

(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号  
特開2004-192105  
(P2004-192105A)

(43) 公開日 平成16年7月8日(2004.7.8)

(51) Int.Cl.<sup>7</sup>  
G06F 3/06

F I  
G06F 3/06 3 O 1 A  
G06F 3/06 5 4 O

テーマコード (参考)  
5 B O 6 5

審査請求 未請求 請求項の数 14 O L (全 22 頁)

(21) 出願番号	特願2002-356477 (P2002-356477)	(71) 出願人	000005108 株式会社日立製作所 東京都千代田区神田駿河台四丁目6番地
(22) 出願日	平成14年12月9日 (2002.12.9)	(74) 代理人	100068504 弁理士 小川 勝男
		(74) 代理人	100086656 弁理士 田中 恭助
		(74) 代理人	100094352 弁理士 佐々木 孝
		(72) 発明者	味松 康行 神奈川県川崎市麻生区王禅寺1099番地 株式会社日立製作所システム開発研究所 内

最終頁に続く

(54) 【発明の名称】 記憶装置の接続装置およびそれを含むコンピュータシステム

(57) 【要約】

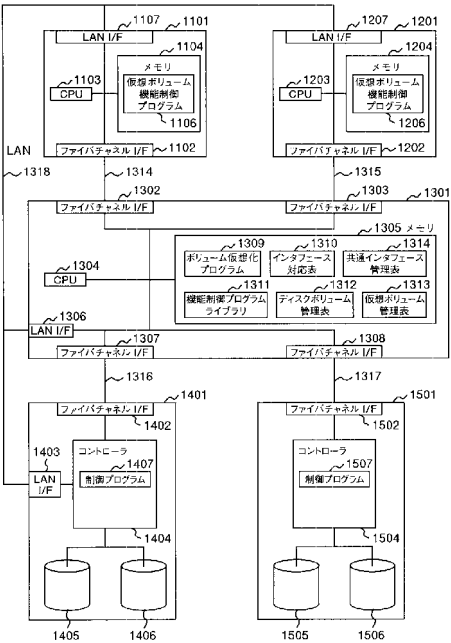
【課題】機能制御のインタフェースが異なる複数の外部記憶装置が混在する環境において、記憶装置の機能をコンピュータから制御する場合に、制御の対象となる記憶装置の種類を識別し種類に応じた処理を実行する必要があるため、制御が複雑化する問題がある。

【解決手段】記憶装置の機能を制御するための共通インタフェースを設ける。共通インタフェースを提供する装置は、ホストコンピュータが認識する記憶領域と、記憶装置が提供する記憶領域の対応関係を管理し、機能制御指示の対象となる記憶領域と、それを提供している記憶装置を対応付ける。機能制御指示の対象となる記憶領域を提供する記憶装置の種類を識別し、その装置に固有の機能制御インタフェースを介して機能制御を指示する。

【選択図】 図1

図 1

全体構成



**【特許請求の範囲】****【請求項 1】**

1 つ以上のホストコンピュータおよび 1 つ以上の外部記憶装置とネットワークで接続された記憶装置の接続装置であり、

前記ホストコンピュータから前記外部記憶に関する機能を制御する指示を受信する第 1 の手段と、

受信した指示から、制御対象となる外部記憶装置を識別する第 2 の手段と、

受信した指示から、外部記憶装置に指示する内容を選択する第 3 の手段と、

識別した制御対象および選択した指示内容に応じて、外部記憶装置に指示を送る方法を選択する第 4 の手段と、

選択した方法により、外部記憶装置に指示を送る第 5 の手段と、

を備えることを特徴とする記憶装置の接続装置。

10

**【請求項 2】**

前記第 2 の手段から参照され、ホストコンピュータが認識する記憶領域と外部記憶装置が提供する記憶領域の対応関係を記憶する第 6 の手段と、

前記第 3 の手段に応じて受信した指示から、ホストコンピュータが指定した制御対象に対応する、外部記憶装置が提供する 1 つ以上の記憶領域を識別する第 7 の手段と、

を更に備えることを特徴とする請求項 1 記載の記憶装置の接続装置。

**【請求項 3】**

外部記憶装置から状態情報を取得する第 8 の手段と、

外部記憶装置から取得した複数の状態情報から、ホストコンピュータが指定した制御対象の状態を作成する第 9 の手段と、

前記ホストコンピュータに前記状態情報を送信する第 10 の手段と、

を更に備えることを特徴とする請求項 1 記載の記憶装置の接続装置。

20

**【請求項 4】**

互いにプロトコルの異なる複数の前記第 1 の手段

を備えることを特徴とする請求項 1 記載の記憶装置の接続装置。

**【請求項 5】**

ホストコンピュータから指定された制御対象および制御内容に応じた、外部記憶装置に指示を送る複数の方法を記憶する第 11 の手段と、

外部記憶装置への指示の失敗を識別する第 12 の手段と、

外部記憶装置への指示が失敗した場合に、指示を送る別の手段を選択する第 13 の手段と、

を更に備えることを特徴とする請求項 1 記載の記憶装置の接続装置。

30

**【請求項 6】**

外部記憶装置への指示が失敗した場合に、その指示に対応する、エラー処理内容を記憶する第 14 の手段と、

外部記憶装置への指示が失敗した場合に、エラー処理が必要な制御対象を識別する第 15 の手段と、

エラー処理が必要であると識別した制御対象に対するエラー処理を実行する第 16 の手段と、

を更に備えることを特徴とする請求項 5 記載の記憶装置の接続装置。

40

**【請求項 7】**

制御対象ごとに、制御を許可する対象を記憶する第 17 の手段と、

受信した指示の、通信プロトコルの送信元に関する情報に基づいて、制御が許可されているか判別する第 18 の手段と、

受信した指示が、許可された制御か判別する第 19 の手段と、

を更に備えることを特徴とする請求項 1 記載の記憶装置の接続装置。

**【請求項 8】**

受信した指示の、指示内容に含まれる送信元の識別子に基づいて、制御が許可されている

50

か判別する第 20 の手段と、  
を備えることを特徴とする請求項 7 記載の記憶装置の接続装置。

【請求項 9】

1 つ以上のホストコンピュータおよび 1 つ以上の外部記憶装置と接続されたネットワークスイッチであり、  
ホストコンピュータから外部記憶に関する機能を制御する指示を受信する手段と、  
受信した指示から、制御対象となる外部記憶装置を識別する手段と、  
受信した指示から、外部記憶装置に指示する内容を選択する手段と、  
識別した制御対象および選択した指示内容に応じて、外部記憶装置に指示を送る方法を選択する手段と、  
選択した手段により、外部記憶装置に指示を送る手段と、  
を備えることを特徴とするネットワークスイッチ。

10

【請求項 10】

インテリジェント機能を持ちプロトコルの異なる 2 以上の記憶装置と、前記記憶装置に共通インタフェースで前記記憶装置の機能を制御する指示を出力する 1 以上のホストコンピュータとに接続され、前記ホストコンピュータからの仮想ボリューム識別子を含む前記指示を共通インタフェースとして受信する第 1 の手段と、前記指示を受け、前記仮想ボリュームを構成するディスクボリュームを求め前記ディスクボリュームに固有なプロトコルに変換して各記憶装置に対し機能を制御する指示を発行する第 2 の手段とを備えたことを特徴とする記憶装置の接続装置。

20

【請求項 11】

インテリジェント機能を持ちプロトコルの異なる 2 以上の記憶装置と、前記記憶装置に共通インタフェースで前記記憶装置の機能を制御する指示を出力する 1 以上のホストコンピュータと、前記記憶装置とホストコンピュータに接続され、前記ホストコンピュータからの前記指示を共通インタフェースとして受信する第 1 の手段と、前記指示を前記記憶装置の指示を受けるプロトコルに変換して各記憶装置に対し発行する第 2 の手段とを有する記憶装置の接続装置とを備えたことを特徴とするコンピュータシステム。

【請求項 12】

前記記憶装置の接続装置は前記記憶装置の機種ごとに制御される機能の制御のための API を記憶するテーブルを持ち、前記第 2 の手段は前記テーブルを参照することを特徴とする請求項 11 記載のコンピュータシステム。

30

【請求項 13】

1 以上のホストコンピュータとインタフェースの異なる 2 以上の記憶装置を含む記憶装置群に接続された記憶装置の接続装置における接続制御方法であって、  
前記ホストコンピュータから共通インタフェースに対し前記記憶装置の機能の制御を指示するものであり制御機能と制御対象である仮想ボリュームの情報を含む制御ブロックを受信し、  
特定の機能制御であることを判別し、  
制御対象となる仮想ボリュームを特定し、  
仮想ボリュームを構成するディスクボリュームを特定し、  
前記記憶装置のディスクボリュームに固有な制御 API を呼び出す際に引数となる API を含む制御ブロックを作成し、  
前記記憶装置に固有なインタフェースと前記制御ブロックから機能を制御する API を呼び出して機能制御を指示することを特徴とする接続制御方法。

40

【請求項 14】

1 以上のホストコンピュータとインタフェースの異なる 2 以上の記憶装置を含む記憶装置群に接続された記憶装置の接続装置における接続制御のため前記接続装置を、  
前記ホストコンピュータから共通インタフェースに対し前記記憶装置の機能の制御を指示するものであり制御機能と制御対象である仮想ボリュームの情報を含む制御ブロックを受信する手段、

50

特定の機能制御であることを判別する手段、  
制御対象となる仮想ボリュームを特定する手段、  
仮想ボリュームを構成するディスクボリュームを特定する手段、  
前記記憶装置のディスクボリュームに固有な制御APIを呼び出す際に引数となるAPIを含む制御ブロックを作成する手段、および  
前記記憶装置に固有なインタフェースと前記制御ブロックから機能を制御するAPIを呼び出して機能制御を指示する手段、  
として機能させるための接続制御プログラム。

【発明の詳細な説明】

【0001】

10

【発明の属する技術分野】

本発明は、ホストコンピュータと複数の記憶装置と接続する技術に関し、特に、ディスクアレイなどの記憶装置がユーザに提供する様々な機能の制御に関する。

【0002】

【従来の技術】

近年、ディスクアレイに代表されるコンピュータの記憶装置のインテリジェント化が進んでいる。例えば、ディスクアレイ装置が提供するミラー機能は、ホストコンピュータがディスクアレイ装置内の記憶領域のデータを更新すると、あらかじめ決められた別の記憶領域にもその更新データを自動的に書き込む。ミラーの停止を指示することにより、その時点での記憶データを保存し、バックアップ等の用途に使用することができる。更新データの別領域への書き込みはディスクアレイ装置内のコントローラによって実行されるため、ホストコンピュータに負荷がかからず、高速な入出力処理が可能となる（例えば、特許文献1参照）。

20

【0003】

一方、システム内の記憶装置の数が増え、種類の異なる記憶装置が混在するということが起こっている。即ち、異なるプロトコルを持った記憶装置が混在することになる。ここで、機能とは記憶装置をミラーすることやスナップショットを取ることを言い、機能制御とはホストコンピュータが記憶装置にこれらの機能を実現させるよう制御することである。

【0004】

30

ホストコンピュータは、ディスクアレイが提供する機能制御インタフェースにアクセスし、あらかじめ定義された記憶装置に固有のプロトコルで指示を送ることにより、機能を制御する。機能制御インタフェースは、In-bandのSCSIコマンドや、LAN経由のTCP/IP通信など、さまざまな形態で実現できる（例えば、特許文献2参照）。

【0005】

また、複数のホストコンピュータと記憶装置をネットワークで接続するストレージエリアネットワーク（SAN: Storage Area Network）の普及が進んでいる。SANはファイバチャネルに代表されるようなデータ入出力専用の高速ネットワークであり、コンピュータの入出力性能を向上することができる。さらに、SANで接続されたコンピュータ群が1つの記憶装置を共用したり、逆に1つのコンピュータが複数の記憶装置にアクセスしたりできるなど、多様な記憶領域の利用が可能となる。最近では、記憶装置/記憶領域の共有を実現するだけでなく、記憶装置が提供する記憶領域（ディスクボリューム）を結合、あるいは分割して仮想的な記憶領域（仮想ボリューム）をホストコンピュータに提供するストレージバーチャライゼーション（Storage Virtualization）の技術も導入されはじめ、より柔軟な記憶領域の管理が可能となっている。

40

【0006】

【特許文献1】

米国特許第5,845,295号明細書

【特許文献2】

50

米国特許第 5 , 8 6 7 , 7 3 6 号明細書

【 0 0 0 7 】

【 発明が解決しようとする課題 】

SAN 環境など、複数の異なる種類の記憶装置が混在するコンピュータシステムにおいて、記憶装置が提供するインテリジェントな機能を利用する場合、ホストコンピュータは制御対象となるディスクアレイの種類を識別し、各記憶装置に固有のインタフェースで機能制御の指示を送る必要がある。同じ機能の制御でも装置毎に異なる制御プログラムをホストコンピュータ内に用意して使い分けなければならない、制御が複雑化する。

【 0 0 0 8 】

また、ストレージバーチャライゼーションにおいては、記憶装置が提供する機能を利用して仮想ボリューム単位でインテリジェントな機能を実現することが考えられるが、その際、仮想ボリュームが複数のディスクボリュームから構成されていると、ディスクボリューム毎に機能の制御を指示する必要がある。そのため、ホストコンピュータは仮想ボリュームの実際の記憶領域を提供する装置を識別し、各装置固有の制御インタフェースを使用する必要があるため制御が複雑化する。

【 0 0 0 9 】

本発明の目的は、異なる種類の記憶装置のさまざまな機能を制御するために、装置の種類に依存しない共通の制御インタフェースを提供し、制御の簡単化を実現することである。

【 0 0 1 0 】

【 課題を解決するための手段 】

本発明は、記憶装置の機能を制御するための共通インタフェースをコンピュータシステム内のバーチャライゼーションサーバやスイッチに設け、記憶装置の種類に依存しない機能制御方法を提供する。ホストコンピュータは、制御の対象となる記憶装置の種類にかかわらず、あらかじめ定義されたプロトコルで共通インタフェースに機能制御の指示を送る。共通インタフェースを提供する装置は、ホストコンピュータが認識する記憶領域と、記憶装置が提供する記憶領域の対応関係を管理する。また、共通インタフェースのプロトコルを各装置に固有のプロトコルに変換する手段を備え、各記憶装置に固有のインタフェースを介して機能制御を実現する手段をもつ。

【 0 0 1 1 】

共通インタフェースを提供する装置が機能制御の指示を受け取ると、指示の対象となる記憶領域と、それを提供している記憶装置を解釈する。例えば、指示が、ある仮想ボリュームに対するものである場合は、仮想ボリュームを構成するディスクボリュームが指示の対象となる。指示の対象となる記憶領域を提供する記憶装置の種類を識別し、その装置に固有の機能制御インタフェースを介して機能制御を指示する。各装置に固有のインタフェースはインタフェースサーバで隠蔽され、ホストコンピュータは制御対象の装置の種類を意識することなく、共通のインタフェースを利用して機能を制御することが可能となる。ホストコンピュータは、複数のインタフェースに対応し、それぞれを使い分ける必要がなくなるため、制御の簡単化が実現できる。

【 0 0 1 2 】

【 発明の実施の形態 】

第一の実施の形態

本実施形態は、ファイバチャネルで接続され、異なる機能制御インタフェースを持つ 2 つのディスクアレイ装置が提供する記憶領域をバーチャライゼーションサーバで仮想化するストレージバーチャライゼーション環境において、ディスクアレイ装置のミラー機能を用いて実現された仮想ボリュームのミラー機能を、バーチャライゼーションサーバが提供する共通インタフェースを利用して制御する例である。

【 0 0 1 3 】

( 1 ) システムの構成

図 1 に本実施形態の全体構成を示す。

【 0 0 1 4 】

10

20

30

40

50

2台のホストコンピュータ1101、1201および2台のディスクアレイ1401、1501があり、それぞれのファイバチャネルインタフェース1102、1202、1402、1502とファイバチャネルケーブル1314～1317によりバーチャライゼーションサーバ1301のファイバチャネルインタフェース1302、1303、1307、1308に接続されている。ホストコンピュータから記憶装置へのデータ入出力は、これらのファイバチャネルケーブル上の標準的なSCSIプロトコルで実現される。以下の説明では、ホストコンピュータやディスクアレイ装置がバーチャライゼーションサーバと直接接続されているが、本発明は通信媒体や接続形態に依存しないため、スイッチを経由した接続やファイバチャネル以外のSCSIやIPネットワークなど、他の通信媒体やプロトコルを用いた構成にも適用可能である。

10

#### 【0015】

ホストコンピュータ1101のメモリ1104内には、仮想ボリュームの機能制御を指示するための仮想ボリューム機能制御プログラム1106があり、CPU1103により実行される。仮想ボリューム機能制御プログラム1106は、ディスクアレイの機能制御指示を送出するホストコンピュータにあらかじめインストールされている。仮想ボリューム機能制御プログラム1106は、後述するように、制御する機能や制御対象となる仮想ボリュームなどの必要な情報をバーチャライゼーションサーバ1301が提供する共通インタフェース（ホストコンピュータから見て共通インタフェースとして供給される仮想ボリューム）に送り、機能制御を指示する。ホストコンピュータ1201も同様の構成を持つ。

20

#### 【0016】

ディスクアレイ1401では、コントローラ1404内の制御プログラム1407が全体の動作を統括する。制御プログラム1407は、ディスクボリューム1405および1406をホストコンピュータの記憶領域として提供する通常のディスクアレイ機能に加え、ミラー機能やスナップショット機能などのさまざまなインテリジェント機能を提供する。ディスクアレイ1501も1401と同様の構成を持ち、ミラー機能を含むインテリジェント機能を提供する。

#### 【0017】

バーチャライゼーションサーバ1301は、全体の動作をボリューム仮想化プログラム1309が制御し、ストレージバーチャライゼーション機能を提供する。記憶領域仮想化の実現方法の詳細は本発明の主眼ではないので、簡単に説明する。ディスクアレイが提供する複数のディスクボリュームから1つの仮想ボリュームを作成する場合、各ディスクボリュームの記憶領域を結合し、1つの仮想ボリュームに見せる。各ディスクボリュームのアドレス空間が仮想ボリュームのアドレス空間の一部にマッピングされ、1つの連続したアドレス空間を実現する。バーチャライゼーションサーバ1301内の制御プログラムが、仮想ボリュームとそれを構成する実際のディスクボリュームの対応関係、および仮想ボリュームのアドレス空間と実際のディスクボリュームのアドレス空間の対応関係を管理する。仮想ボリュームは、ホストコンピュータにはSCSIのLU(Logical Unit)、すなわち通常のディスクボリュームとして認識される。ホストコンピュータがReadまたはWriteコマンドを発行して仮想ボリュームにアクセスすると、バーチャライゼーションサーバ1301が宛て先のディスクボリュームおよびアドレスを解釈し、実際のディスクボリュームにコマンドを発行する。アクセス先が複数のディスクボリュームにまたがる領域である場合は、複数のコマンドを発行する。これにより、複数のディスクボリュームを結合した1つの仮想ボリュームが実現できる。

30

40

#### 【0018】

また、バーチャライゼーションサーバ1301は、ディスクアレイの機能を制御するための共通インタフェースを実現するために、仮想ボリューム管理表1313、ディスクボリューム管理表1312、インタフェース対応表1310、機能制御プログラムライブラリ1311、共通インタフェース管理表1314を持つ。これらの詳細については、後述する。

50

## 【0019】

## (2) ディスクアレイ装置の機能制御の説明

ミラー機能を例にディスクアレイの機能制御について説明する。ミラー機能は2つのディスクボリュームの一方を主ボリューム、他方を副ボリュームとして扱いデータの二重化を行う機能で、初期化、中断、再同期、解消、状態取得の5つの制御が可能である。

## 【0020】

例えば、ディスクアレイ1401のボリューム1405を主ボリューム、1406を副ボリュームとするペアとすると、まずミラーの初期化が指示されると、主ボリューム1405の内容を全て1406にコピーする。初期化終了後、主ボリュームの内容が更新されると、コントローラ1404は対応する副ボリューム1406の内容も自動的に更新する。ミラーの中断が指示されると、1405のデータを更新しても1406は自動更新されず、1406には中断指示時のデータが保存される。再同期を指示すると、2つのボリュームが同一の内容をもつように中断中に主副間で差異が生じた部分をコピーする。解消を指示すると、主副のペアが解消され、再び初期化を指示されない限りミラー機能が無効化される。状態取得を指示すると、コントローラ1404は、初期化処理中、中断中、エラー状態などの現在の状態を返す。状態は、制御指示と同様にあらかじめ定義されたデータ型で制御インタフェースを通して返される。

10

## 【0021】

ディスクアレイ1401では、LANインタフェース1403が各機能の制御インタフェースとして提供され、外部からあらかじめ決められたプロトコルで指示を受け、それに基づいてコントローラ1404が機能制御を行う。LANインタフェースのあらかじめ決められたポートに、例えば図5で示されるような制御ブロックを受け取ると、コントローラの制御プログラム1407が1バイト目5101に格納されているコマンドを解釈し、指示の操作内容を判別する。コマンドに続くパラメータフィールド5102～5104は、操作対象のディスクボリューム等、指示を実行するために必要なパラメータを記録するフィールドとして、あらかじめコマンド毎にデータの意味が定義されている。これにより、LANインタフェース1403に図5で示される制御ブロックを送ることで、外部からディスクアレイ1401の機能を制御することが可能となる。

20

## 【0022】

ディスクアレイ1501ではLANインタフェースがなく、機能制御の指示はファイバチャネルインタフェース1502を介して、制御対象のディスクボリュームへのSCSI Mode Selectコマンドとして送られる。状態取得は対象ボリュームへのMode Senseコマンドで指示され、状態はSenseデータとして返送される。

30

## 【0023】

## (3) 共通インタフェースの説明

バーチャライゼーションサーバが提供する共通インタフェースは、ディスクアレイ装置が提供する機能制御インタフェースと同様にさまざまな方法で実現可能であるが、本実施形態ではホストコンピュータからはファイバチャネルで接続されたディスクボリュームとして認識される仮想ボリュームとして実現する例を説明する。

## 【0024】

ホストコンピュータは、共通インタフェースを他の仮想ボリュームと同様に読み書き可能な記憶領域として認識し、ReadおよびWriteコマンドを発行することにより制御情報をやりとりする。ただし、実際の記憶領域となるディスクボリュームは割り当てられず、ボリューム仮想化プログラム1309によりアドレス空間のみ提供される。ホストコンピュータから図8のようなデータ構造をもつ制御ブロックを、共通インタフェースである仮想ボリュームの、例えば先頭512バイトに書くことで機能制御を指示する。図8のコマンド8101はミラー機能、スナップショット機能など、制御する機能を表し、サブコマンド8102は、例えばミラー機能の場合初期化、中断などの指示を表す。コマンドおよびサブコマンドは、各機能および指示に対してあらかじめ番号を定義しておくことにより、番号で指定する。また、仮想ボリューム識別子8103および8104は、機能を

40

50

適用する仮想ボリュームの識別子の指定に利用する。例えばミラーの初期化の場合、仮想ボリューム識別子 8 1 0 3 は主ボリューム、仮想ボリューム識別子 8 1 0 4 は副ボリュームを指定する。対象となる仮想ボリュームの指定方法は、識別子に限らず、仮想ボリュームを一意に識別できる情報であればよく、例えば、仮想ボリュームに割り当てられたバーチャライゼーションサーバのファイバチャネルインタフェースの WWN と LUN で指定する形態も可能である。その他のパラメータ 8 1 0 5 ~ 8 1 0 6 はそれ以外に必要な情報を指定するための領域として使用される。

#### 【 0 0 2 5 】

ボリューム仮想化プログラム 1 3 0 9 は、共通インタフェースの先頭 5 1 2 バイトへの Write コマンドを受け取ると、機能制御の指示とみなし、先頭 5 1 2 バイトを図 8 のデータ構造に従って解釈する。先頭 5 1 2 バイト以外への Write は、エラーとなる。また、バーチャライゼーションサーバからホストコンピュータへ仮想ボリュームの状態等の情報を受け渡す場合は、ホストコンピュータから共通インタフェースに Read コマンドを発行する。その場合、情報は Read コマンドに対するデータとして返され、特に図には示さないが、Write コマンドで書かれる制御ブロックと同様に、あらかじめ定義されたデータ構造が用いられる。

#### 【 0 0 2 6 】

管理者は、共通インタフェースに割り当てるファイバチャネルインタフェースの WWN および LUN を、ホストコンピュータの仮想ボリューム機能制御プログラムのパラメータとして設定する。あるいは、ホストコンピュータ上の仮想ボリューム機能制御プログラムがファイバチャネルインタフェースに接続された各 LU に Inquiry コマンドを発行し、あらかじめ定義された共通インタフェースに固有なベンダ ID やプロダクト ID を返す LU を探すことにより、共通インタフェースを自動で特定することもできる。

#### 【 0 0 2 7 】

共通インタフェースを実現するために、バーチャライゼーションサーバは仮想ボリューム管理表 1 3 1 3、ディスクボリューム管理表 1 3 1 2、インタフェース対応表 1 3 1 0、機能制御プログラムライブラリ 1 3 1 1、共通インタフェース管理表 1 3 1 4 を持つ。共通インタフェース管理表 1 3 1 4 は、図 1 2 に示す構成を持ち、共通インタフェースに割り当てられたファイバチャネルインタフェースの WWN と LUN を管理する。ここで記録された WWN (ファイバチャネルインタフェースに割り当てられたもの) と LUN の組み合わせは、他の仮想ボリュームでは使用できない。複数の共通インタフェースを定義することにより、例えば、ホストコンピュータ毎に異なる共通インタフェースを割り当てることもできる。

#### 【 0 0 2 8 】

仮想ボリューム管理表 1 3 1 3 は図 2 に示すような構成を持ち、各仮想ボリュームと、それを構成する実際のディスクボリュームの対応関係を管理する。各仮想ボリュームには、識別子として一意な通し番号が振られる (2 1 0 1)。また、ホストコンピュータがその仮想ボリュームにアクセスするために割り当てられるバーチャライゼーションサーバのファイバチャネルポートの WWN (World Wide Name) と (2 1 0 2)、仮想ボリュームの LUN (Logical Unit Number) が記録される (2 1 0 3)。さらに、表の 2 1 0 4 ~ 2 1 0 7 には仮想ボリュームを構成する実際のディスクボリュームの識別子 (通し番号) が記録される。図 2 の仮想ボリューム 1 の例では、仮想ボリューム 1 は WWN が wwn 3 のポートから lun 1 の LUN でアクセスされ、識別子が 1 および 3 の 2 つのディスクボリュームから構成される。Vol. 1 は一つ目のディスクボリュームを、Vol. 2 は二つ目のディスクボリュームを示す。制御処理の項目 (2 1 0 8) は、その仮想ボリュームがなんらかの機能制御を処理中であるかどうかのフラグである。処理中であれば「処理中」が記録される。

#### 【 0 0 2 9 】

ディスクボリュームは、仮想ボリュームとは別に図 3 のような構成を持つディスクボリューム管理表 1 3 1 2 で管理される。各ディスクボリュームは識別子として一意な通し番号

10

20

30

40

50



を持ち(3101)、そのディスクボリュームに割り当てられたディスクアレイ装置のファイバチャネルポートのWWN(3102)と、そのディスクボリュームのLUN(3103)および容量(3104)が記録される。また、そのディスクボリュームを提供するディスクアレイ装置の機種(3105)と装置識別子が記録される(3106)。機種情報は、例えば、Inquiryコマンドに対して返されるベンダIDやプロダクトIDのような、装置の種別に固有な情報である。一般に機種が変わるとプロトコルも変わる。装置識別子は、ディスクアレイ装置毎に割り当てられる一意な通し番号である。3107~3112には、各ボリュームを対象としてさまざまな機能を制御する場合に使用するAPIとディスクアレイ装置が提供する機能制御インタフェースが記録される。3108、3110、3112のインタフェースは、例えばディスクアレイのLANポートに制御パケットを送るインタフェースである場合はIPアドレスとポート番号を、制御対象のディスクボリュームにMode SelectやMode Senseで指示を送る場合はそのディスクボリュームに割り当てられているWWNやLUNを記録する。

10

#### 【0030】

各APIは、機能制御プログラムライブラリ1311で提供され、例えば、図9の構造体を引数とし、制御プログラム1309に動的にリンクされる関数として実現される。関数を機能制御ライブラリ1311から読み出す。その関数がAPIとして表される。各APIは、目的の機能を実現するために、対象となるディスクボリュームを提供しているディスクアレイ装置固有のプロトコルでディスクアレイに指示を送る。制御指示として送られるデータの構造は、装置毎に定義されているため、各APIは図9の共通インタフェースのデータ構造から図5のような装置固有のパラメータに変換する論理を含む。また、特に図では示さないが、API呼び出しからの戻り値が必要な場合も、呼び出し引数と同様に共通のデータ構造で値が返される。

20

#### 【0031】

図3のディスクボリューム管理表の各エントリは、バーチャライゼーションサーバが新しいディスクボリュームを認識する際にバーチャライゼーションサーバの管理者が作成する。ただし、項目のいくつかは、管理者の操作に頼らずボリューム仮想化プログラムがボリューム管理表へ自動的に登録することもできる。例えば、ファイバチャネルインタフェースに接続されたLUを走査し、そのWWN、LUNを検出したり、各ディスクボリュームにSCSI Read Capacityコマンドを発行することで容量を得たりすることができる。機種情報はSCSI Inquiryコマンドをディスクボリュームに発行し、ベンダIDやプロダクトIDを取得して記録することができる。また、各APIは、あらかじめ機種情報と使用するAPIの対応関係を図4のようなインタフェース対応表1310としてインストールしておくことにより、装置識別子に応じて必要なAPIをディスクボリューム管理表に記録することができる。

30

#### 【0032】

##### (4) 共通インタフェースを利用したミラー初期化

共通インタフェースを利用してディスクアレイ装置の機能を制御する例を、仮想ボリュームのミラー初期化を例に説明する。

#### 【0033】

共通インタフェースを用いた機能制御について説明する前に、その前段階として、本実施形態における仮想ボリュームの作成手順をボリューム仮想化プログラム1309の一部をなす図6のフローチャートを用いて説明する。バーチャライゼーションサーバに新しいディスクボリュームが接続されると、まずボリューム仮想化プログラムがそれをファイバチャネルインタフェースに接続されたデバイスとして認識する(ステップ6001)。次に、そのディスクボリュームのWWN、LUN、装置識別子、機能制御のAPIを、前述のように管理者もしくはボリューム仮想化プログラムがディスクボリューム管理表に登録する(ステップ6002)。登録したディスクボリュームから仮想ボリュームを作成すると、そのWWN、LUNおよび使用されるディスクボリュームを仮想ボリューム管理表に登録する(ステップ6003)。最後に、ホストコンピュータから仮想ボリュームを認識す

40

50

ることにより、仮想ボリュームがホストコンピュータの記憶領域として使用可能になる。

#### 【0034】

図1において、上記のような手順により、仮想ボリューム1がホストコンピュータ1101に提供され、仮想ボリューム2がホストコンピュータ1201に提供されている場合に、仮想ボリューム1および2をペアとするミラーを初期化する例を、同じくボリューム仮想化プログラムの一部である図7のフローチャートを用いて説明する。

#### 【0035】

まず、ステップ7001でホストコンピュータから共通インタフェースにWriteコマンドを発行し、先頭512バイトに図8の制御ブロックを書く。制御ブロックのコマンド8101にはミラー機能を指定し、サブコマンド8102には初期化を指定する。仮想ボリューム識別子1、2(8103、8104)には仮想ボリューム1および2の識別子を指定する。この制御ブロックを、図12の共通インタフェース管理表に記録されたファイバチャネルポートのLUに受信すると、パーチャライゼーションサーバのボリューム仮想化プログラムはミラー機能の初期化指示であることを判別し(ステップ7002)、ボリューム仮想化プログラム内のミラーの初期化用の処理ルーチンを実行する。定義されていないコマンドを含む制御ブロックのWriteはエラーとなる。ミラー初期化用の処理ルーチンは、制御ブロックの仮想ボリューム識別子8103および8104を参照して、制御対象となる仮想ボリュームを特定する(ステップ7003)。

10

#### 【0036】

ここで、仮想ボリューム管理表を参照し、対象の仮想ボリュームが他の機能制御を処理中であるか調べる(ステップ7012)。既に他の処理中であれば、ステップ7010に進む。処理中でなければ、仮想ボリューム管理表の制御処理の項目に「処理中」を記録する(ステップ7013)。さらに、仮想ボリューム管理表を参照し、それぞれの仮想ボリュームを構成するディスクボリュームを特定する(ステップ7004)。図2の仮想ボリューム管理表の例では、仮想ボリューム1はディスクボリューム1と3、仮想ボリューム2はディスクボリューム2と4から構成されていることがわかる。

20

#### 【0037】

次に、ステップ7005において、ディスクボリュームをミラーのペアとして組み合わせる。ディスクボリュームがミラーのペアとなるためには主ボリュームと副ボリュームが同じディスクアレイ装置内に存在し、さらにボリュームの容量が等しくなければならない。各ボリュームの容量および提供しているディスクアレイ装置は、ディスクボリューム管理表の容量および装置識別子から確認できる。主ボリュームとなるディスクボリューム群と、副ボリュームとなるディスクボリューム群の中から、それぞれ前記の条件を満たすものをミラーのペアとして組み合わせる。もし、容量が一致しないペアや同じ装置内に存在しないペアがあった場合は、ミラーの初期化失敗として終了する(ステップ7006)。条件を満たす組み合わせが複数ある場合、例えば、主ボリュームおよび副ボリュームとなる仮想ボリュームがそれぞれ、ディスクボリューム1と3および2と4で構成され、ディスクボリューム1と2および3と4でペアを組むことが可能で、一方、ディスクボリューム1と4および3と2でもペア形成が可能であるような場合も考えられる。そのような場合、ディスクボリュームの識別子が最も小さいもの同士を組み合わせる。前記の例の場合、主ボリュームとなるディスクボリュームは識別子の小さい順に1、3であり、副ボリュームについては同様に2、4となる。従って、識別子が小さい順に1と2、3と4がペアとなる。このようにすることで、別の機能指示の際にディスクボリューム同士のペア関係を知る必要がある場合にも、同じ仮想ボリュームペアについては必ず初期化処理と同じディスクボリュームのペアが得られる。

30

40

#### 【0038】

ミラーが組めることを確認したら、全ての組み合わせについて、ディスクアレイ装置にミラー初期化の指示を送る。各ペアについて、制御APIを呼び出す際に引数となる図9の構造を持つ制御データを作成する(ステップ7007)。ディスクボリューム管理表から、各ディスクボリュームに対する制御指示に使用されるAPIおよびインタフェースを調

50

べ、前述の制御データとインタフェースを引数としてAPIを呼び出す（ステップ7008）。例えば、ディスクボリューム1、2のペアについては、ディスクボリューム管理表を参照して、登録されたIPアドレスとポート番号を制御インタフェースとするfunc1のAPIにより初期化を指示する。これを全てのペアについて繰り返す（ステップ7009）。全てのペアに指示したら、仮想ボリューム管理表の「処理中」の記録を削除する（ステップ7014）。最後に、制御の指示が完了したことをホストコンピュータに伝える。これは、共通インタフェースへのWriteコマンドに対するステータス返送として実現する（ステップ7010）。

#### 【0039】

このように、ステップ7002においてホストコンピュータから共通インタフェースで受信した制御ブロックを、ステップ7007とステップ7008においてディスクボリューム固有のプロトコルに変換してミラー初期化指示を出している。上記の手順により、複数のディスクボリュームから構成されている仮想ボリュームのミラー機能を、ディスクアレイ装置の種類にかかわらず共通のインタフェースから制御することが可能となる。

#### 【0040】

（5） 共通インタフェースを利用したミラーの状態取得  
ある仮想ボリュームのミラーの状態を取得する場合も、前述のミラーの初期化と同様の手順で、ホストコンピュータから共通インタフェースに指示を送り、バーチャライゼーションサーバがディスクアレイ装置からミラーの状態を取得するためのAPIを呼び出す。ただし、状態取得の場合は、バーチャライゼーションサーバが各ディスクアレイ装置から取得したディスクボリュームのミラーの状態を仮想ボリュームのミラーの状態に変換し、ホストコンピュータに伝える必要がある点異なる。

#### 【0041】

ミラーの状態取得の手順を、図10のフローチャートを用いて説明する。図10もボリューム仮想化プログラム1309の一部である。ステップ10001～10006までは図7のステップ7001～7005と同様である。ステップ10008～10010で、各ディスクボリュームのミラーペアの状態を取得する。次に、これらの状態から仮想ボリュームのミラーペアの状態を作成する（ステップ10011）。例えば、ディスクボリュームのミラーペアの状態が初期化前の「無効」、初期化処理中である「初期化」、初期化が完了し主副が同じデータを持つ「同期」、ミラーを中断している「中断」、中断から再び同期状態に戻す処理中である「再同期」、障害が起きている「エラー」の6つの状態であるとする、仮想ボリュームのミラーペアの状態は図11のように作成することができる。全てのディスクボリュームのミラーペアが一致した場合は、仮想ボリュームのミラーペアの状態もその状態になる。もし一致しない状態があれば、ディスクボリュームのミラーペアの状態の整合性が取れていないため、仮想ボリュームのミラーペアの状態は「エラー」となる。ただし、初期化処理および再同期処理では、全てのディスクボリュームのミラーペアが同時に処理を終えるとは限らないため、早く処理が終わったミラーペアの状態は「同期」となる可能性がある。その場合は不一致があっても仮想ボリュームのミラーペアの状態をエラーではなく、それぞれ「初期化」および「再同期」とする。

#### 【0042】

仮想ボリュームのミラーペアの状態を作成したら、仮想ボリューム管理表の「処理中」の記録を削除し（ステップ10012）、ホストコンピュータにステータスを返す（ステップ10013）。このステータスとはSCSIプロトコルの一部として規格で定義されたもので、通常エラーがなければGood（バイトコード＝00000）が返される。これにより、ホストコンピュータの仮想ボリューム機能制御プログラムは状態取得処理の完了を認識し、状態を読み出すために共通インタフェースにReadコマンドを送る。バーチャライゼーションサーバのボリューム仮想化プログラムは、ステップ10011で作成した仮想ボリュームのミラーペアの状態をReadコマンドに対するデータとして返送し（ステップ10015）、最後にステータスを返す（ステップ10016）。

#### 【0043】

10

20

30

40

50

上述のように、図 11 は仮想ボリュームのミラー状態と、仮想ボリュームを構成する実際のディスクボリューム群のミラー状態の対応表である。図 10 のステップ 10008 ~ 10010 で取得した実際のディスクボリューム群のミラー状態を図 11 に照らし合わせて仮想ボリュームのミラー状態を作成する（ステップ 10011）。図 11 の対応表による変換がプロトコルの変換にあたる。

#### 【0044】

上記の手順により、バーチャライゼーションサーバからホストコンピュータへ情報を渡す必要がある機能制御も実現することができる。

#### 【0045】

##### （6） 効果

本実施形態により、異なる機能制御インタフェースを持つ複数のディスクアレイ装置が存在するストレージバーチャライゼーション環境において、仮想ボリューム単位でミラー機能などのストレージ機能を適用することができる。その際、構成する記憶領域が、どの装置に存在するかを意識することなく、共通の方法で機能制御を指示することができる。

#### 【0046】

なお、本実施形態ではストレージバーチャライゼーションの基本機能と、機能制御の共通インタフェースが同じバーチャライゼーションサーバで提供されているが、それぞれの機能を個別の装置内で実現することも可能である。また、サーバに限らず、同様のストレージバーチャライゼーション機能を持つスイッチ装置等に適用することもできる。ここでは、これらを記憶装置の接続装置と呼ぶ。さらに、ディスクボリューム単位の制御に限らず、例えば、ディスクアレイをファイルサーバ、ファイバチャネルと SCSI プロトコルを LAN と NFS プロトコル、バーチャライゼーションサーバ内の管理単位をファイルシステム、共通インタフェースを LAN インタフェースとすることにより、ファイルシステム単位での機能制御にも適用することができる。

#### 【0047】

##### 第二の実施形態

本実施形態の構成は、図 1 の第一の実施形態とほぼ同様であるが、バーチャライゼーションサーバで複数の共通インタフェースを提供する。また、バーチャライゼーションサーバがディスクアレイ装置に制御を指示するときに、一部のディスクボリュームに対する指示が失敗した場合に、エラー処理を実行する手段を備える。以下、第一の実施形態と異なる点のみを説明する。

#### 【0048】

##### （1） 複数の機能制御インタフェース

本実施形態では、バーチャライゼーションサーバが 2 つの共通インタフェースをホストコンピュータに提供する。第一の実施形態で説明した仮想ボリュームによるインタフェースに加え、図 1 の LAN インタフェース 1306 を利用した LAN 経由のインタフェースを提供する。共通インタフェース管理表 1314 は、図 12 の構成の表に加えて、図 13 の構成の表も持つ。図 13 の表に記録された LAN インタフェース（IP アドレス）とポート番号の組み合わせをあて先とする入力データは、機能制御ブロックとして解釈される。LAN インタフェースで共通インタフェースを実現する場合、第一の実施形態でファイバチャネル上の SCSI コマンドおよびデータの送受信で実現したように、イーサネット（ゼロックス社の登録商標）上のデータ送受信で実現する。SCSI のステータスの代わりに、処理の終了を示すために確認のパケット（acknowledge）を返す。この第二の共通インタフェースを利用することにより、例えばファイルシステムを介さないところのファイバチャネルインタフェースへの入出力が不可能なホストコンピュータでは、LAN インタフェースを利用した機能制御指示が可能となる。即ち、ファイバチャネルインタフェースの入出力がファイルシステム経由でのみ可能なホストコンピュータでは全ての入出力はファイル操作となるため、ユーザが入出力のアドレスやパケットのデータ構造を指定することが出来ず、共通インタフェースに制御パケットを送ることが出来ない。そのようなホストコンピュータでは、LAN インタフェースを利用することで共通インタフェ

10

20

30

40

50

ースに制御パケットを送ることが出来る。

#### 【0049】

図1のホストコンピュータ1201上の仮想ボリューム機能制御プログラム1206は、LAN通信の機能を持ち、仮想ボリュームとして提供される第一の実施形態の共通インタフェースではなく、LANインタフェースを利用して実現された共通インタフェースにアクセスする機能を持つ。共通インタフェースのIPアドレスおよびポート番号は、第一の実施形態における共通インタフェースのWWNとLUNと同様に、管理者が仮想ボリューム機能制御プログラムに設定する。

#### 【0050】

ディスクアレイ1401は、第一の実施形態で説明したLANインタフェースによる機能制御に加え、ディスクアレイ1501と同様にファイバチャネルインタフェース経由での Mode Select / Mode Sense コマンドによる機能制御方法を提供する。すなわち、バーチャライゼーションサーバ上のボリューム仮想化プログラムは、LAN 経由もしくはファイバチャネル経由のどちらでもディスクアレイ1401に機能制御の指示を送ることが可能である。

#### 【0051】

バーチャライゼーションサーバ内のボリューム管理表は、ディスクアレイ1401の複数の制御インタフェースに対応するために変更される。また、機能制御を指示したときにエラーが発生した場合の回復処理を定義するための項目が追加される。変更したディスクボリューム管理表を図14に示す。図14で14101から14106までの項目は、図3のボリューム識別子3101から装置識別子3106までの項目に対応し、変更はないので説明を省略する。14107は機能制御を指示する際に呼び出すAPIであるが、ディスクアレイ1401のように複数の制御インタフェースを持つ装置のディスクボリュームについては、それぞれのインタフェースに対応する複数のAPIを登録する。図14の例では、ディスクボリューム1を提供し、装置識別子Aで表されるディスクアレイは2つの制御インタフェースを持ち、ミラー初期化のAPIとしてそれぞれfunc1とfunc5を登録する。一方、装置識別子Bで表されるディスクアレイは制御インタフェースを1つしか持たないため、APIも1つ(func5のみ)を登録する。また、それぞれのAPIで利用する制御インタフェースを14108に記録する。

#### 【0052】

14109のRecovery APIは、エラー処理を行う際に呼び出すAPIである。例えば、2つのディスクボリュームからなる仮想ボリュームを対象とする機能制御を指示し、一方のディスクボリュームに対する制御が成功し他方の制御が失敗した場合に、仮想ボリュームの状態を指示前の状態に戻すために、成功した方の指示を取り消すときに使用する。Recovery APIはAPIと同じ引数で呼び出され、APIと同様の方法で登録する。

#### (2) エラー処理

本実施形態における機能制御の指示の手順は、第一の実施形態とほぼ同様であるが、ディスクアレイが提供する複数の制御インタフェースと、あらかじめボリューム管理表に登録したRecovery APIを利用したエラー処理を行う点が異なる。以下、ミラー初期化指示の手順について第一の実施形態からの変更部分のみ説明する。ボリューム管理表のミラー初期化のRecovery APIには、ミラーの解消を指示するAPIがあらかじめ登録されているとする。

#### 【0053】

図15は、図7のミラーの初期化処理の、ステップ7006とステップ7014の間の変更部分を示したものである。ステップ15001で制御指示を送る、ディスクボリュームのミラーペアを1つ選び、ボリューム管理表からそのディスクボリュームのミラー初期化指示に使用するAPIを1つ選ぶ(ステップ15002)。そのAPIを呼び出し(ステップ15002)、呼び出しが成功であれば全てのディスクボリュームのミラーペアについて同様の処理を繰り返す(ステップ15003、15004)。もしLANやファイバ

チャンネルケーブルの切断などの理由でディスクアレイへの制御が指示できず、APIの呼び出しが失敗したら、別のAPIをボリューム管理表から選択し、リトライする（ステップ15005）。これはAPIが1つのディスクボリュームに対していくつかあるので、試していないAPIがあるかを判定するものである。失敗は、タイムアウトやディスクアレイからのエラーステータスの受信で認識できる。ボリューム管理表に登録されている全てのAPIで呼び出しが失敗したら、これまでにAPI呼び出しが成功したディスクボリュームのミラーペアについてRecovery APIを呼び出し、ミラーの解消を指示する（ステップ15006、15007）。このRecovery APIの呼び出しを、API呼び出しが成功した全てのディスクボリュームのミラーペアについて繰り返す（ステップ15008）。全てのAPI呼び出しが成功、もしくはエラー処理がすべて完了したら、ステップ7014以降の終了処理に進む。 10

【0054】

### （3） 効果

本実施形態では、バーチャライゼーションサーバが複数の共通インタフェースを提供することにより、ホストコンピュータが使用する通信手段を選択することができる。そのため、各ホストコンピュータが制御指示を送るために利用できる通信手段が異なる場合にも、第一の実施形態と同様の効果が得られる。また、複数のディスクボリュームから構成される仮想ボリュームを対象とする機能を制御する場合、ディスクアレイに複数の制御指示を送り、その一部が失敗したときに、既に成功した指示を取り消し、仮想ボリュームを一貫性の取れた状態に戻すことができる。なお、本実施形態では各APIに対して単一のRecovery APIを定義したがる、エラー状態毎に使用するRecovery APIを定義することもできる。言い換えれば、第三の実施形態では図14に示すように、ある1つのAPIによる処理を取り消すAPIとして1つのRecovery APIを定義している。しかし、1つのAPIに対して複数のRecovery APIを定義するような実装も考えられる。例えば、通信路障害によるエラー状態と、ストレージ装置のコントローラ障害によるエラー状態では別のRecovery APIを用いることが考えられる。このように複数のRecovery APIをエラーの種類によって使い分けることも出来る。また、本実施形態では、APIの呼び出しが成功した制御対象にはRecovery APIを使用しないが、必要に応じて、API呼び出しが成功した場合に使用するRecovery APIを定義することもできる。 20 30

【0055】

### 第三の実施形態

本実施形態は、本発明をファイバチャネルネットワークによるSAN環境に適用し、ディスクボリュームの機能制御に共通インタフェースを利用する例である。バーチャライゼーションサーバではなく、ファイバチャネルスイッチで共通インタフェースを提供する。さらに、機能制御のログを管理コンソールで記録し、ホストコンピュータから指示できる制御を制限するセキュリティ機能を提供する。本実施形態では、共通インタフェースとして、制御の対象となるボリュームへのMode SelectおよびMode Senseコマンドの送信により機能制御を指示する。

【0056】

### （1） システムの構成

図16に本実施形態の全体構成を示す。

【0057】

ホストコンピュータ16101、16201およびディスクアレイ16401、16501の構成、動作は、第一の実施形態と同様であり、ファイバチャネルスイッチ16301に接続されている。ただし、ホストコンピュータ上の仮想ボリューム機能制御プログラムは、第一実施形態とは異なり、Read、Writeコマンドではなく、Mode Select、Mode Senseコマンドを送信する。それぞれのページコードには、あらかじめ機能制御用に定義したペンダユニークな番号を使用する。パラメータリストには、図17のような構造のデータを用いる。Mode SelectやMode Sense 40 50

e コマンドは、制御対象となるボリュームに送るため、制御対象をパラメータリスト内で指定する必要はない。ミラーペアの初期化のように、2つのボリュームを指定する必要がある制御では、パラメータリスト内で2つ目のボリュームを指定する。そのようなボリュームは、例えば、制御対象ボリュームに割り当てられているディスクアレイのポートのWWNとLUNをパラメータリストの一部として指定する。

#### 【0058】

ファイバチャネルスイッチ16301について、通常のファイバチャネルスイッチと異なる点を説明する。ホストコンピュータやディスクアレイと接続するためのファイバチャネルポートを提供するポートモジュール16302は、ファイバチャネルインタフェース16310とインタフェース制御プロセッサ16311から構成される。インタフェース制御プロセッサ16311は、共通インタフェースへの制御指示を受け取ると、CPU16307に通知する。具体的には、ファイバチャネルインタフェースで受信されたSCSIコマンドのCDB(Command Descriptor Block)を解釈し、受信したコマンドが機能制御用として定義されたページコードを持つMode Select、Mode Senseコマンドであれば、共通インタフェースへのアクセスであると判断して、送信元ホストコンピュータのWWNと、送信先ボリュームのWWNおよびLUN、コマンドの内容をCPU16307に通知する。

10

#### 【0059】

CPU16307はメモリ16308内のスイッチ制御プログラム16312を実行し、スイッチの動作を制御する。インタフェース対応表16313、ボリューム制御プログラムライブラリ16314、ボリューム管理表16315は第一の実施形態と同様の構成である。これらを用いてスイッチ制御プログラム16312がプロトコル変換を行う。詳細は第一、第二の実施形態と同じであるため省略する。本実施形態では、ストレージパーティライゼーション機能はないため、仮想ボリューム管理表は持たない。また、制御対象となるディスクボリューム自体がホストコンピュータからの制御コマンドのあて先、すなわち共通インタフェースとなるため、共通インタフェース管理表も持たない。

20

#### 【0060】

制御権限管理表16316は、図18のように構成され、各ディスクボリュームを対象とする機能制御について、制御を指示することのできるホストコンピュータをファイバチャネルインタフェースのWWNで指定する。図18の例では、ディスクボリューム1に対するミラー初期化および中断の制御は、wwn1およびwwn2のWWNを持つホストコンピュータが指示することができる。また、スナップショットの取得については、wwn2のホストコンピュータのみ指示できることを表している。これらの情報は、管理者が登録・変更する。

30

#### 【0061】

スイッチ制御プログラム16312は、共通インタフェースを介した制御指示を受け取ると、LANインタフェース16309から、あらかじめ管理者により設定されたIPアドレスとポート番号で指定される管理コンソール16601に制御記録を通知する。通知内容には、時刻、制御の指示元、制御の許可および不許可、制御対象、制御内容、ディスクアレイへの指示に用いられたインタフェース、および制御の成功/失敗を識別する情報が含まれる。

40

#### 【0062】

管理コンソール16601では、制御記録を蓄積するためのログ記録プログラム16605がCPU16602により実行されている。ログ記録プログラムは、ファイバチャネルスイッチからの制御記録がLANインタフェース16603に到着するのを待ち、それをログファイル16606に記録する。また、ログ記録プログラムは、管理者が記録された情報を閲覧するために、ログを表示する機能を持つ。

#### 【0063】

### (2) 共通インタフェースへの制御指示の制限とログの記録

本実施形態において、Mode Selectコマンドによりディスクボリュームの機能

50

制御を指示するスイッチ制御プログラム 16312 の処理の流れを図 19 のフローチャートを用いて説明する。

【0064】

まず、ホストコンピュータ上の仮想ボリューム制御プログラムが、制御対象となるディスクボリュームに Mode Select コマンドを送る（ステップ 19001）。Mode Select のページコードには、機能制御用としてあらかじめ割り当てたコードを用い、図 17 に示す制御ブロックをパラメタリストとして送る。ファイバチャネルスイッチのポートモジュールは、通常はスイッチモジュールを介して他のポートモジュールにコマンドをフォワードするが、コマンドが機能制御用ページコードを持つ Mode Select であることを検出すると、コマンドの内容とその送信元の WWN および送信先の WWN と LUN を CPU に通知する（ステップ 19002）。スイッチの CPU は、ステップ 19003 で、機能制御が、指示の送信元に対して、許可されていることを確認する。まずディスクボリューム管理表を参照し、送信先ボリュームの WWN および LUN から制御対象のボリュームの識別子を求める。次に、制御権限管理表を参照し、そのボリュームを対象とする、制御ブロックで指定された制御機能が送信元の WWN に対して許可されていることを確認する。

10

【0065】

もし、制御を許可する WWN の一覧に、指示の送信元 WWN が記録されていなければ制御が許可されていないので、LAN インタフェースから管理コンソールのログ記録プログラムに指示を拒否したという記録を送信し（ステップ 19004）、ステップ 19010 に進む。一方、制御が許可されていれば、第一の実施形態と同様に、ディスクボリューム管理表に登録されている API および制御インタフェースを用いて、ディスクアレイに対して機能制御を指示する（ステップ 19006）。API の呼び出し、すなわちディスクアレイの機能制御の成否を確認し（ステップ 19007）、API 呼び出しの成功あるいは失敗をそれぞれステップ 19008、19009 でログに記録する。最後に、指示の送信元ホストコンピュータに SCSI ステータスを返して（ステップ 19010）、処理を終了する。

20

【0066】

（3） 効果

本実施形態により、ストレージバーチャライゼーションを用いない通常の SAN 環境において、共通インタフェースを用いたディスクボリュームの機能制御が可能となる。また、各ディスクボリュームの機能毎に制御を許可するホストコンピュータを指定し、制御権限を管理することができる。さらに、制御の履歴をログに記録することができる。なお、本実施形態では、制御権限をホストコンピュータ単位で管理したが、他の管理単位にも適用可能である。例えば、制御権限管理表の制御許可対象をユーザ ID で管理し、ホストコンピュータからの制御指示にユーザ ID を含めることにより、ユーザ単位での権限管理ができる。また、本実施形態では、ログ情報を直ちに外部の管理コンソールに送信するが、スイッチ内部の記憶領域に蓄積し、定期的に管理コンソールに送信、あるいは管理コンソールから取得することも可能である。

30

【0067】

以上述べたように、本発明の実施例によれば、各装置に固有のインタフェースは隠蔽され、ホストコンピュータは制御対象の装置の種類を意識することなく、共通のインタフェースを利用して機能を制御することが可能となる。これにより、制御指示を発行するプログラムの構成が簡便化される。

40

【0068】

複数の共通インタフェースを提供することにより、ホストコンピュータが使用する通信手段を選択することができる。そのため、制御指示を送るために利用できる通信手段が異なる装置が混在する環境にも対応することができる。また、複数のディスクボリュームを対象とする一連の機能制御を指示する場合、その一部が失敗したときのために、エラー処理の機能制御を定義できる。

50



## 【 0 0 6 9 】

さらに、機能制御の種別および制御の対象毎に、制御を許可するコンピュータを管理することができる。また、制御の履歴をログに記録することができる。

## 【 0 0 7 0 】

## 【 発明の効果 】

以上述べたように、本発明によれば、各装置に固有のインタフェースは隠蔽され、ホストコンピュータは制御対象の装置の種類を意識することなく、共通のインタフェースを利用して機能を制御することが可能となる。これにより、制御指示を発行するプログラムの構成が簡単化される。

## 【 図面の簡単な説明 】

10

【 図 1 】 本発明の第一実施形態の全体構成図。

【 図 2 】 本発明の第一実施形態における仮想ボリューム管理表の構成を示す図。

【 図 3 】 本発明の第一実施形態におけるディスクボリューム管理表の構成を示す図。

【 図 4 】 本発明の第一実施形態におけるインタフェース対応表の構成を示す図。

【 図 5 】 本発明の第一実施形態におけるディスクアレイの制御ブロックの構成を示す図。

【 図 6 】 本発明の第一実施形態における仮想ボリューム作成のフローチャート。

【 図 7 】 本発明の第一実施形態におけるミラーペア初期化のフローチャート。

【 図 8 】 本発明の第一実施形態における共通インタフェースの制御ブロックの構成を示す図。

【 図 9 】 本発明の第一実施形態における制御 A P I の引数データの構成を示す図。

20

【 図 1 0 】 本発明の第一実施形態における仮想ボリュームの情報取得のフローチャート。

【 図 1 1 】 本発明の第一実施形態におけるディスクボリュームと仮想ボリュームの状態の対応表。

【 図 1 2 】 本発明の第一実施形態における共通インタフェース管理表の構成を示す図。

【 図 1 3 】 本発明の第二実施形態における共通インタフェース管理表の追加部分の構成を示す図。

【 図 1 4 】 本発明の第二実施形態におけるボリューム管理表の構成を示す図。

【 図 1 5 】 本発明の第二実施形態におけるミラーペア初期化のフローチャート。

【 図 1 6 】 本発明の第三実施形態の全体構成図。

【 図 1 7 】 本発明の第三実施形態における共通インタフェースの制御ブロックの構成を示す図。

30

【 図 1 8 】 本発明の第三実施形態における制御権限管理表の構成を示す図。

【 図 1 9 】 本発明の第三実施形態におけるディスクボリュームの機能制御指示のフローチャート。

## 【 符号の説明 】

1 1 0 1 , 1 2 0 1    ホストコンピュータ

1 3 0 1    パーチャライゼーションサーバ

1 4 0 1 , 1 5 0 1    ディスクアレイ

1 1 0 6 , 1 2 0 6    仮想ボリューム機能制御プログラム

1 3 0 9    ボリューム仮想化プログラム

40

1 3 1 0    インタフェース対応表

1 3 1 1    機能制御プログラムライブラリ

1 3 1 2    ディスクボリューム管理表

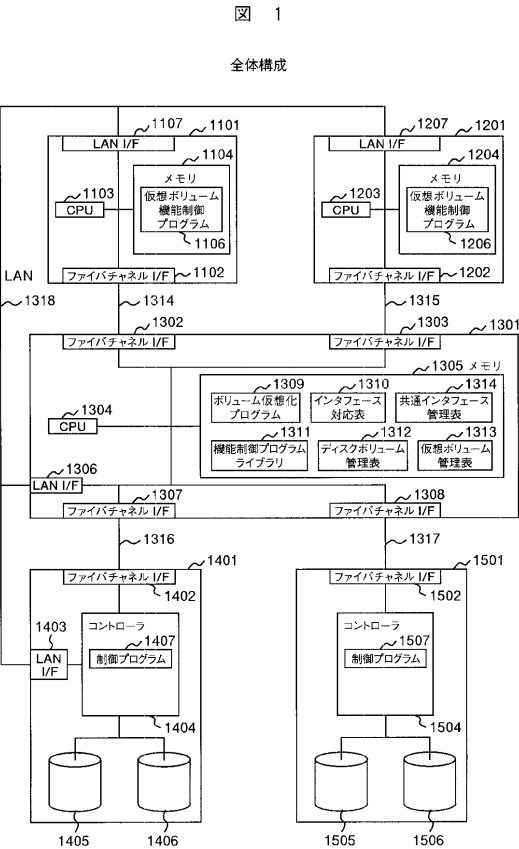
1 3 1 3    仮想ボリューム管理表

1 3 1 4    共通インタフェース管理表

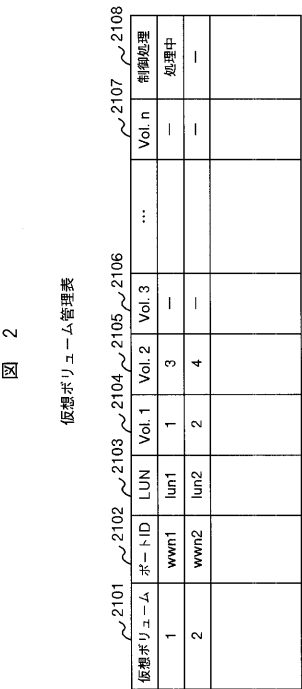
1 4 0 4 , 1 5 0 4    コントローラ

1 4 0 5 , 1 4 0 6 , 1 5 0 5 , 1 5 0 6    ディスクボリューム。

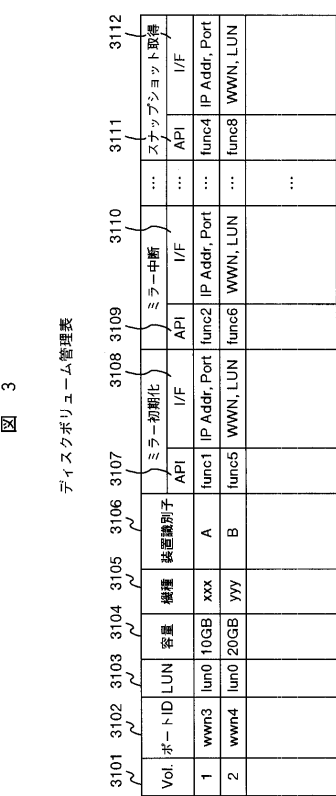
【図 1】



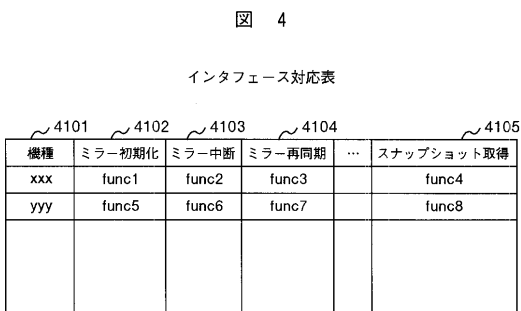
【図 2】



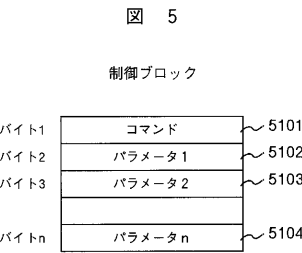
【図 3】



【図 4】



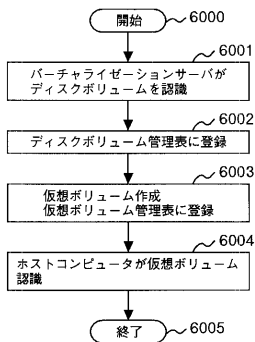
【図 5】



【図 6】

図 6

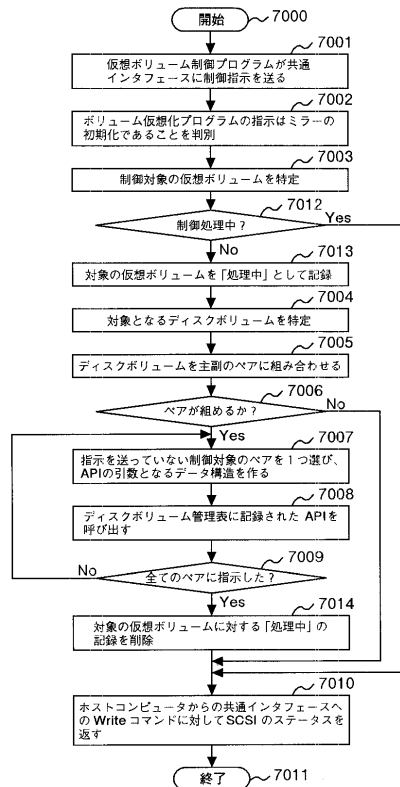
## 仮想ボリュームの作成手順



【図 7】

図 7

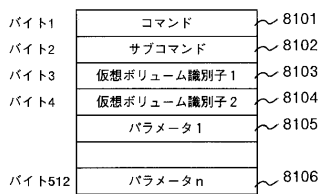
## ミラーペアの初期化フロー



【図 8】

図 8

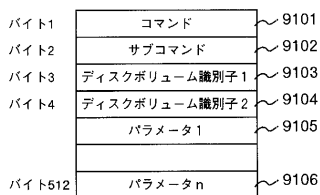
## 制御ブロック



【図 9】

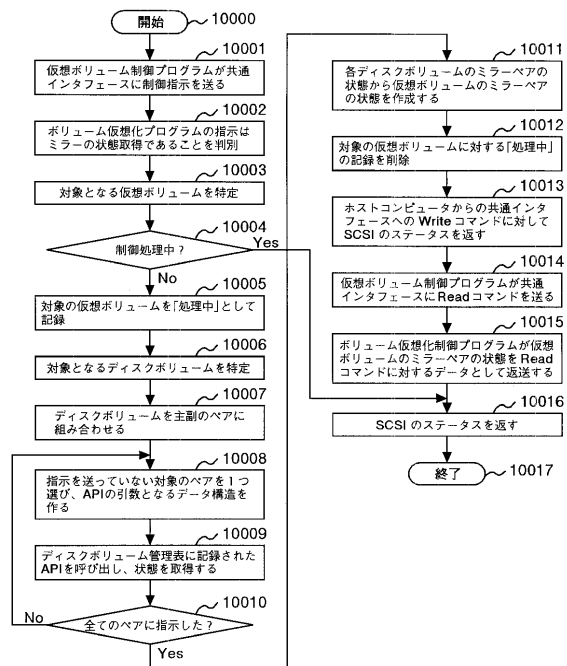
図 9

## 制御 API の引数データ構成



【図 10】

図 10



【図 1 1】

ディスクボリュームと仮想ボリュームの状態の対応表

ディスクボリュームのミラーペアの状態		仮想ボリュームのミラーペアの状態	
全てのミラーペアの状態が一致		ディスクボリュームのミラーペアの状態と同じ	
全てのミラーペアの状態が初期化または同期のいずれか		初期化	
全てのミラーペアの状態が再同期または同期のいずれか		再同期	
上記以外		エラー	

図 1 1

【図 1 2】

図 1 2

共通インタフェース管理表

ポートID	LUN
wwn5	lun0
wwn6	lun7

【図 1 3】

図 1 3

共通インタフェース管理表（追加）

ポートID	ポート番号
IP Addr.1	Port num.1
IP Addr.2	Port num.2

【図 1 4】

ディスクボリューム管理表

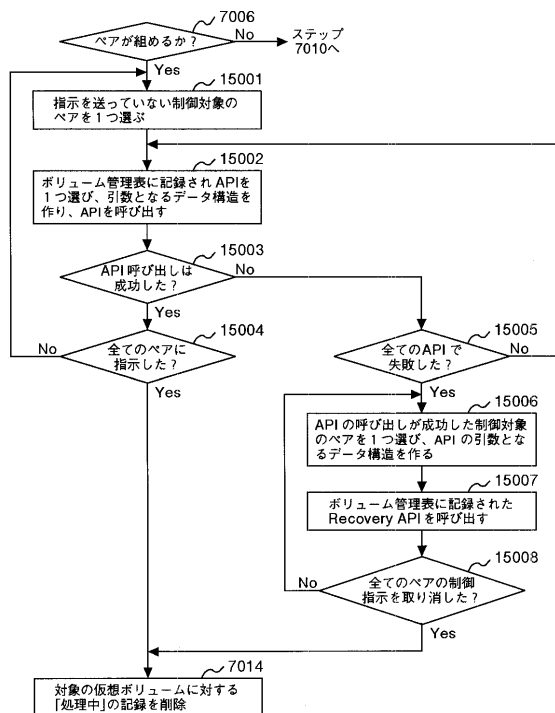
Vol.	...	14101	14106	14107	14108	14109	14110	14111	14112
		ミラー初期化		スナップショット取得		Recovery API		Recovery API	
		API		API		API		API	
		I/F		I/F		I/F		I/F	
		func1		func10		func4		func11	
		A		WWN2, LUN0		WWN2, LUN0		WWN2, LUN0	
		func5		func12		func8		func13	
		B		func14		WWN1, LUN0		func15	
		...		...		...		...	
		...		...		...		...	

図 1 4

【図 1 5】

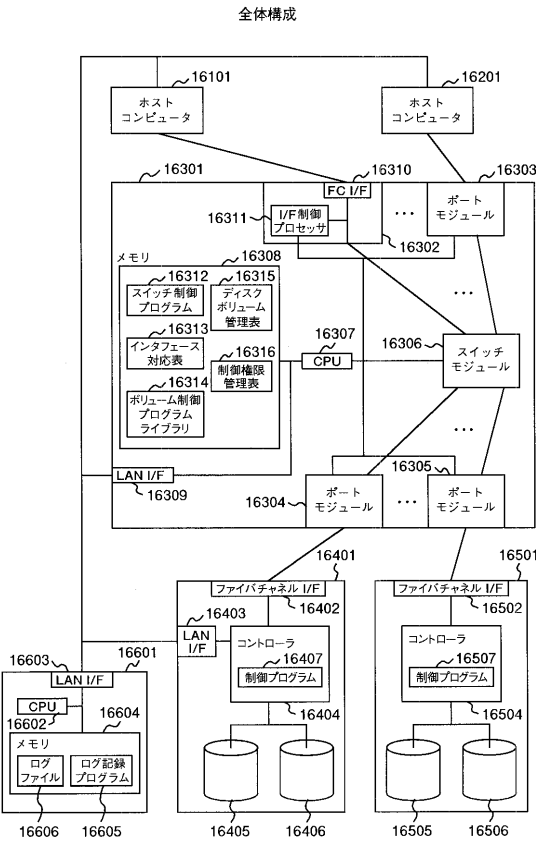
図 1 5

ミラーペアの初期化フロー



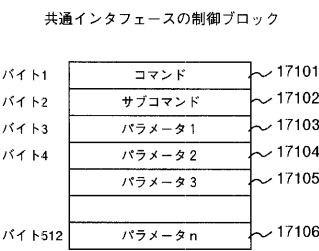
【 図 1 6 】

図 1 6



【 図 1 7 】

図 1 7



【 図 1 8 】

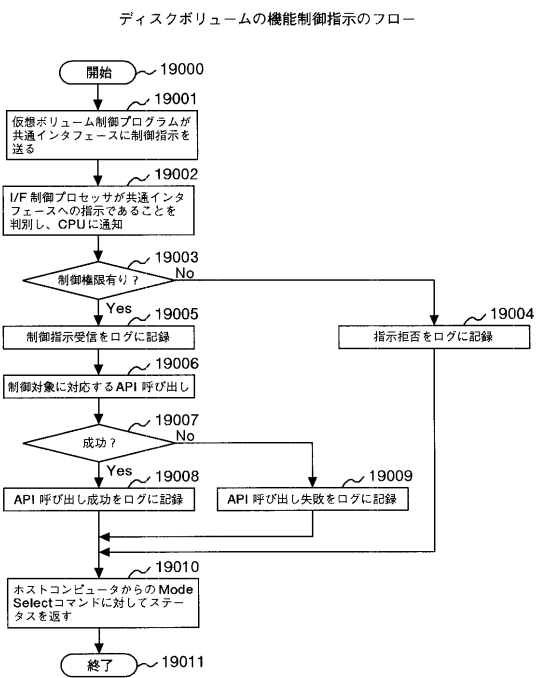
図 1 8

制御権限管理表

Vol.	機能制御	制御許可
1	ミラー初期化	wwn1, wwn2
	ミラー中断	wwn1, wwn2
	:	:
	スナップショット取得	wwn2
2	ミラー初期化	wwn3, wwn4
	ミラー中断	wwn3, wwn4
	:	:
	スナップショット取得	—
:	:	:

【 図 1 9 】

図 1 9



---

フロントページの続き

(72)発明者 山本 康友

神奈川県川崎市麻生区王禅寺 1 0 9 9 番地 株式会社日立製作所システム開発研究所内

(72)発明者 村岡 健司

神奈川県小田原市中里 3 2 2 番地 2 号 株式会社日立製作所 R A I D システム事業部内

F ターム(参考) 5B065 BA01 CA11 CA15 CA18 CA30 CC02 ZA01