、“設定”ボタンをクリックして“CD/DVD-ROM”の設定画面を開きます。次回からの起動時にディスクブートするために、“CD/DVD ドライブのマウント”のチェックを外しておきます。“OK”をクリックします。

B-6. ZFS 情報の確認

(1) OpenSolaris を起動して、端末エミュレータを起動します。メニュー欄の“”をクリックするか、下図のように背景を右クリックしてメニューの中から“端末を開く”をクリックしてください。



(2) 端末エミュレータ起動画面です。

(3) OS 情報の確認を行います。端末エミュレータ上で以下のコマンドを実行します。

```
root@opensolaris:~$ uname –a ⏎
SunOS opensolaris 5.11 snv_111b i86pc i386 i86pc Solaris
root@opensolaris:~$ cat /etc/release ⏎
                            OpenSolaris 2009.06 snv_111b X86
          Copyright 2009 Sun Microsystems, Inc.  All Rights Reserved.
                        Use is subject to license terms.
                             Assembled 07 May 2009
root@opensolaris:~$
```

(4) rpool の確認を行います。pool 情報の取得には“zpool”コマンドを使用します。また、“zpool get all <ストレージプール名>”で詳細情報を確認することができます。端末エミュレータ上で以下のコマンドを実行します。

```
root@opensolaris:~$ zpool list ⏎
NAME   SIZE    USED   AVAIL   CAPHEALTHALTROOT
rpool  8.94G   3.49G   5.44G   39% ONLINE -
root@opensolaris:~$ zpool status rpool ⏎
  pool : rpool
 state: ONLINE
 scrub : none requested
config:

        NAME  STATE      READ  WRITE CKSUM
        rpool  ONLINE  0 0 0
          c7d0s0ONLINE  0 0 0

errors: No known data errors
root@opensolaris:~$ zpool get all rpool ⏎
NAME  PROPERTY     VALUE                    SOURCE
rpool  size            8.94G                    -
rpool  used            3.49G                    -
rpool  available       5.44G                    -
rpool  capacity        39%                      -
rpool  altroot         -                        default
rpool  health          ONLINE                   -
rpool  guid            6824308044772421765      default
rpool  version         14                       default
rpool  bootfs          rpool/ROOT/opensolaris   local
rpool  delegation      on                       default
rpool  autoreplace     off                      default
rpool  cachefile       -                        default
rpool  failmode        wait                     default
rpool  listsnapshots   off                      default
root@opensolaris:~$
```

(5) zpool のバージョンの確認を行います。端末エミュレータ上で以下のコマンドを実行します。

```
root@opensolaris:~$ zpool upgrade –v ⏎
This system is currently running ZFS pool version 14.

The following versions are supported:

VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
 7   Separate intent log devices
 8   Delegated administration
 9   refquota and refreservation properties
 10  Cache devices
 11  Improved scrub performance
 12  Snapshot properties
 13  snapused property
 14  passthrough-x aclinherit support
For more information on a particular version, including supported releases, see:

http://www.opensolaris.org/os/community/zfs/version/N

Where 'N' is the version number.
root@opensolaris:~$
```

(6) rpool 上の zfs filesystem の確認を行います。端末エミュレータ上で以下のコマンドを実行します。

```
root@opensolaris:~$ zfs list ⏎
NAME                      USED   AVAIL  REFER  MOUNTPOINT
rpool                     3.90G  4.90G  77.5K  /rpool
rpool/ROOT                3.01G  4.90G  19K    legacy
rpool/ROOT/opensolaris    3.01G  4.90G  2.87G  /
rpool/dump                 274M  4.90G  274M   -
rpool/export               125M  4.90G  21K    /export
rpool/export/home          125M  4.90G  21K    /export/home
rpool/export/home/taro     125M  4.90G  125M   /export/home/taro
rpool/swap                 512M  5.30G  101M   -
root@opensolaris:~$
```

B-7. pool の作成および確認

(1) pool の作成には、“zpool create”コマンドを使用します。“spare”オプションを使用して、スペアディスクの設定も可能です。指定できる RAID 構成は以下に示します。

- mirror　ミラー
- raidz　シングルパリティ
- raidz2　デュアルパリティ

ここでは、以下の構成内容で作成します。作業は端末エミュレータ上で行います。

- ディスク 7 本
- RAID-Z2(4＋2)
- HotSpare×1

(2) “format”コマンドを使用して、ディスクの確認を行います。確認後、“Ctrl-C”で終了します。1～7 番が zfs 用に用意した仮想ディスクデバイスです。0 番は OS 用です。

```
root@opensolaris:~$ format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c7d0 <DEFAULT cyl 1171 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@1,1/ide@0/cmdk@0,0
       1. c9t0d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@0,0
       2. c9t1d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@1,0
       3. c9t2d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@2,0
       4. c9t3d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@3,0
       5. c9t4d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@4,0
       6. c9t5d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@5,0
       7. c9t6d0 <DEFAULT cyl 1022 alt 2 hd 128 sec 32>
          /pci@0,0/pci8086,2829@d/disk@6,0
Specify disk (enter its number): ^C
root@opensolaris:~$
```

(3) “zpool　create”コマンドを使用して、Pool　tpool(RAID-Z2)を作成します。“c9t0d0～c9t5d0”で RAID　Z2 を作成し、“c9t6d0”をスペアに割り当てます。また、raidz2 の代わりに raidz1、mirror 等を指定することによって異なる RAID 構成を作成することもできます。

```
root@opensolaris:~$ zpool create tpool raidz2 c9t0d0 c9t1d0 c9t2d0 c9t3d0
c9t4d0 c9 t5d0 spare c9t6d0 ⏎
```

- 作成後、“zpool list”コマンドで tpool の追加を確認します。

```
root@opensolaris:~$ zpool list ⏎
NAME  SIZE USED AVAILCAPHEALTH    ALTROOT
rpool  8.94G3.50G5.44G39% ONLINE     -
tpool  11.9G2.81M  11.9G  0% ONLINE     -
root@opensolaris:~$
```

(4) “zpool status”コマンドを使用して、tpool の情報を確認します。

```
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: ONLINE
 scrub : none requested
config:

        NAME        STATE     READ WRITE CKSUM
        tpool       ONLINE       0     0     0
          raidz2    ONLINE       0     0     0
            c9t0d0  ONLINE       0     0     0
            c9t1d0  ONLINE       0     0     0
            c9t2d0  ONLINE       0     0     0
            c9t3d0  ONLINE       0     0     0
            c9t4d0  ONLINE       0     0     0
            c9t5d0  ONLINE       0     0     0
        spares
          c9t6d0 AVAIL

errors: No known data errors
root@opensolaris:~$
```

(5) tpool が“/tpool”にマウントされていることを確認します。

```
root@opensolaris:~$ zfs list ⏎
NAME     USED AVAILREFER MOUNTPOINT
rpool    3.90G4.90G  77.5K  /rpool
rpool/ROOT    3.01G4.90G    19Klegacy
rpool/ROOT/opensolaris  3.01G4.90G  2.87G  /
rpool/dump    274M 4.90G   274M  -
rpool/export    125M 4.90G    21K/export
rpool/export/home  125M 4.90G    21K/export/home
rpool/export/home/taro  125M 4.90G   125M /export/home/taro
rpool/swap   512M 5.30G   101M  -
tpool     152K 7.79G   38.0K /tpool
root@opensolaris:~$ df –k ⏎
 Filesystem             1K-blocks      Used Available Use% Mounted on
 rpool/ROOT/opensolaris
                        8144465   3005270   5139195  37% /
 swap                     564264       304    563960   1% /etc/svc/volatile
 /usr/lib/libc/libc_hwcap2.so.1
                        8144465   3005270   5139195  37% /lib/libc.so.1
 swap                     563972        12    563960   1% /tmp
 swap                     564004        44    563960   1% /var/run
 rpool/export           5139216        21   5139195   1% /export
 rpool/export/home      5139216        21   5139195   1% /export/home
 rpool/export/home/taro
                        5267089    127894   5139195   3% /export/home/taro
 rpool                  5139272        78   5139195   1% /rpool
 tpool                  8163426        38   8163388   1% /tpool
root@opensolaris:~$
```

(6) tpool のデータセットのプロパティと、プール全体の統計、およびプール内の各ディスクの使用状況統計を確認します。

```
root@opensolaris:~$ zfs get all tpool ⏎
NAME PROPERTY   VALUE    SOURCE
tpool type      filesystem  -
tpool creation  水 10 月 14 11:21 2009 -
tpool used      152K     -
tpool available 7.79G    -
      ・
      ・
      ・
tpool usedbychildren  114K     -
tpool usedbyrefreservation 0     -
root@opensolaris:~$ zpool iostat –v ⏎
                capacity     operations    bandwidth
pool          used  avail   read  write   read  write
----------  -----  -----  -----  -----  -----  -----
--                          --     --      -      -
rpool        3.50G  5.44G      9      2   596K  43.2K
  c7d0s0     3.50G  5.44G      9      2   596K  43.2K
----------  -----  -----  -----  -----  -----  -----
--                          --     --      -      -
tpool        2.81M  11.9G      0     37     81  72.0K
  raidz2     2.81M  11.9G      0     37     81  72.0K
    c9t0d0       -      -      1     24  64.9K   262K
    c9t1d0       -      -      1     23  64.9K   262K
    c9t2d0       -      -      1     24  65.6K   263K
    c9t3d0       -      -      1     24  65.6K   263K
    c9t4d0       -      -      1     23  64.9K   262K
    c9t5d0       -      -      1     23  65.6K   261K
----------  -----  -----  -----  -----  -----  -----
--                          --     --      -      -
root@opensolaris:~$
```

B-8. pool の操作

(1) pool の export を行います。pool のエクスポート(アンマウント)は“zpool export <ストレージプール名>”を使用します。また、“zpool import”コマンドで、インポート可能な pool を一覧表示します。

```
root@opensolaris:~$ zpool export tpool ⏎
root@opensolaris:~$ zpool status tpool ⏎
cannot open 'tpool': no such pool
root@opensolaris:~$ zpool import ⏎
   pool : tpool
     id: 10914955493912306664  ←同じ名前の pool が複数ある場合、この id を利用
state: ONLINE          して pool のインポートが可能です。
action: The pool can be imported using its name or numeric identifier.
config:

          tpool         ONLINE
            raidz2      ONLINE
              c9t0d0    ONLINE
              c9t1d0    ONLINE
              c9t2d0    ONLINE
              c9t3d0    ONLINE
              c9t4d0    ONLINE
              c9t5d0    ONLINE
          spares
            c9t6d0
root@opensolaris:~$
```

(2) “zpool import <ストレージプール名>”コマンドでインポートします。“zpool import <ID>”でもインポートが可能です。

```
root@opensolaris:~$ zpool import tpool ⏎
root@opensolaris:~$ zpool status tpool ⏎
   pool : tpool
 state: ONLINE
 scrub : none requested
config:

          NAME        STATE     READ      WRITE     CKSUM
          tpool       ONLINE       0          0         0
            raidz2    ONLINE       0          0         0
              c9t0d0  ONLINE       0          0         0
              c9t1d0  ONLINE       0          0         0
              c9t2d0  ONLINE       0          0         0
              c9t3d0  ONLINE       0          0         0
              c9t4d0  ONLINE       0          0         0
              c9t5d0  ONLINE       0          0         0
          spares
            c9t6d0 AVAIL

errors: No known data errors
root@opensolaris:~$ zpool import ⏎
root@opensolaris:~$
```

(3) “zpool remove <ストレージプール名> <dev>”コマンドを使用して、pool からスペアディスクを外します。

```
root@opensolaris:~$ zpool remove tpool c9t6d0 ⏎
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: ONLINE
 scrub : none requested
config:

        NAME        STATE     READ    WRITE   CKSUM
        tpool       ONLINE       0        0       0
          raidz2    ONLINE       0        0       0
            c9t0d0  ONLINE       0        0       0
            c9t1d0  ONLINE       0        0       0
            c9t2d0  ONLINE       0        0       0
            c9t3d0  ONLINE       0        0       0
            c9t4d0  ONLINE       0        0       0
            c9t5d0  ONLINE       0        0       0
  ←ここに spare が無いことを確認します。
errors: No known data errors
root@opensolaris:~$
```

(4) “zpool add <ストレージプール名> spare <dev>”コマンドを使用して、pool にスペアディスクを追加します。

```
root@opensolaris:~$ zpool add tpool spare c9t6d0 ⏎
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: ONLINE
 scrub : none requested
config:

        NAME        STATE     READ    WRITE    CKSUM
        tpool       ONLINE       0        0       0
          raidz2    ONLINE       0        0       0
            c9t0d0  ONLINE       0        0       0
            c9t1d0  ONLINE       0        0       0
            c9t2d0  ONLINE       0        0       0
            c9t3d0  ONLINE       0        0       0
            c9t4d0  ONLINE       0        0       0
            c9t5d0  ONLINE       0        0       0
        spares
          c9t6d0 AVAIL  ←spare が追加されたことを確認します。

errors: No known data errors
root@opensolaris:~$
```

(5) “zpool offlline <ストレージプール名> <dev>”コマンドを使用して、ディスクをオフラインにします。

```
root@opensolaris:~$ zpool offline tpool c9t0d0 ⏎
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: DEGRADED
status: One or more devices has been taken offline by the administrator.
   Sufficient replicas exist for the pool to continue functioning in a
   degraded state.
action: Online the device using 'zpool online' or replace the device with
   'zpool replace'.
 scrub : none requested
config:

        NAME        STATE     READ  WRITE  CKSUM
        tpool       DEGRADED     0      0      0
          raidz2    DEGRADED     0      0      0
            c9t0d0  OFFLINE      0      0      0
            c9t1d0  ONLINE       0      0      0
            c9t2d0  ONLINE       0      0      0
            c9t3d0  ONLINE       0      0      0
            c9t4d0  ONLINE       0      0      0
            c9t5d0  ONLINE       0      0      0
        spares
          c9t6d0 AVAIL

errors: No known data errors
root@opensolaris:~$
```

(6) “zpool online <ストレージプール名> <dev>”コマンドを使用して、ディスクをオンラインにします。

```
root@opensolaris:~$ zpool online tpool c9t0d0 ⏎
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: ONLINE
 scrub : resilver completed after 0h0m with 0 errors on Wed Oct 14 15:22:12 2009
config:

        NAME        STATE     READ  WRITE  CKSUM
        tpool       ONLINE       0      0      0
          raidz2    ONLINE       0      0      0
            c9t0d0  ONLINE       0      0      0  512 resilvered
            c9t1d0  ONLINE       0      0      0
            c9t2d0  ONLINE       0      0      0
            c9t3d0  ONLINE       0      0      0
            c9t4d0  ONLINE       0      0      0
            c9t5d0  ONLINE       0      0      0
        spares
          c9t6d0 AVAIL

errors: No known data errors
root@opensolaris:~$
```

(7) “zpool detach <ストレージプール名> <dev>”コマンドを使用して、pool からディスクを取り外します。RAID-Z/Z2 はそのまま取り外すことができないため、“zpool replace <ストレージプール名> <dev> <spare dev>”コマンドを使用して、最初にスペアと入れ替えます。

```
root@opensolaris:~$ zpool detach tpool c9t0d0 ⏎
cannot detach c9t0d0: only applicable to mirror and replacing vdevs
root@opensolaris:~$ zpool replace tpool c9t0d0 c9t6d0 ⏎
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: ONLINE
 scrub : resilver completed after 0h0m with 0 errors on Wed Oct 14 15:23:40 2009
config:

        NAME          STATE     READ WRITE CKSUM
        tpool         ONLINE       0     0     0
          raidz2      ONLINE       0     0     0
            spare     ONLINE       0     0     0
              c9t0d0  ONLINE       0     0     0
              c9t6d0  ONLINE       0     0     0  36.5K resilvered
            c9t1d0    ONLINE       0     0     0
            c9t2d0    ONLINE       0     0     0
            c9t3d0    ONLINE       0     0     0
            c9t4d0    ONLINE       0     0     0
            c9t5d0    ONLINE       0     0     0
        spares
          c9t6d0 INUSE  currently in use

errors: No known data errors
root@opensolaris:~$
```

(8) “zpool detach <ストレージプール名> <dev>”コマンドを使用して、pool からディスクを取り外します。再度ディスクを追加する場合は、“zpool add”コマンドでスペアを追加してください。“zpool replace”コマンドを使用して、ディスク配置を元に戻すこともできます。

```
root@opensolaris:~$ zpool detach tpool c9t0d0 ⏎
root@opensolaris:~$ zpool status tpool ⏎
  pool : tpool
 state: ONLINE
 scrub : resilver completed after 0h0m with 0 errors on Wed Oct 14 15:23:40 2009
config:

        NAME        STATE     READ WRITE CKSUM
        tpool       ONLINE       0     0     0
          raidz2    ONLINE       0     0     0
            c9t6d0  ONLINE       0     0     0  36.5K resilvered
            c9t1d0  ONLINE       0     0     0
            c9t2d0  ONLINE       0     0     0
            c9t3d0  ONLINE       0     0     0
            c9t4d0  ONLINE       0     0     0
            c9t5d0  ONLINE       0     0     0

errors: No known data errors
root@opensolaris:~$
```

(9) “zpool destroy <ストレージプール名>”コマンドを使用して、pool を削除します。

```
root@opensolaris:~$ zpool destroy tpool ⏎
root@opensolaris:~$ zpool status tpool ⏎
cannot open 'tpool': no such pool
root@opensolaris:~$
```

B-9. ZFS File System の操作

(1) 「B-7. pool の作成および確認」で作成した pool と同じディスク構成で“RAID-Z2”を作成します。

```
root@opensolaris:~$ zpool create tpool raidz2 c9t0d0 c9t1d0 c9t2d0 c9t3d0 c9t4d0 c9t5d0 spare c9t6d0 ⏎
root@opensolaris:~$ zpool list ⏎
NAME  SIZE USED AVAILCAPHEALTH    ALTROOT
rpool  8.94G3.50G5.44G39% ONLINE     -
tpool  11.9G  206K  11.9G  0% ONLINE     -
root@opensolaris:~$
```

(2) “zfs create <filesystem 名>”コマンドで ZFS ファイルシステムを作成します。作成した時点でマウントされます。

```
root@opensolaris:~$ zfs create tpool/data1 ⏎
root@opensolaris:~$ zfs create tpool/data2 ⏎
root@opensolaris:~$ zfs list ⏎
NAME                    USED  AVAIL  REFER  MOUNTPOINT
rpool                  3.90G  4.90G  77.5K  /rpool
rpool/ROOT             3.01G  4.90G    19K  legacy
rpool/ROOT/opensolaris 3.01G  4.90G  2.86G  /
rpool/dump              274M  4.90G   274M  -
rpool/export            125M  4.90G    21K  /export
rpool/export/home       125M  4.90G    21K  /export/home
rpool/export/home/taro  125M  4.90G   125M  /export/home/taro
rpool/swap              512M  5.30G   101M  -
tpool                   236K  7.79G  40.0K  /tpool
tpool/data1            38.0K  7.79G  38.0K  /tpool/data1
tpool/data2            38.0K  7.79G  38.0K  /tpool/data2
root@opensolaris:~$
```

(3) “zfs umount”コマンドを使用して、アンマウントします。

```
root@opensolaris:~$ zfs umount tpool/data1 ⏎
root@opensolaris:~$ df –k ⏎
Filesystem             1K-blocks      Used   Available Use%  Mounted on
rpool/ROOT/opensolaris
                       8141363   3004124   5137240    37%  /
swap                    534724       304    534420     1%  /etc/svc/volatile
/usr/lib/libc/libc_hwcap2.so.1
                       8141363   3004124   5137240    37%  /lib/libc.so.1
swap                    534432        12    534420     1%  /tmp
swap                    534464        44    534420     1%  /var/run
rpool/export           5137261        21   5137240     1%  /export
rpool/export/home      5137261        21   5137240     1%  /export/home
rpool/export/home/taro
                       5265134    127894   5137240     3%  /export/home/taro
rpool                  5137317        78   5137240     1%  /rpool
tpool                  8163344        42   8163302     1%  /tpool
tpool/data2            8163340        38   8163302     1%  /tpool/data2
root@opensolaris:~$
```

(4) “zfs mount”コマンドを使用して、マウントします。“mount”コマンドは使用できません。

```
root@opensolaris:~$ zfs mount tpool/data1 ⏎
root@opensolaris:~$ df –k ⏎
 Filesystem              1K-blocks      Used    Available    Use%    Mounted on
 rpool/ROOT/opensolaris
                         8141363    3004124    5137240      37%    /
 swap                     534724        304     534420       1%    /etc/svc/volatile
 /usr/lib/libc/libc_hwcap2.so.1
                         8141363    3004124    5137240      37%    /lib/libc.so.1
 swap                     534432         12     534420       1%    /tmp
 swap                     534464         44     534420       1%    /var/run
 rpool/export            5137261         21    5137240       1%    /export
 rpool/export/home       5137261         21    5137240       1%    /export/home
 rpool/export/home/taro
                         5265134     127894    5137240       3%    /export/home/taro
 rpool                   5137317         78    5137240       1%    /rpool
 tpool                   8163344         42    8163302       1%    /tpool
 tpool/data2             8163340         38    8163302       1%    /tpool/data2
 tpool/data1             8163340         38    8163302       1%    /tpool/data1
root@opensolaris:~$
```

(5) ファイルシステムの情報を表示します。“zfs get <all | ファイルシステム名>”コマンドを使用します。

```
root@opensolaris:~$ zfs get all tpool/data1 ⏎
NAME         PROPERTY  VALUE                  SOURCE
tpool/data1 type          filesystem             -
tpool/data1 creation      水 10 月 14 17:08 2009  -
tpool/data1 used          38.0K                  -
tpool/data1 available     7.79G                  -
      ・
      ・
      ・
tpool/data1 usedbychildren        0       -
tpool/data1 usedbyrefreservation 0          -
root@opensolaris:~$
```

(6) “zfs set mountpoint=<マウント先> <ファイルシステム名>”コマンドを使用してマウントポイントの変更を行います。自動的に再マウントされます。

```
root@opensolaris:~$ zfs set mountpoint=/data1 tpool/data1 ⏎
root@opensolaris:~$ zfs list ⏎
 NAME                      USED    AVAIL    REFER    MOUNTPOINT
 rpool                    3.90G    4.90G    77.5K    /rpool
 rpool/ROOT               3.01G    4.90G      19K    legacy
 rpool/ROOT/opensolaris   3.01G    4.90G    2.86G    /
 rpool/dump                274M    4.90G     274M    -
 rpool/export              125M    4.90G      21K    /export
 rpool/export/home         125M    4.90G      21K    /export/home
 rpool/export/home/taro    125M    4.90G     125M    /export/home/taro
 rpool/swap                512M    5.30G     101M    -
 tpool                     250K    7.79G    42.0K    /tpool
 tpool/data1              38.0K    7.79G    38.0K    /data1
 tpool/data2              38.0K    7.79G    38.0K    /tpool/data2
root@opensolaris:~$
```

● “mount”コマンド等、旧来の方法を利用したい場合はマウントポイントを“legacy”に設定します。マウント時は“mount –F zfs”を使用します。

```
root@opensolaris:~$ mount -F zfs tpool/data2 /mnt ⏎
filesystem 'tpool/data2' cannot be mounted using 'mount -F zfs'
Use 'zfs set mountpoint=/mnt' instead.
If you must use 'mount -F zfs' or /etc/vfstab, use 'zfs set mountpoint=legacy'.
See zfs(1M) for more information.
root@opensolaris:~$ zfs set mountpoint=legacy tpool/data2 ⏎
root@opensolaris:~$ zfs list ⏎
NAME                      USED  AVAIL  REFER  MOUNTPOINT
rpool                    3.90G  4.90G  77.5K  /rpool
rpool/ROOT               3.01G  4.90G    19K  legacy
rpool/ROOT/opensolaris   3.01G  4.90G  2.86G  /
rpool/dump                274M  4.90G   274M  -
rpool/export              125M  4.90G    21K  /export
rpool/export/home         125M  4.90G    21K  /export/home
rpool/export/home/taro    125M  4.90G   125M  /export/home/taro
rpool/swap                512M  5.30G   101M  -
tpool                     260K  7.79G  40.0K  /tpool
tpool/data1              38.0K  7.79G  38.0K  /data1
tpool/data2              38.0K  7.79G  38.0K  legacy
root@opensolaris:~$

root@opensolaris:~$ zfs mount tpool/data2 ⏎
cannot mount 'tpool/data2': legacy mountpoint
use mount(1M) to mount this filesystem
root@opensolaris:~$ mount -F zfs tpool/data2 /mnt ⏎
root@opensolaris:~$ df –k ⏎
df: the --kilobytes option is deprecated; use -k instead
Filesystem             1K-blocks      Used   Available Use% Mounted on
rpool/ROOT/opensolaris
                       8141369   3004125   5137245    37% /
swap                    534684       304    534380     1% /etc/svc/volatile
/usr/lib/libc/libc_hwcap2.so.1
                       8141369   3004125   5137245    37% /lib/libc.so.1
swap                    534392        12    534380     1% /tmp
swap                    534424        44    534380     1% /var/run
rpool/export           5137266        21   5137245     1% /export
rpool/export/home      5137266        21   5137245     1% /export/home
rpool/export/home/taro
                       5265139    127894   5137245     3% /export/home/taro
rpool                  5137322        78   5137245     1% /rpool
tpool                  8163320        38   8163282     1% /tpool
tpool/data1            8163320        38   8163282     1% /data1
tpool/data2            8163320        38   8163282     1% /mnt
root@opensolaris:~$
```

(7) ファイルシステムの設定変更を行います。“zfs set <プロパティ>=<値> <ファイルシステム名>”コマンドで、各種プロパティの設定変更を行うことができます。ファイルシステム作成時に、“zfs create -o <プロパティ>=<値>”コマンドを使用することで、設定を反映することもできます。また、“compression=on”とすることでファイルシステムを圧縮モードにすることができます(圧縮は設定後の書き込みから有効です)。

```
root@opensolaris:~$ zfs set compression=on tpool/data1 ⏎
root@opensolaris:~$ zfs get compression tpool/data1 ⏎
NAME   PROPERTY VALUE   SOURCE
tpool/data1 compression on  local
root@opensolaris:~$ zfs get all tpool/data3 ⏎
cannot open 'tpool/data3': dataset does not exist
root@opensolaris:~$ zfs create -o compression=on tpool/data3 ⏎
root@opensolaris:~$ zfs get compression tpool/data3 ⏎
NAME   PROPERTY VALUE   SOURCE
tpool/data3 compression on  local
root@opensolaris:~$
```

(8) “zfs list -t snapshot”コマンドを使用して、スナップショットの確認を行います。初期インストール時点で OS 領域のスナップショットが作成されています。また、Solaris 10/08 などの ZFS バージョンが古い環境では、“zfs list”コマンドで確認します。その際、スナップショットとファイルシステムが同時に表示されます。

```
root@opensolaris:~$ zfs list -t snapshot ⏎
NAME       USED AVAILREFER MOUNTPOINT
rpool/ROOT/opensolaris@install   146M      -  2.82G -
root@opensolaris:~$
```

(9) スナップショットのテストの準備を行います。スナップショットのテストのために、“tpool/data1”へファイルをコピーします。ここでは、先程の「マウントポイントの変更」で実行された、“tpool/data1”→“data1”への変更を“data1”→“tpool/data1”に戻して作業を行っています。

```
root@opensolaris:~$ zfs set mountpoint=/tpool/data1 tpool/data1 ⏎
root@opensolaris:~$ zfs list ⏎
 NAME                   USED   AVAIL   REFER  MOUNTPOINT
 rpool                  3.90G  4.90G   77.5K  /rpool
 rpool/ROOT             3.01G  4.90G     19K  legacy
 rpool/ROOT/opensolaris 3.01G  4.90G  2.86G  /
 rpool/dump              274M  4.90G   274M  -
 rpool/export            125M  4.90G     21K  /export
 rpool/export/home       125M  4.90G     21K  /export/home
 rpool/export/home/taro  125M  4.90G   125M  /export/home/taro
 rpool/swap              512M  5.30G   101M  -
 tpool                   307K  7.79G   40.0K  /tpool
 tpool/data1            38.0K  7.79G   38.0K  /tpool/data1
 tpool/data2             38.0K  7.79G   38.0K  legacy
 tpool/data3             38.0K  7.79G   38.0K  /tpool/data3
root@opensolaris:~$

root@opensolaris:~$ cp /etc/* /tpool/data1/. ⏎
cp: omitting directory `/etc/X11'
cp: omitting directory `/etc/acct'
cp: omitting directory `/etc/amd64'
  .
  .
  .
cp: omitting directory `/etc/zfs'
cp: omitting directory `/etc/zones'
```

```
root@opensolaris:~$ ls /tpool/data1 ⏎
TIMEZONE getty     mnttab   printers.conf sudoers
aliases  gksu.conf  motd     profile   sulogin
auto_home gnome-vfs-mime-magic mount   project  swap
      ・
      ・
      ・
fsdb   mkfs    power.confshutdown
fstyp  mknod   power.conf-Orig sock2path
root@opensolaris:~$
```

(10) “zfs snapshot <スナップショット名>”コマンドでスナップショットを作成します。また、“zfs get <all | スナップショット名>”コマンドでスナップショットの情報を確認できます。スナップショットは読み込み専用です。

```
root@opensolaris:~$ zfs snapshot tpool/data1@snap ⏎
root@opensolaris:~$ zfs list -t snapshot ⏎
NAME       USED AVAILREFER MOUNTPOINT
rpool/ROOT/opensolaris@install   146M      -  2.82G -
tpool/data1@snap          0      -  1.05M-
root@opensolaris:~$ zfs get all tpool/data1@snap ⏎
NAME    PROPERTY   VALUE    SOURCE
tpool/data1@snaptype    snapshot  -
tpool/data1@snapcreation    水 10 月 14 17:25 2009 -
tpool/data1@snapused    0    -
      ・
      ・
      ・
tpool/data1@snapprimarycache   all     default
tpool/data1@snapsecondarycache  all     default
root@opensolaris:~$
```

(11) スナップショットの確認を行います。スナップショットデータは“/<mountpoint>/.zfs/snapshot/<スナップショット名>”以下に保存されています。

```
root@opensolaris:~$ ls /tpool/data1/.zfs/snapshot/snap ⏎
TIMEZONE getty     mnttab   printers.conf sudoers
aliases  gksu.conf  motd     profile   sulogin
auto_home gnome-vfs-mime-magic mount   project  swap
      ・
      ・
      ・
fsdb   mkfs    power.confshutdown
fstyp  mknod   power.conf-Orig sock2path
root@opensolaris:~$
```

(12) “zfs rollback”コマンドを使用して、スナップショットの復元を行います。

```
root@opensolaris:~$ rm /tpool/data1/* ⏎
root@opensolaris:~$ ls /tpool/data1 ⏎
root@opensolaris:~$ zfs list -t snapshot ⏎
NAME       USED AVAILREFER MOUNTPOINT
rpool/ROOT/opensolaris@install   146M      -  2.82G -
tpool/data1@snap   1.05M       -  1.05M -
root@opensolaris:~$ zfs rollback tpool/data1@snap ⏎
root@opensolaris:~$ ls /tpool/data1 ⏎
TIMEZONE getty     mnttab   printers.conf sudoers
aliases  gksu.conf  motd     profile   sulogin
auto_home gnome-vfs-mime-magic mount   project  swap
      ・
```

```
        .
        .
fsdb    mkfs    power.confshutdown
fstyp   mknod   power.conf-Origsock2path
root@opensolaris:~$
```

(13) “zfs destroy”コマンドを使用して、スナップショット、およびファイルシステムを削除します。

```
root@opensolaris:~$ zfs destroy tpool/data1@snap ⏎
root@opensolaris:~$ zfs list -t snapshot ⏎
NAME       USED AVAILREFER MOUNTPOINT
rpool/ROOT/opensolaris@install   146M      -  2.82G -
root@opensolaris:~$ zfs destroy tpool/data1 ⏎
root@opensolaris:~$ zfs destroy tpool/data2 ⏎
root@opensolaris:~$ zfs destroy tpool/data3 ⏎
root@opensolaris:~$ zfs list ⏎
 NAME                      USED    AVAIL    REFER   MOUNTPOINT
 rpool                    3.90G    4.90G    77.5K   /rpool
 rpool/ROOT               3.01G    4.90G      19K   legacy
 rpool/ROOT/opensolaris   3.01G    4.90G    2.86G   /
 rpool/dump                274M    4.90G     274M   -
 rpool/export              125M    4.90G      21K   /export
 rpool/export/home         125M    4.90G      21K   /export/home
 rpool/export/home/taro    125M    4.90G     125M   /export/home/taro
 rpool/swap                512M    5.30G     101M   -
 tpool                      138K    7.79G    42.0K   /tpool
root@opensolaris:~$
```

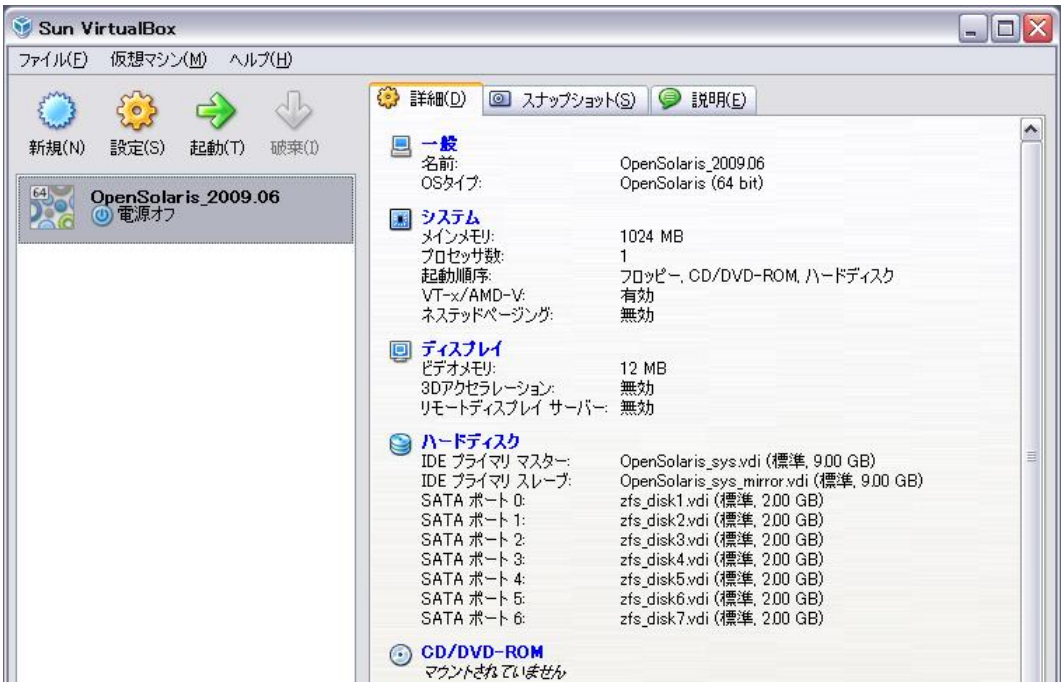
B-10. rpool のミラー化

rpool は Solaris でシステム領域を zfs とした場合に設定されるデフォルト名です。OpenSolaris では標準で利用可能ですが、Solaris 10 10/08 ではテキストインストーラを利用しなければいけません。rpool でサポートされるのは mirror および spare のみで、RAID Z、RAID Z2 はサポートされません。また、MBR(Grub 等で使用する領域)は対象外で、Grub データの書き込みなどの別途作業が必要になります。rpool のミラー化手順は、以下のようになります。

1. ミラー対象のディスクを追加
2. fdisk による Solaris パーティションの作成
3. パーティション情報の設定(ミラー元と同じ設定にする)
4. “zpool”コマンドによる rpool のミラー化
5. “installgrub”コマンドによる Grub のインストール

以上の作業を説明していきます。

(1) Sun xVM VirtualBox 起動画面です。ミラー用ディスクの追加を行います。ディスクの追加方法は、「B-2. 新規仮想ディスクの作成」を参照して、システム用ディスクと同じ容量で作成します。ここではミラー用ディスクの名前を“OpenSolaris_sys_mirror”としています。以下に追加後の図を示します。



(2) “zpool status rpool”コマンドを使用して、rpool の状態を確認します。

```
root@opensolaris:~$ zpool status rpool ⏎
  pool : rpool
 state: ONLINE
 scrub : none requested
config:

        NAME      STATE   READ WRITE  CKSUM
        rpool     ONLINE     0     0     0
          c7d0s0  ONLINE     0     0     0

errors: No known data errors
root@opensolaris:~$
```

(3) “format”コマンドを使用して、ミラー用ディスクの確認を行います。

```
root@opensolaris:~$ format ⏎
Searching for disks...done

AVAILABLE DISK SELECTIONS:
       0. c7d0 <DEFAULT cyl 1171 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@1,1/ide@0/cmdk@0,0
       1. c7d1 <DEFAULT cyl 1172 alt 2 hd 255 sec 63>
          /pci@0,0/pci-ide@1,1/ide@0/cmdk@1,0
       2. c9t0d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@0,0
       3. c9t1d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@1,0
       4. c9t2d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@2,0
       5. c9t3d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@3,0
       6. c9t4d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@4,0
       7. c9t5d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@5,0
       8. c9t6d0 <ATA-VBOX HARDDISK-1.0-2.00GB>
          /pci@0,0/pci8086,2829@d/disk@6,0
Specify disk (enter its number): 1 ⏎   ←ディスク番号を選択するとメニューが表示されます。
selecting c7d1
Controller working list found
[disk formatted, defect list found]
No Solaris fdisk partition found.←Fdisk パーティションが存在しないことを意味します。

FORMAT MENU:
        disk  - select a disk
        type  - select (define) a disk type
        partition - select (define) a partition table
        current - describe the current disk
        format  - format and analyze the disk
        fdisk  - run the fdisk program
        repair  - repair a defective sector
        show  - translate a disk address
        label  - write label to the disk
        analyze - surface analysis
        defect  - defect list management
        backup- search for backup labels
        verify  - read and display labels
        save  - save new disk/partition definitions
        volname - set 8-character volume name
        !<cmd>  - execute <cmd>, then return
        quit
format> fdisk ⏎
No fdisk table exists. The default partition for the disk is:
  ↑Fdisk パーティションが存在しない場合に表示されます。
    システムがディスクの一部のみを使用している場合は、n として、手動で作成します。
  a 100% "SOLARIS System" partition

Type "y" to accept the default partition,  otherwise type "n" to edit the
 partition table.
y ⏎
↑“fdisk”を実行すると、すべて Solaris パーティションで良いかを聞かれるので、“y”を応答します。
format> quit ⏎←“quit”で format を終了します。
```

(4) “format”→“＜番号＞”→“partition”→“print”で実行することでパーティション内部の情報を参照することができます。以下にディスク、“c7d0”と“c7d1”のパーティション情報を示します。

- **ディスク c7d0 の場合**

```
partition> print ⏎
Current partition table (unnamed):
Total disk cylinders available: 1171 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders         Size            Blocks
  0       root    wm       1 - 1170       8.96GB    (1170/0/0) 18796050
  1 unassigned    wm       0                 0       (0/0/0)          0
  2     backup    wu       0 - 1170       8.97GB    (1171/0/0) 18812115
  3 unassigned    wm       0                 0       (0/0/0)          0
  4 unassigned    wm       0                 0       (0/0/0)          0
  5 unassigned    wm       0                 0       (0/0/0)          0
  6 unassigned    wm       0                 0       (0/0/0)          0
  7 unassigned    wm       0                 0       (0/0/0)          0
  8     boot      wu       0 -     0      7.84MB    (1/0/0)      16065
  9 unassigned    wm       0                 0       (0/0/0)          0

partition>
```

- **ディスク c7d1 の場合**

```
partition> print ⏎
Current partition table (original):
Total disk cylinders available: 1171 + 2 (reserved cylinders)

Part      Tag    Flag     Cylinders         Size            Blocks
  0 unassigned    wm       0                 0       (0/0/0)          0
  1 unassigned    wm       0                 0       (0/0/0)          0
  2     backup    wu       0 - 1170       8.97GB    (1171/0/0) 18812115
  3 unassigned    wm       0                 0       (0/0/0)          0
  4 unassigned    wm       0                 0       (0/0/0)          0
  5 unassigned    wm       0                 0       (0/0/0)          0
  6 unassigned    wm       0                 0       (0/0/0)          0
  7 unassigned    wm       0                 0       (0/0/0)          0
  8     boot      wu       0 -     0      7.84MB    (1/0/0)      16065
  9 alternates    wm       1 -     2      15.69MB   (2/0/0)      32130

partition>
```

(5) “prtvtoc”コマンドで元ディスクのパーティション情報を取得し、“fmthard”コマンドでミラー先のパーティション情報を変更します。

```
root@opensolaris:~$ prtvtoc /dev/rdsk/c7d0s0 > /tmp/prtvtoc.txt ⏎
root@opensolaris:~$ fmthard -s /tmp/prtvtoc.txt /dev/rdsk/c7d1s0 ⏎
fmthard:  New volume table of contents now in place.
root@opensolaris:~$
```

● (4)の手順で、ディスク“07d1s0”のパーティション情報が変更されたことを確認します。

```
partition> print ⏎
Current partition table (original):
Total disk cylinders available: 1171 + 2 (reserved cylinders)
```

Part	Tag	Flag	Cylinders	Size	Blocks	
0	**root**	**wm**	**1 - 1170**	**8.96GB**	**(1170/0/0)**	**18796050**
1	unassigned	wu	0	0	(0/0/0)	0
2	backup	wu	0 - 1170	8.97GB	(1171/0/0)	18812115
3	unassigned	wu	0	0	(0/0/0)	0
4	unassigned	wu	0	0	(0/0/0)	0
5	unassigned	wu	0	0	(0/0/0)	0
6	unassigned	wu	0	0	(0/0/0)	0
7	unassigned	wu	0	0	(0/0/0)	0
8	boot	wu	0 - 0	7.84MB	(1/0/0)	16065
9	**unassigned**	**wu**	**0**	**0**	**(0/0/0)**	**0**

```
partition>
```

(6) “zpool attatch -f rpool <元ディスク> <ミラー先>”コマンドでミラー化を行います。コマンドを実行後、ディスクのシンク(同期化)が行われます。

```
root@opensolaris:~$ zpool attach rpool c7d0s0 c7d1s0 ⏎
invalid vdev specification
use '-f' to override the following errors:
/dev/dsk/c7d1s0 overlaps with /dev/dsk/c7d1s2
root@opensolaris:~$ zpool attach -f rpool c7d0s0 c7d1s0 ⏎
Please be sure to invoke installgrub(1M) to make 'c7d1s0' bootable.
root@opensolaris:~$ zpool status rpool ⏎
  pool : rpool
 state: ONLINE
status: One or more devices is currently being resilvered.  The pool will
   continue to function, possibly in a degraded state.
action: Wait for the resilver to complete.    【注意】
 scrub : resilver in progress for 0h0m, 13.53% done, 0h4m to go
config:

        NAME        STATE     READ  WRITE  CKSUM
        rpool       ONLINE       0      0      0
          mirror    ONLINE       0      0      0
            c7d0s0  ONLINE       0      0      0
            c7d1s0  ONLINE       0      0      0  482M resilvered

errors: No known data errors
root@opensolaris:~$
```

【注意】この場合、再同期化が完了するまでしばらく待ちます。完了しないまま作業した場合、エラーになりミラー用ディスクでのブートができません。再同期化が完了した時の図を以下に、エラー時の図を(12)に示します。

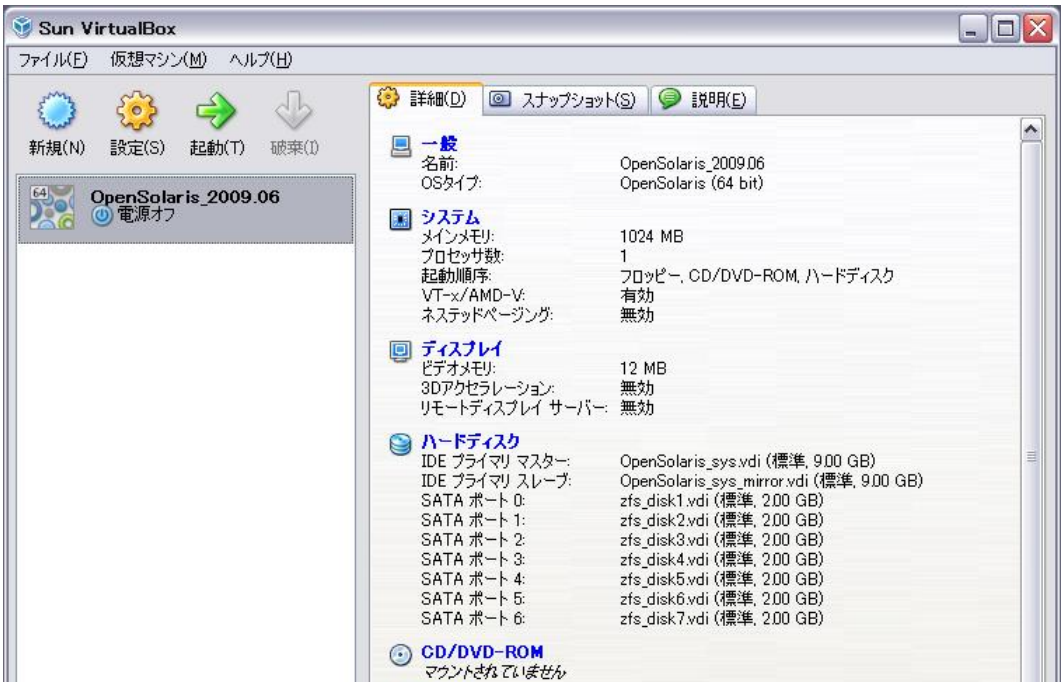
● 再同期化の完了

```
root@opensolaris:~$ zpool status rpool ⏎
  pool : rpool
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
  attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
  using 'zpool clear' or replace the device with 'zpool replace'.
  see : http://www.sun.com/msg/ZFS-8000-9P
 scrub : resilver completed after 0h0m with 0 errors on Thu Oct 15 15:18:44 2009
config:

        NAME        STATE     READ WRITE  CKSUM
        rpool       ONLINE       0     0     0
          mirror    ONLINE       0     0     0
            c7d0s0  ONLINE       0     0    12  14.9M resilvered
            c7d1s0  ONLINE       0     0     0

errors: No known data errors
root@opensolaris:~$
```

(7) “installgrub <stage1> <stage2> <dev>”コマンドを使用して、ミラー用ディスクへの Grub のインストールを行います。

```
root@opensolaris:~$ installgrub /boot/grub/stage1 /boot/grub/stage2
/dev/rdsk/c7d1s0 ⏎
stage1 written to partition 0 sector 0 (abs 16065)
stage2 written to partition 0, 271 sectors starting at 50 (abs 16115)
root@opensolaris:~$
```

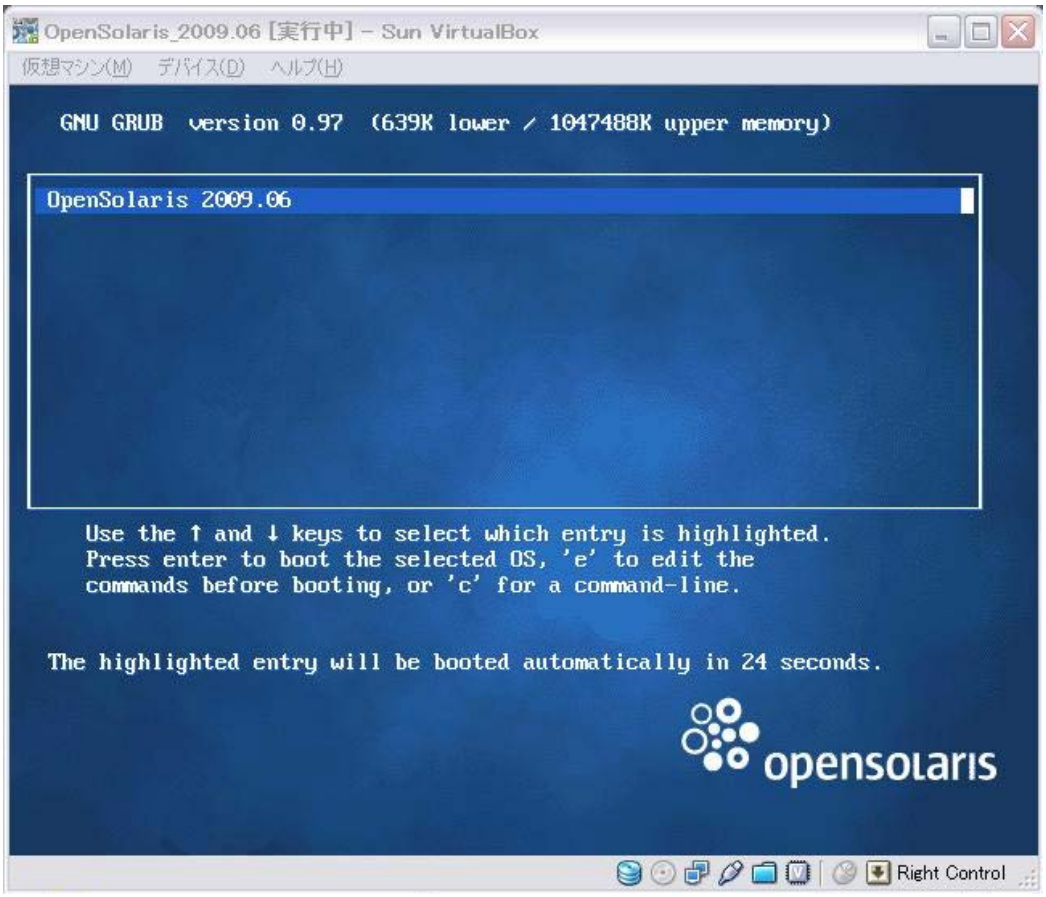
(8) 一度シャットダウンして、Sun xVM VirtualBox 起動画面の“設定”ボタンをクリックします。



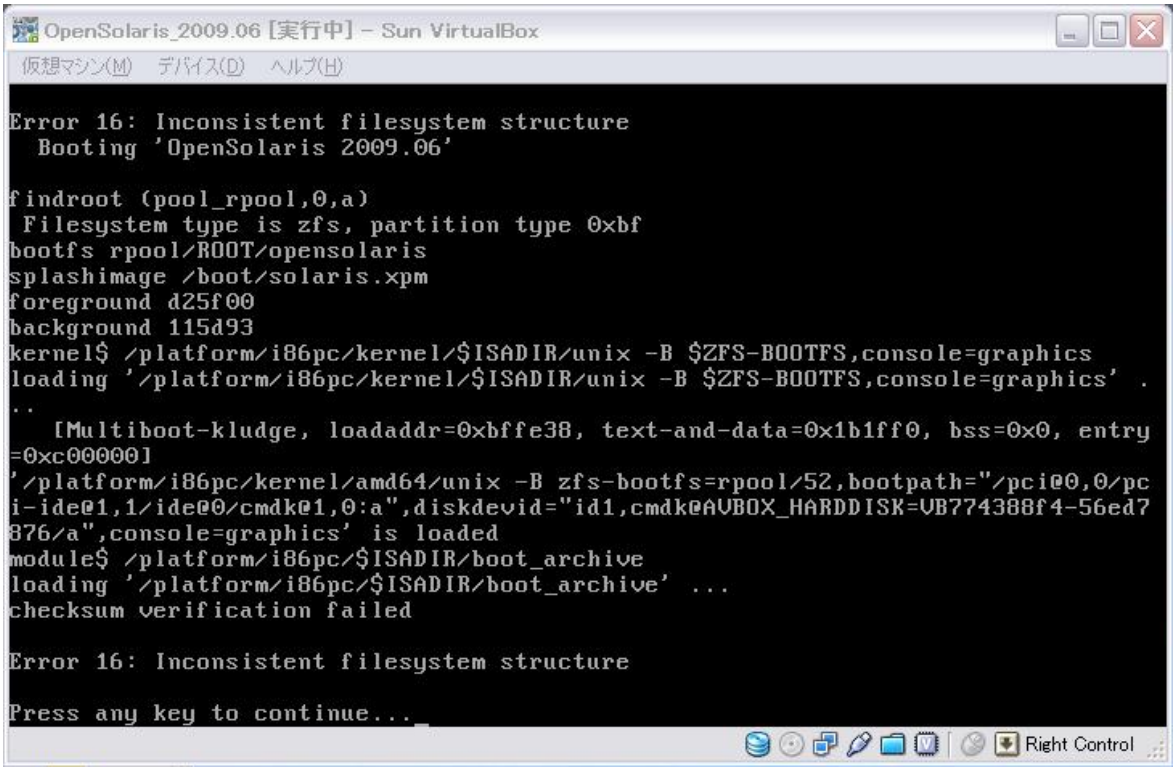
(9) ハードディスクの設定画面です。“ ”ボタンでシステムディスクを除去します。“OK”をクリックします。これは実際にディスク装置を取り外したことと同じです。

(10) Sun xVM VirtualBox 起動画面です。ハードディスクの構成を確認して、ミラー用のディスクでの起動になっていることを確認します。

(11) Grub 起動画面です。そのままリターンします。



(12) (6)での再同期化が完了しないまま作業を行い、Grub 起動画面でリターンした場合の画面です。



- このままリターンした場合、次のように文字が橙色になった Grub 画面が表示されます。

(13) 正常に起動した後、“zpool status rpool”コマンドで確認します。

```
root@opensolaris:~$ zpool status rpool ⏎
  pool : rpool
 state: DEGRADED
status: One or more devices could not be opened.  Sufficient replicas exist for
   the pool to continue functioning in a degraded state.
action: Attach the missing device and online it using 'zpool online'.
   see : http://www.sun.com/msg/ZFS-8000-2Q
 scrub : none requested
config:

        NAME        STATE     READ WRITE CKSUM
        rpool       DEGRADED  0     0     0
          mirror    DEGRADED  0     0     0
            c7d0s0  UNAVAIL   0     0     0  cannot open
            c7d1s0  ONLINE    0     0     0

errors: No known data errors
root@opensolaris:~$
```

(14) システムディスクを元に戻した場合、以下のように表示されます。

```
root@opensolaris:~$ zpool status rpool ⏎
  pool : rpool
 state: ONLINE
status: One or more devices has experienced an unrecoverable error.  An
    attempt was made to correct the error.  Applications are unaffected.
action: Determine if the device needs to be replaced, and clear the errors
    using 'zpool clear' or replace the device with 'zpool replace'.
   see : http://www.sun.com/msg/ZFS-8000-9P
 scrub : resilver completed after 0h0m with 0 errors on Thu Oct 15 15:18:44 2009
config:

        NAME          STATE     READ WRITE CKSUM
        rpool         ONLINE       0     0     0
          mirror      ONLINE       0     0     0
            c7d0s0    ONLINE       0     0    12  14.9M resilvered
            c7d1s0    ONLINE       0     0     0

errors: No known data errors
root@opensolaris:~$
```

付録 C. RAID 概要

RAID(Redundant Arrays of Inexpensive【備考】Disks)は複数のハードディスクをまとめて仮想的な一つのディスクとして管理することで、安全に、より高速にデータを処理し、故障時にシステムを停止することなくサービスを提供することができる技術です。

RAID には容量、速度、安全性の要素を組み合わせた特徴をもつ構成があります。これらは“RAID 0”や“RAID 1”のように番号を末尾に付けた名称で呼ばれています。これを RAID レベルといいます。

次に RAID の各レベルの特徴を説明します。

RAID 0

ストライピングと呼ばれる記録方式の RAID です。仮想的なディスクの容量を大きくすることができます。また、1 台のディスクにアクセスが集中するのを防ぎアクセスを分散することで高速化を図る RAID レベルです。

例として 2 台のディスクを RAID 0 で構成すると次のようになります。

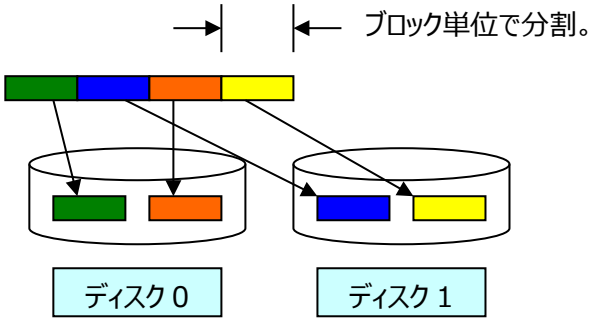


図 C-1. RAID 0 の動き

最初のブロック(緑)をディスク 0、二番目のブロック(青)をディスク 1、三番目のブロック（橙）をディスク 0、四番目のブロック(黄)をディスク 1 に書き込み(または読み込み)します。容量はディスク 2 台分になります。また交互にアクセスすることで前のブロックのアクセス完了を待たずに次のブロックをアクセスできます。これで高速化を図ることができます。

利点は容量を大きくすることができること、および高速化です。一方、弱点は 1 台のディスクが故障すると全部の仮想ディスクがダメになることです。

【備考】 Independent とも言われています。元々、低価格の沢山のディスクを一つにまとめる の意でした。

RAID 1

ミラーリングと呼ばれる記録方式の RAID です。1 台のディスクの全く同じクローンを持つことで片方のディスクが故障しても処理を続けることができます。安全性は高まりますが、2 台のディスクを使用しても容量は 1 台分のディスクと同じで、経済的ではありません。

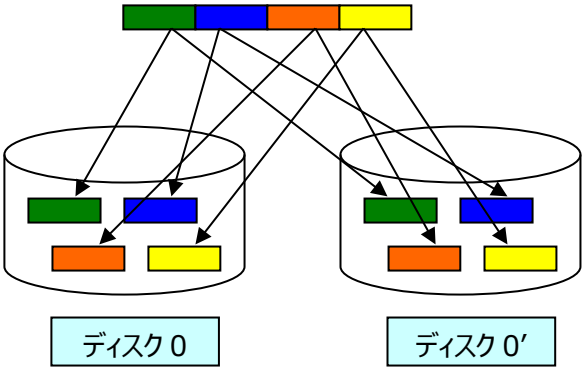


図 C-2. RAID 1 の動き

RAID 1E

RAID 1E はストライプされたミラーです。RAID 1 に類似しています。データのミラーリングとストライピングの両方が行われます。これにより、より多くのディスクドライブを構成することができます。

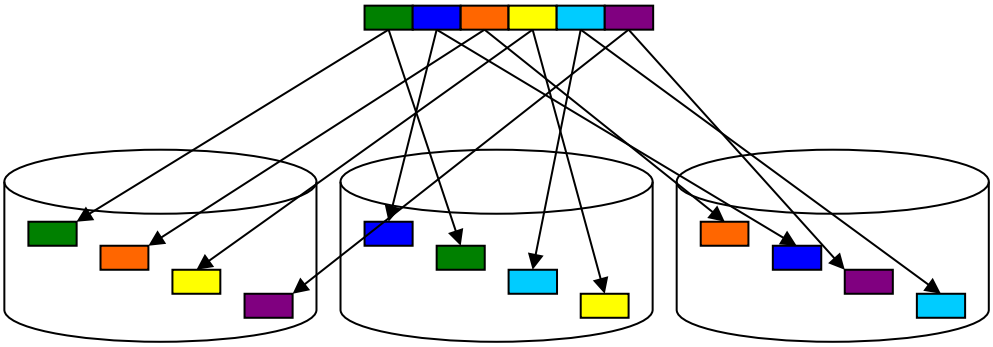


図 C-3. RAID 1E の動き

RAID 10

RAID 1 のミラーリングと RAID 0 のストライピングを組み合わせたものです。RAID 1+0 も同じ意味です。RAID 0 だけでは 1 台のディスクが故障すると全部がダメになります。RAID 10 にすることで冗長性が高くなります。図は省略させていただきます。

RAID 2

エラーを修復するための冗長コードをデータとともに複数のディスクに記録します。冗長コードが ECC(Error Correcting Code)、データを分割する単位がビット、またはバイトのため、容量、性能面で課題の多い RAID 方式です。図は省略させていただきます。

RAID 3

データをビットまたはバイトで分割して全てのディスクに一斉にアクセスする RAID です。パリティのディスクを別に持っています。データディスクが 1 台故障した場合はパリティディスクより復元できます。2 台故障するとデータは失われます。容量は 2/3 になります。ビット、またはバイトのパリティを生成し、その単位でディスクをアクセスするため性能は悪くなります。

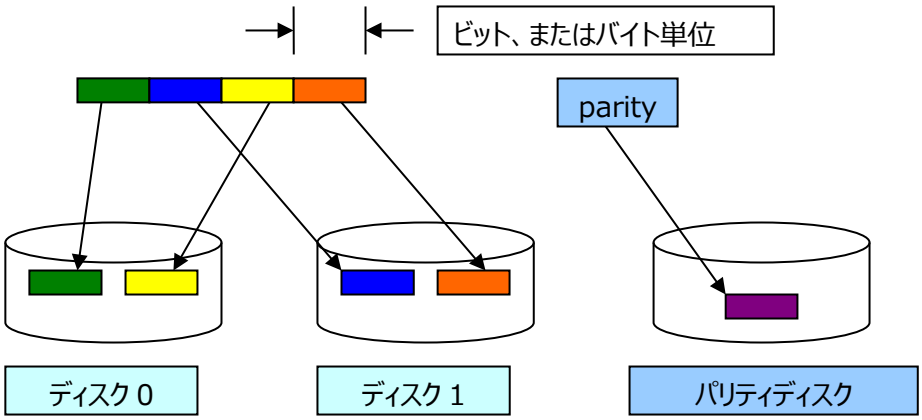


図 C-4. RAID 3 の動き

RAID 4

RAID 3 はビット、またはバイト単位でデータを分割してパリティを生成し、それらをディスクに一斉に書き込み/読み取りしていました。RAID 4 ではデータの分割単位を指定したブロック単位で操作することで高速化を図っています。これは RAID 0 にパリティディスクの仕組みを追加したものと考えて下さい。

図は RAID 3 と同じですので省略させていただきます。

RAID 5

RAID 3 における I/O 効率の悪さや、RAID 4 におけるパリティ書き込み時の負荷という欠点を改良し、パリティを分散した RAID が RAID 5 です。どのディスクが故障してもデータを復元できます。最低 3 台のディスクが必要です。

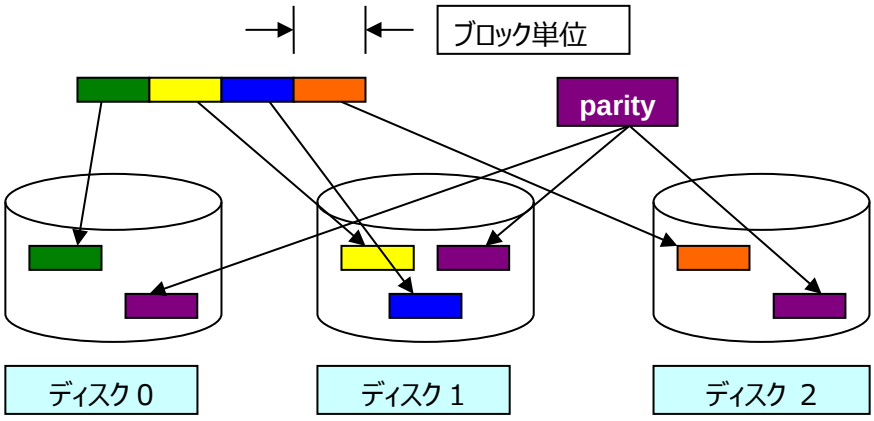


図 C-5. RAID 5 の動き

RAID 5EE

RAID 5EE はホットスペースと呼ばれます。RAID 5 に類似しています。RAID 5EE には分散スペアのドライブが含まれるため、最低 4 台の構成になります。ホットスペアとは異なり、分散スペアではディスクドライブ間で均等にデータ、およびパリティがストライピングされます。この分散スペアにより、ディスクドライブの障害後、アレイの再構成時間が短くなります。RAID 5EE ではデータが保護され、リード/ライトの速度が向上します。しかし、パリティ、およびスペアのデータ用に領域が使用されるためディスクドライブ 2 台分の容量が減少します。

図は省略させていただきます(データブロック分散の順序が不明のため)。

RAID 50

RAID 50 は 2 つ以上の RAID 5 アレイとして設定されたディスクドライブで構成されます。最小のディスク台数は 6 台です。両方の RAID 5 アレイの全てのディスクドライブ間でデータ、およびパリティデータがストライピングされます。

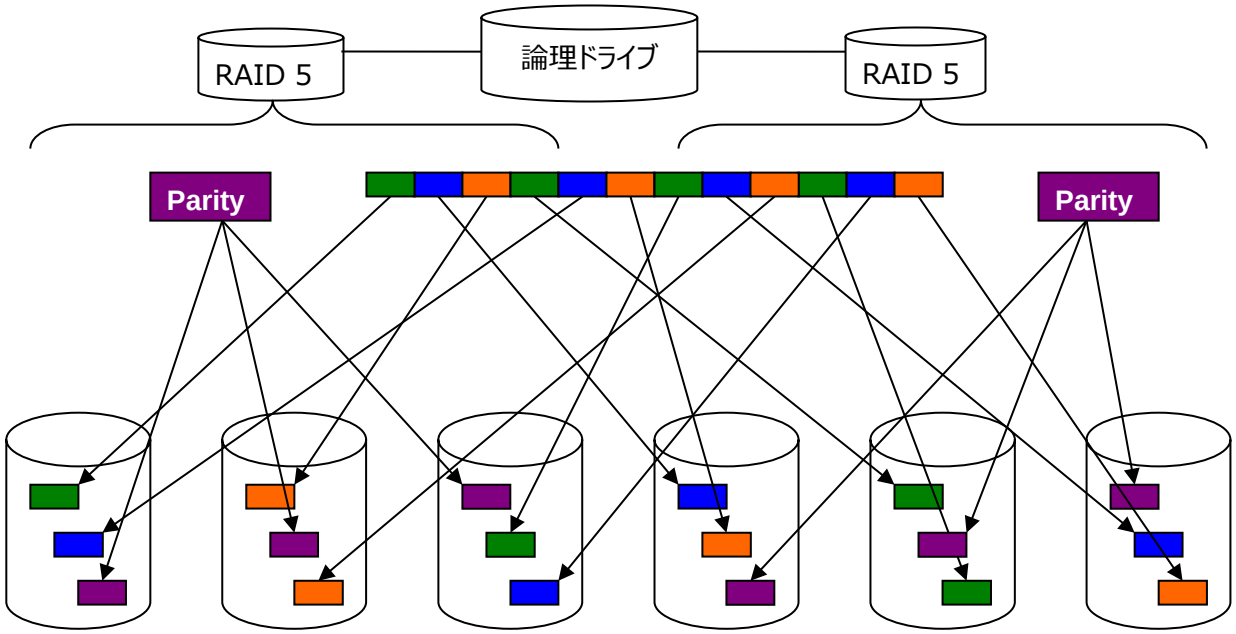


図 C-6. RAID 50 の動き

RAID 6

RAID 5 の改良版です。一つのデータブロックあたり 2 つのパリティを生成します。このことで、2 台のディスクが同時に故障してもデータが復元可能な RAID です。パリティが倍になることで負荷が大きくなり、またディスク容量が減ります。

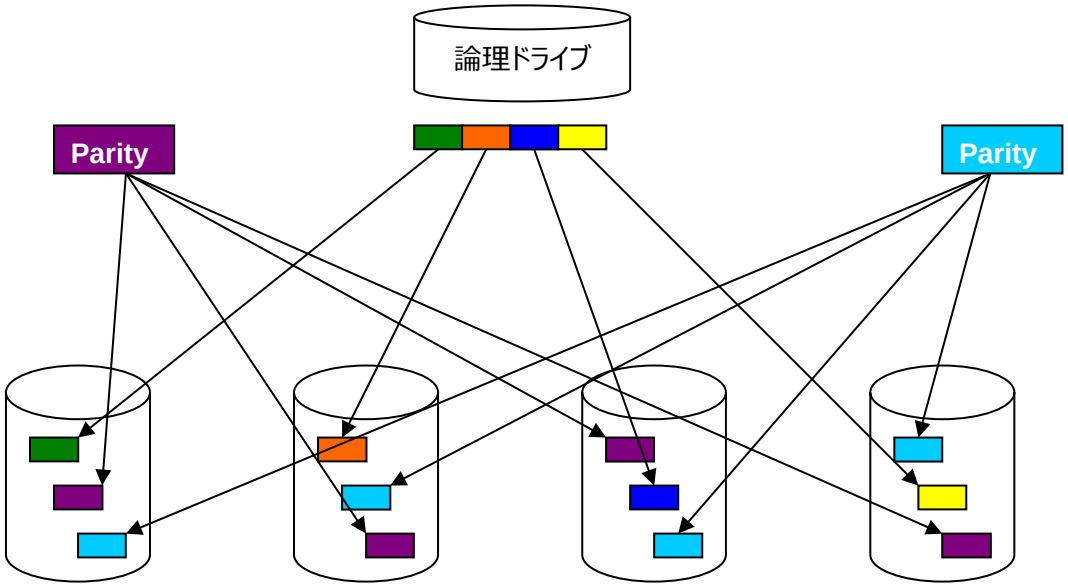


図 C-7. RAID 6 の動き

RAID 60

RAID 50 と同様にデュアルドライブ故障保護と呼ばれます。8 台のディスクドライブで構成され、2 台以上の RAID 6 が含まれます。両方の RAID 6 内の全てのディスクドライブ間で、データ、および 2 セットのパリティがストライピングされます。

2 セットのパリティによってデータ保護が強化され、ストライピングによって性能が向上します。

図は RAID 50 に RAID 6 を組み合わせたものになります。図は省略させていただきます。

RAID Z

データとパリティを複数のディスクに格納する仮想デバイスで、RAID 5 に似ています。

RAID 5 は、データを復旧させるのに必要なパリティデータを複数のディスクに分散して配置する RAID です。ここで紹介する RAID Z はその RAID-5 の安全性と速度を向上させた、Sun による拡張版です。

また、RAID Z は、データの保存とそのデータに対応するパリティを保存するドライブが別になるように動的に割り当てられています。そのため、2 台のドライブからパリティチェックを行うことができます。RAID Z は、最低 2 台のディスクが必要になります

RAID Z では、RAID 5 の以下の問題点を解消しました。

- 動的なストライプ幅　　：各論理ブロックは自身のストライプ
- 全ての書き込みが完全ストライプ書き込み：read-modify-write の排除
- RAID-5 の“Write-Hole”を解決　：ソフトウェアで解決(NVRAM 不要)

以下に概念図を示します。このように動いているかどうかは不明です。

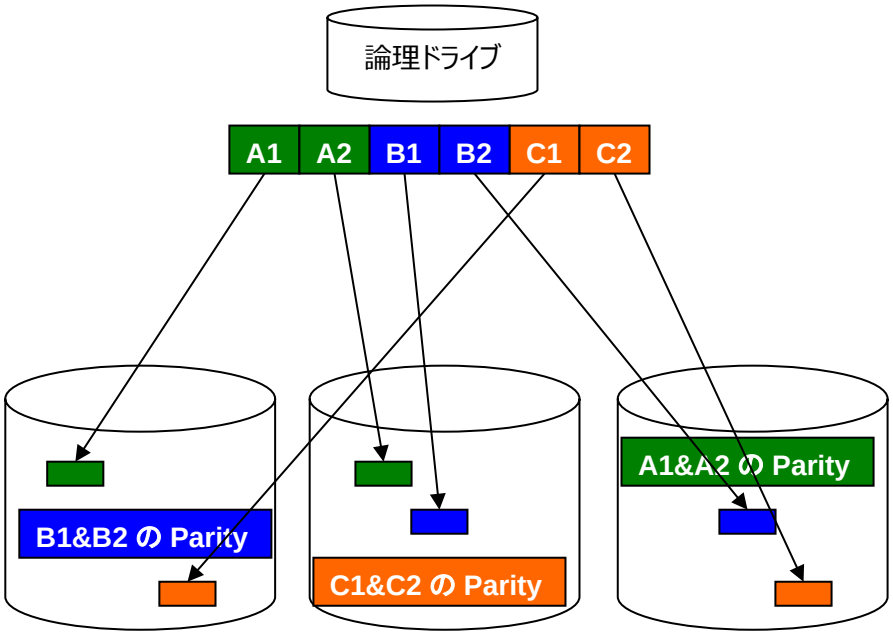


図 C-8.　RAID Z の動き

RAID-Z2

データとパリティを複数のディスクに格納する仮想デバイスで、RAID 6 に似ています。

RAID Z2 では、演算式の異なる 2 つのパリティをそれぞれ 2 つのドライブに記録することにより、1 つまたは 2 つのデバイスで障害が発生しても、データを失うことなく処理を続行できることを意味します。RAID Z2 は、最低 3 台のディスクが必要になります。

以下に概念図を示します。このように動いているかどうかは不明です。

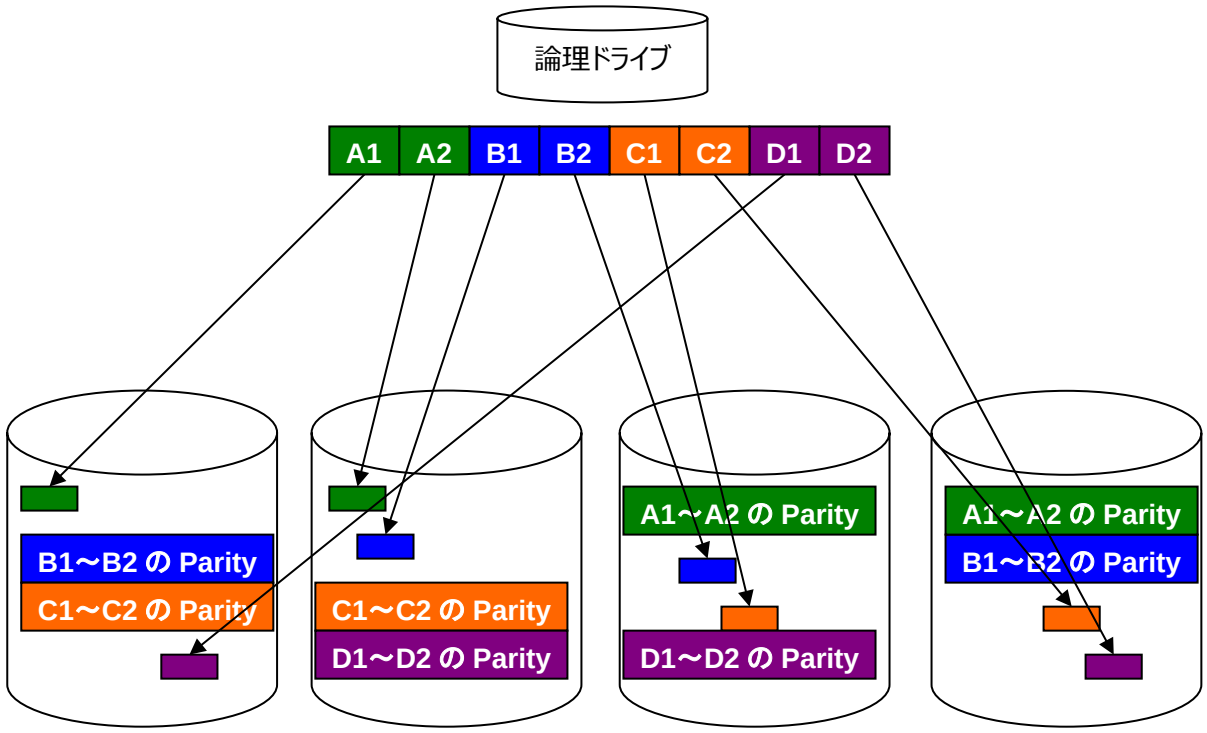


図 C-9. RAID Z2 の動き

付録 D. 参考文献と WEB

D-1. 参考文献

表 D-1-1. 参考文献

項目	文献
Solaris 10 10/09 ZFS	Solaris ZFS 管理ガイド 819-6260-16 Sun Microsystems, Inc
ZFS 活用	ZFS 仮想化されたファイルシステムの徹底活用 長原宏治、佐藤通敏、今井悟志、加藤久慶著 (株)アスキー・メディアワークス
Windows の NTFS	INSIDE Windows NT ファイルシステム Helen Custer 著 小畑喜一、五十嵐宰、大西照代訳 アスキー出版局
UNIX カーネル	UNIX カーネルの設計 Maurice J.Bach 著 坂本文、多田好克、村井純訳 共立出版
数値計算ガイド	Sun ONE Studio 8 数値計算ガイド 817-2921-10 Sun Microsystems,Inc
コンピュータ全般	ヘネシー&パターソン コンピュータ・アーキテクチャ John L.Hennessy & David A.Patterson 富田眞治、村上和彰、新寛治男訳 日経 BP 社

D-2. 参考 WEB

表 D-2-1. 参考 WEB

項目	URL
ファイルシステム【備考 1】	https://ja.wikipedia.org/wiki/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%82%B7%E3%82%B9%E3%83%86%E3%83%A0
ZFS ファイルシステム 1	https://docs.oracle.com/cd/E26924_01/html/E25824/zfsover-1.html
ZFS ファイルシステム 2	https://docs.oracle.com/cd/E26924_01/html/E25824/zfsover-2.html
ZFS を USB で使う	https://solaris-user.com/solaris_beans/usb_harddisk.html
ZFS を Sun Virtual Box で使う	https://docs.oracle.com/cd/E55843_01/html/E54238/integration__oracle_storage_connect__oracle_virtual_machine_storage_connect_pl.html
Solaris ZFS その 1	https://www.atmarkit.co.jp/ait/articles/0903/13/news125.html
Solaris ZFS その 2	https://www.atmarkit.co.jp/ait/articles/0903/13/news125_2.html
Solaris ZFS その 3	https://www.atmarkit.co.jp/ait/articles/0903/13/news125_3.html
zpool ログ性能【備考 2】	https://docs.oracle.com/cd/E19253-01/819-6260/gfgaa/index.html
RAID【備考 3】	https://ja.wikipedia.org/wiki/RAID
ZFS ファイルシステム紹介	https://www.slideshare.net/satorumiyazaki/hbstudy12-zfs
ZFS ディスク形式(PDF)	ZFS On-Disk Specification (Draft) Sun Microsystemd,Inc https://www.dubeyko.com/development/FileSystems/ZFS/ondiskformat0822.pdf
mkfile で作成したディスクも ZFS 構成可能!!	https://lildude.co.uk/zfs-cheatsheet
無限大数の彼方へ	http://www.sf.airnet.ne.jp/~ts/language/largenumber.html

【備考 1】「ファイルシステムはこんなにたくさんの種類があるのですか!?」と感動する程に詳細に解説された Wikipedia です。本書ではごく一部しか取り上げることができませんでしたが、各位、必見の WEB です。
【備考 2】ZFS の slog、clog を NVRAM に設定することで、ランダム書き込みの性能を飛躍的に向上させるものです。
【備考 3】RAID 基礎からパリティ計算の方法、アルゴリズム、手順などわかりやすく記述されています。

END REPORT