(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4119971号
(P4119971)
(45)発行日　平成20年7月16日(2008.7.16)　(24)登録日　平成20年5月9日(2008.5.9)

(51)Int.Cl.　F I
G06F 13/14 (2006.01)　G06F 13/14 320A
G06F 13/14 320H

請求項の数 19　(全 27 頁)

(21)出願番号　特願平10-185331
(22)出願日　平成10年6月30日(1998.6.30)
(65)公開番号　特開平11-120117
(43)公開日　平成11年4月30日(1999.4.30)
審査請求日　平成17年6月24日(2005.6.24)
(31)優先権主張番号　08/885149
(32)優先日　平成9年6月30日(1997.6.30)
(33)優先権主張国　米国(US)

(73)特許権者　597004720
サン・マイクロシステムズ・インコーポレーテッド
Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃ．
アメリカ合衆国　カリフォルニア州　９５０５４　サンタ　クララ　ネットワーク　サークル　４１５０
(74)代理人　100064621
弁理士　山川　政樹
(72)発明者　スリニバサン・ヴィスワナサン
アメリカ合衆国・９４５３６・カリフォルニア州・フレモント・レキシントン　ストリート・３８７２５・アパートメント　３３９

最終頁に続く

(54)【発明の名称】グローバル・ファイル・システムを基礎とした、クラスタ上の装置を可視的にするシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
物理装置が接続されている複数のノードを含むクラスタ上の物理装置をグローバルに可視的にするシステムであって、
クラスタ上で実行するグローバル・ファイル・システムと、
それぞれが物理装置の対応するサブセットを管理するものであって、それぞれがグローバルに一意のメジャー番号に関連づけられている複数の装置ドライバと、
各装置に対して、その装置がどのノードからもアクセスされるように、少なくとも１つのグローバルに一意の識別子を確立するよう構成された装置レジストラと
を備え、
そのグローバルに一意の識別子が、
クラスタのユーザが装置を識別するグローバルに一意の論理名と、そして
グローバル・ファイル・システムが装置を識別するグローバルに一意の物理名と
を含み、
グローバルに一意の物理名が、
装置を管理するために使用される装置ドライバに対応する第１の部分と、そして
装置を管理するために使用される装置ドライバに関してグローバルに一意であるマイナー名に対応する第２の部分と
を含み、
それぞれの装置の論理名と物理名との間の１対１のマッピングが装置レジストラによっ

て確立される、
システム。
【請求項２】
さらに、
前記装置レジストラによって維持され、クラスタ内の装置の物理的関連を表す装置情報（ｄｅｖ＿ｉｎｆｏ）データ構造であって、その各物理的関連は、前記グローバル・ファイル・システムによって維持されている装置ファイルの物理名に対応している、装置情報（ｄｅｖ＿ｉｎｆｏ）データ構造を備え、
前記装置レジストラが、
接続された装置に対してグローバルに一意の装置タイプ（ｄｅｖ＿ｔ）値を決定し、ｄｅｖ＿ｉｎｆｏデータ構造内にエントリを作成するとともに、接続された装置に対する対応する物理名を作成し、
接続した装置に対して、ｄｅｖ＿ｔ値に基づく論理名と対応する物理名を作成し、接続された装置のｄｅｖ＿ｔ値を、接続された装置を表す装置ファイルに関連付ける、
ように構成されている請求項１に記載のシステム。
【請求項３】
前記ｄｅｖ＿ｉｎｆｏデータ構造が、クラスタ内の装置の物理的関連を表すｄｅｖ＿ｉｎｆｏツリーを含み、ルート・ノードとツリーのリーフ・ノードが、グローバル・ファイル・システムによって維持されている装置の一つを表す、装置のグローバルな一意の物理名に対応している請求項２記載のシステム。
【請求項４】
それぞれのノードが装置レジストラの少なくとも一部とｄｅｖ＿ｉｎｆｏデータ構造とのホストとなる請求項２記載のシステム。
【請求項５】
各前記装置ドライバが、ローカル・マイナー番号を管理する各接続されたローカル装置に割り当てられ、
前記装置レジストラは、前記装置ドライバによって装置の特定の一つに割り当てられたローカル・マイナー番号がグローバルに一意であるかどうかを判定し、そのローカル・マイナー番号がグローバルに一意でないときそのローカル・マイナー番号をグローバル・マイナー番号に変換し、さらに、特定の装置に対応する装置ドライバに対してはメジャー番号を用い、特定の装置に対してはグローバル・マイナー番号を用いて特定の装置のためのグローバルに一意のｄｅｖ＿ｔ値を生成するように構成された装置ドライバ・インタフェース（ＤＤＩ）を備えている、
請求項１に記載のシステム。
【請求項６】
前記装置レジストラは、
グローバルに一意のｄｅｖ＿ｔ値をグローバルに一意の物理名にマッピングするように構成された装置ドライバインターフェースと、
グローバルに一意の物理名をグローバルに一意の論理名にマッピングするように構成されたリンク生成器と
を備えている請求項２記載のシステム。
【請求項７】
装置が所定の装置クラスに分類され、これらのクラスが次のうちの少なくとも１つ、すなわち、
個別に列挙されているノード上の特定のドライバによって管理される、少なくとも１回のオカレンスを有する装置を指定する「ｄｅｖ＿ｅｎｕｍｅｒａｔｅ」と、
ローカルにアクセスされ、各ノード上の管理側装置ドライバと１対１の関係を有する、各ノード上で使用できる装置を指定する「ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ」と、
そのようなノード上の装置ドライバによってアクセスできる装置を指定する「ｄｅｖ＿ｇｌｏｂａｌ」と、

特定のノード上のドライバからアクセスされ、装置ドライバと１対１の関係を有する、「ｄｅｖ＿ｎｏｄｅｂｏｕｎｄ」と
のうち少なくとも１つを含んでいる請求項６に記載のシステム。
【請求項８】
クラスタ内の各装置に対して、装置を管理する装置ドライバのメジャー番号と、ローカル・マイナー番号と、グローバル・マイナー番号と、装置のホストとなるノードのｉｄとを含んでいる永続的な装置構成システム（ＤＣＳ）データベースを維持する、クラスタ・ノードの１つでホストとなる装置構成システム（ＤＣＳ）をさらに備えており、
ＤＣＳとＤＣＳデータベースへの要求によって、ＤＤＩがグローバルに一意の識別子を生成する請求項７に記載のシステム。
【請求項９】
ＤＣＳがｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプの各装置に対して、同じドライバに関連する全ての装置インスタンスのための、それぞれのグローバルに一意のグローバル・マイナー番号を生成するように構成され、
特定のノードと関係されていないそれぞれの一意のグローバル・マイナー番号に関連させられているｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプの各装置に対して、ＤＣＳが、第１ＤＣＳデータベース記録と、ノードの番号のいくつかに対応する複数の第２ＤＣＳデータベース記録を生成するように構成されており、各第２ＤＣＳデータベース記録はそれぞれの一意のグローバル・マイナー番号に関連しており、第１ＤＣＳデータベース記録によってユーザがｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプ装置にローカルにアクセスでき、第２ＤＣＳデータベース記録によってユーザがあらゆるノードのｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプ装置にアクセスできるようになっている請求項８記載のシステム。
【請求項１０】
装置クラスのそれぞれ用の１つのＤＳＯをさらに備え、それぞれのＤＳＯを用いて、それぞれの装置クラスの装置インスタンスにグローバル・マイナー番号を割当て、ＤＣＳがグルーバル・マイナー番号を割り当てるためにＤＳＯを使用するように構成されている請求項９記載のシステム。
【請求項１１】
関連づけられた物理装置を有する少なくとも１つのサブセットの複数のクラスタ・ノードの各ノードにおける使用のためのコンピュータ・プログラム製品であって、
それぞれが物理装置の対応するサブセットを管理するものであって、それぞれがグローバルに一意のメジャー番号に関連づけられている複数の装置ドライバと、そして
各装置に対して、その装置がどのノードからでもアクセスされるようにする、少なくとも一つのグローバルに一意の識別子を確立するように構成された装置レジストラと
を備え、
そのグローバルに一意の識別子が、
クラスタのユーザが装置を識別するグローバルに一意の論理名と、そして
グローバル・ファイル・システムが装置を識別するグローバルに一意の物理名と
を含み、
グローバルに一意の物理名が、
装置を管理するために使用される装置ドライバに対応する第１の部分と、そして
装置を管理するために使用される装置ドライバに関してグローバルに一意であるマイナー名に対応する第２の部分と
を含み、
それぞれの装置の論理名と物理名との間の１対１のマッピングが装置レジストラによって確立される、
コンピュータ・プログラム製品。
【請求項１２】
さらに、
前記装置レジストラによって維持され、クラスタ内の装置の物理的関連を表す装置情報（

ｄｅｖ＿ｉｎｆｏ）データ構造であって、その各物理的関連は、前記グルーバル・ファイル・システムによって維持されている装置ファイルの物理名に対応している、装置情報（ｄｅｖ＿ｉｎｆｏ）データ構造を備え、
前記装置レジストラが、
接続された装置に対してグローバルに一意の装置タイプ（ｄｅｖ＿ｔ）値を決定し、
ｄｅｖ＿ｉｎｆｏデータ構造内にエントリを作成するとともに、接続された装置に対する対応する物理名を作成し、
接続した装置に対して、ｄｅｖ＿ｔ値に基づく論理名と対応する物理名を作成し、
接続された装置のｄｅｖ＿ｔ値を、接続された装置を表す装置ファイルに関連付けるように構成されている請求項１１に記載のコンピュータ・プログラム製品。

【請求項１３】
前記ｄｅｖ＿ｉｎｆｏデータ構造が、クラスタ内の装置の物理的関連を表すｄｅｖ＿ｉｎｆｏツリーを含み、ルート・ノードとツリーのリーフ・ノードが、グルーバル・ファイル・システムによって維持されている装置の一つを表す、装置のグローバルな一意の物理名に対応している請求項１２記載のコンピュータ・プログラム製品。

【請求項１４】
それぞれのノードが装置レジストラの少なくとも一部とｄｅｖ＿ｉｎｆｏデータ構造とのホストとなる請求項１２記載のコンピュータ・プログラム製品。

【請求項１５】
各前記装置ドライバが、ローカル・マイナー番号を管理する各接続されたローカル装置に割り当てられ、
前記装置レジストラは、前記装置ドライバによって装置の特定の一つに割り当てられたローカル・マイナー番号がグローバルに一意であるかどうかを判定し、そのローカル・マイナー番号がグローバルに一意でないときそのローカル・マイナー番号をグローバル・マイナー番号に変換し、さらに、特定の装置に対応する装置ドライバに対してはメジャー番号を用い、特定の装置に対してはグローバルマイナー番号を用いて特定の装置のためのグローバルに一意のｄｅｖ＿ｔ値を生成するように構成された装置ドライバ・インタフェース（ＤＤＩ）を備えている請求項１４に記載のコンピュータ・プログラム製品。

【請求項１６】
前記装置レジストラは、
グローバルに一意のｄｅｖ＿ｔ値をグローバルに一意の物理名にマッピングするように構成された装置ドライバインターフェースと、
グローバルに一意の物理名をグローバルに一意の論理名にマッピングするように構成されたリンク生成器と備えている請求項１２記載のコンピュータ・プログラム製品。

【請求項１７】
装置が所定の装置クラスに分類され、これらのクラスが次のうちの少なくとも１つ、すなわち、
個別に列挙されているノード上の特定のドライバによって管理される、少なくとも１回のオカレンスを有する装置を指定する「ｄｅｖ＿ｅｎｕｍｅｒａｔｅ」と、
ローカルにアクセスされ、各ノード上の管理側装置ドライバと１対１の関係を有する、各ノード上で使用できる装置を指定する「ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ」と、
そのようなノード上の装置ドライバによってアクセスできる装置を指定する「ｄｅｖ＿ｇｌｏｂａｌ」と、
特定のノード上のドライバからアクセスされ、装置ドライバと１対１の関係を有する、「ｄｅｖ＿ｎｏｄｅｂｏｕｎｄ」と
のうち少なくとも１つを含んでいる請求項１６に記載のコンピュータ・プログラム製品。

【請求項１８】
クラスタ内の各装置に対して、装置を管理する装置ドライバのメジャー番号と、ローカル・マイナー番号と、グローバル・マイナー番号と、装置のホストとなるノードのｉｄとを含んでいる永続的な装置構成システム（ＤＣＳ）データベースを維持する、クラスタ・

ノードの１つでホストとなる装置構成システム（ＤＣＳ）をさらに備えており、
ＤＣＳとＤＣＳデータベースへの要求によって、ＤＤＩがグローバルに一意の識別子を生成する請求項１７に記載のコンピュータ・プログラム製品。
【請求項１９】
ＤＣＳがｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプの各装置に対して、同じドライバに関連する全ての装置インスタンスのための、それぞれのグローバルに一意のグローバル・マイナー番号を生成するように構成され、
特定のノードと関係されていないそれぞれの一意のグローバル・マイナー番号に関連させられているｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプの各装置に対して、ＤＣＳが、第１ＤＣＳデータベース記録と、ノードの番号のいくつかに対応する複数の第２ＤＣＳデータベース記録を生成するように構成されており、各第２ＤＣＳデータベース記録はそれぞれの一意のグローバル・マイナー番号に関連しており、第１ＤＣＳデータベース記録によってユーザがｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプ装置にローカルにアクセスでき、第２ＤＣＳデータベース記録によってユーザがあらゆるノードのｄｅｖ＿ｅｎｕｍｅｒａｔｅタイプ装置にアクセスできるようになっている請求項１８記載のコンピュータ・プログラム製品。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は一般に、ファイル・システムを通じて装置アクセスを提供するシステムおよび方法に係り、特に、クラスタ上の装置をグローバルに可視的にするシステムおよび方式に関する。
【０００２】
【従来の技術】
複数のコンピュータを含むクラスタ上でＵｎｉｘベースのコンピュータ・アプリケーションを使用することがますます一般的になっている。クラスタ・オペレータのターゲットは、クラスタの動作を、単一のコンピュータの場合と同様にアプリケーション／ユーザに対して透過的なものにすることである。たとえば、クラスタは通常、ユーザが、ファイルがどこに存在するかにかかわらずクラスタ上のすべての従来型のファイルを見てそれにアクセスできるようにするグローバル・ファイル・システムを用意している。しかし、上記透過性はクラスタ上の装置アクセスには及ばない。
【０００３】
通常、Ｕｎｉｘベースのシステム上の装置アクセスは、装置をファイルとして扱う特殊ファイル・システム（たとえば、ＳｐｅｃＦＳ）を通して行われる。この特殊ファイル・システムは、単一のノード上でのみ動作する。すなわち、このファイル・システムでは、特定のノード上の装置を見てそれにアクセスすることができるのはそのノード上のユーザだけであり、これは、クラスタ上のグローバル装置可視性のターゲットに反する。このような制限は、様々なノード上で動作する特殊ファイル・システムの間で調整が行われないことと、装置のグローバル可視性に対処する装置命名方式がないことによるものである。次に、従来技術の装置アクセス・システムのこのような点を図１ないし４を参照して説明する。
【０００４】
図１には、中央演算処理装置（ＣＰＵ）１０２と、高速メモリ１０４と、複数の物理的装置１０６と、ＣＰＵ１０２がデータを制御しメモリ１０２および物理的装置１０６と交換できるようにする一群の物理的装置インタフェース１０８（たとえばバスまたは他の電子式インタフェース）とを含む従来型のコンピュータ・システムのブロック図が示されている。メモリ１０２は、ランダム・アクセス・メモリ（ＲＡＭ）でも、あるいはキャッシュ・メモリでもよい。
【０００５】
物理的装置１０６には、高可用性装置１１２や、プリンタ１１４や、カーネル・メモリ１１６や、通信装置１１８や、記憶装置１２０（たとえば、ディスク装置）を含めることが

できるが、これらに限らない。プリンタ１１４および記憶装置１２０は良く知られている。高可用性装置１１２には、関連する二次装置を有する記憶装置やプリンタが含まれる。そのような装置は、二次装置が、そのそれぞれの一次装置が故障したときに一次装置に代替することができるので可用性が高い。カーネル・メモリ１１６は、システム性能累計・報告統計メモリ１０２のプログラム済み領域である。通信装置１１８はモデムと、ＩＳＤＮインタフェース・カードと、ネットワーク・インタフェース・カードと、その他の種類の通信装置とを含む。装置１０６は、擬似装置１２２、すなわち実際の物理的装置に関連付けられないソフトウェア装置を含むこともできる。

【０００６】

コンピュータ１００のメモリ１０４は、オペレーティング・システム１３０と、アプリケーション・プログラム１５０と、データ構造１６０を記憶することができる。オペレーティング・システム１３０は、コンピュータ１００が動作可能であるかぎりＣＰＵ１０２内で実行され、プロセッサ１０２と、ＣＰＵ１０２内で実行中のアプリケーション１５０とのためのシステム・サービスを行う。オペレーティング・システム１３０は、Ｓｕｎ ワークステーション上で使用されるＳｏｌａｒｉｓ™オペレーティング・システムのｖ．２．６に基づいて構成され、カーネル１３２と、ファイル・システム１３４と、装置ドライバ１４０と、装置ドライブ・インタフェース（ＤＤＩ）フレームワーク１４２とを含む。ＳｏｌａｒｉｓおよびＳｕｎはそれぞれ、サンマイクロシステムズ社（Ｓｕｎ Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃ．）の商標および登録商標である。カーネル１１６は、メモリ１０４、ファイル・システム１３４、または装置１０６へのアクセスを求める要求など、アプリケーション１５０からのシステム呼出しを処理する。ファイル・システム１３４およびその装置１０６および装置ドライバ１４０との関係について図２Ａおよび２Ｂを参照して説明する。

【０００７】

図２Ａを参照するとわかるように、Ｓｏｌａｒｉｓオペレーティング・システムのｖ．２６およびそれ以前のバージョンによって使用されるファイル・システム１３４の高レベル表現が示されている。Ｓｏｌａｒｉｓでは、ファイル・システム１３４は、すべてのファイル、装置１０６、ネットワーク・インタフェース（コンピュータ１００はネットワーク化されていると仮定する）がアクセスされる媒体である。この３つの異なる種類のアクセスはそれぞれ、ファイル・システム１３４の３つの構成要素、すなわちＵｎｉｘファイル・システム１３８ｕ（ＵＦＳ）、特殊ファイル・システム１３８ｓ（ＳｐｅｃＦＳ）、ネットワーク・ファイル・システム１３８ｎ（ＮＦＳ）によって行われる。

【０００８】

Ｓｏｌａｒｉｓでは、アプリケーション１５０は最初、ファイル、装置、またはネットワーク・インタフェース（本明細書では「ターゲット」と呼ぶ）に対するオープン要求をカーネル１３２を介してファイル・システム１３４に発行することによってそのターゲットにアクセスする。ファイル・システム１３４は次いで、必要に応じてＵＦＳ１３８ｕ、ＳｐｅｃＦＳ１３８ｓ、またはＮＦＳ１３８ｎへ要求を送る。ターゲットが首尾良くオープンされた場合、ＵＦＳ、ＳｐｅｃＦＳ、またはＮＦＳは、要求されたファイル、装置、またはネットワーク・ノードにマップされているｖノード・オブジェクト１３６をファイル・システム１３４に返す。ファイル・システム１３４は次いで、ｖノード・オブジェクト１３６をファイル記述子１７４にマップし、そのファイル記述子はカーネル１３２を介してアプリケーション１５０に返される。要求側アプリケーションはその後で、ファイル記述子１７４を使用して、返されたｖノード・オブジェクト１３６に関連付けられた対応するファイル、装置、またはネットワーク・ノードにアクセスする。

【０００９】

ｖノード・オブジェクト１３６は、カーネル１３２とファイル・システム１３４との間のインタフェースとして働くｖノード／ＶＦＳインタフェースまたはレイヤ（ＶＦＳ）１７２に従って一般的な１組のファイル・システム・サービスを行う。Ｓｏｌａｒｉｓは、ｖノード・オブジェクト１３６から継承し、ＵＦＳ、ＳｐｅｃＦＳ、ＮＦＳに関連付けられ

た各ターゲットタイプについてカスタマイズされたメソッドとデータ構造も含むｉノード・オブジェクト１３６ｉ、ｓノード・オブジェクト１３６ｓ、ｒノード・オブジェクト１３６ｒも備える。これらのクラス１３６ｉ、１３６ｓ、１３６ｒは、ｖノード１３６とそのそれぞれのターゲットとの間の低レベル・インタフェースを形成する。したがって、ＵＦＳ、ＳｐｅｃＦＳ、またはＮＦＳがｖノード・オブジェクトを返すとき、そのオブジェクトは、実際のターゲット動作を実行する対応するｉノード、ｓノード、またはｒノードに関連付けられる。Ｓｏｌａｒｉｓファイル・システムの一般的な性質について論じたが、次に、Ｓｏｌａｒｉｓで使用されるファイル・ベースの装置アクセス方法について論じる。

【００１０】

図２Ｂを参照するとわかるように、Ｓｏｌａｒｉｓアプリケーション１５０は通常、それがオープンする必要のある装置の論理名１６６を使用してファイル・システム１３４（カーネル１３２を介して）に装置アクセス要求を発行する。たとえば、アプリケーション１５０はコマンドｏｐｅｎ（／ｄｅｖ／ｄｓｋ／ｄｉｓｋ＿ｌｏｇｉｃａｌ＿ａｄｄｒｅｓｓ）を用いてＳＣＳＩ装置へのアクセスを要求することができる。論理名／ｄｅｖ／ｄｓｋ／ｄｉｓｋ＿ｌｏｇｉｃａｌ＿ａｄｄｒｅｓｓは、オープンする装置が特定の論理アドレスにあるディスクであることを示す。Ｓｏｌａｒｉｓでは、ＳＣＳＩディスクの論理アドレスは「ｃ０ｔ０ｄ０ｓｘ」であってよく、この場合、「ｃ０」はＳＣＳＩコントローラ０を表し、ｔ０はターゲット０を表し、ｄ０はディスク０を表し、ｓｘは特定のディスクのｘ番目のスライスを表す（ＳＣＳＩディスク・ドライバは最大で８つのスライスを有することができる）。

【００１１】

論理名は、１つのリンク・ジェネレータ１４４、すなわちＤＤＩフレームワーク１４２の１つのユーザ空間拡張子によって割り当てられ、装置の接続時に装置のドライバ１４０から供給される情報とＤＤＩフレームワーク１４２によって生成される装置の対応する物理名とに基づくものである。特定の装置ドライバ１４０のインスタンスがノード１００に接続されると、ＤＤＩフレームワーク１４２はそのドライバ１４０の接続ルーチンを呼び出す。ドライバ１４０は次いで、そのインスタンスに関連付けることのできる各装置についてＤＤＩフレームワーク１４２のｄｄｉ＿ｃｒｅａｔｅ＿ｍｉｎｏｒ＿ｎｏｄｅｓメソッド１４６に固有のローカル識別子を割り当てそれを呼び出す。通常、この固有のローカル識別子はマイナー名（たとえば、「ａ」）およびマイナー番号（たとえば、「２」）を構成する。ｄｄｉ＿ｃｒｅａｔｅ＿ｍｉｎｏｒ＿ｎｏｄｅｓメソッド１４６は、呼び出されると、所与の装置を表すリーフ・ノードをＤｅｖＩｎｆｏツリー１６２内に作成する。たとえば、ＳＣＳＩドライブ（すなわち、インスタンス）は最大で８つのスライス（すなわち、デバイス）を有することができるので、ローカルＳＣＳＩドライバ１４０は８つのスライスのそれぞれに固有のローカル識別子を割り当て、ローカル識別子を用いてｄｄｉ＿ｃｒｅａｔｅ＿ｍｉｎｏｒ＿ｎｏｄｅｓメソッド１４６を最大で８回呼び出す。

【００１２】

各装置１０６には、ターゲット装置１０６に関する構成情報を与えるＵＦＳファイル１７０も関連付けられる。特定のＵＦＳファイル１７０ｉの名前は、コンピュータ上の装置の物理的位置から導かれる物理名１６８ｉと同じである。たとえば、ＳＣＳＩ装置は物理名１６８／ｄｅｖｉｃｅｓ／ｉｏｍｍｕ／ｓｂｕｓ／ｅｓｐ１／ｓｄ@ａｄｄｒ：ｍｉｎｏｒ＿ｎａｍｅを有することができ、この場合、ａｄｄｒは装置ドライバのアドレスであり、ｍｉｎｏｒ＿ｎａｍｅは装置インスタンスのマイナー名である。この名前は装置ドライバｓｄによって割り当てられる。物理名がどのように割り当てられるかについては以下で図３を参照して説明する。

【００１３】

ファイル・システム１３４は、それ自体が、ターゲット装置の論理名が与えられた場合にターゲット装置をオープンできるようにするために、論理ファイル名１６６を物理ファイル名１６８にマップする論理名空間データ構造１６４を使用する。装置１０６の物理名は

、コンピュータ・システム１００に関連付けられた装置タイプ、バス接続、コントローラ、ドライバ、装置の階層を表す装置情報（ＤｅｖＩｎｆｏ）ツリー１４０（図１に示されている）内の装置の位置から導かれる。物理名１６８によって識別される各ファイル１７０は、その属性として、ターゲット装置に固有に関連付けられた識別子またはｄｅｖ＿ｔ（装置タイプの短縮形）を含む。このｄｅｖ＿ｔ値は、ＳｐｅｃＦＳ１３８ｓを介して正しいターゲット装置にアクセスするためにファイル・システム１３４によって使用される。次に、図３を参照して、ｄｅｖ＿ｔ値がどのように割り当てられ、ＤｅｖＩｎｆｏツリー１４０がどのようにＤＤＩフレームワーク１４２によって維持されるかについて説明する。

【００１４】

図３を参照すると、コンピュータ・システム１００に関する仮説的なＤｅｖＩｎｆｏツリー１６２の図が示されている。ＤｅｖＩｎｆｏツリー１６２の各ノードは、コンピュータ１００に関連付けられた装置システムの物理的構成要素に対応する。それぞれの異なるレベルは、それぞれの異なるレベルの装置階層に対応する。上位のノードに直接接続されたノードは、上位のレベルのオブジェクトのインスタンスであるオブジェクトを表す。したがって、ＤｅｖＩｎｆｏツリーのルート・ノードは常に「Ｉ」ノードであり、この下に装置階層全体が存在する。中間ノード（すなわち、リーフ・ノードおよびリーフ親ノード以外のノード）は、ネクサス装置と呼ばれ、コントローラ、バス、ポートなどの中間構造に対応する。ＤｅｖＩｎｆｏツリーの最も低いレベルの次に装置ドライバがあり、それぞれ、１つまたは複数の装置をエクスポートまたは管理することができる。リーフ・レベルには、それぞれ、装置タイプに応じていくつかの装置インスタンスをエクスポートできる実際の装置がある。たとえば、ＳＣＳＩ装置は最大７個のインスタンスをもつことができる。

【００１５】

図３に示した仮説的なＤｅｖＩｎｆｏツリー１６２は、物理アドレスａｄｄｒ０にあるメモリ・マップされた入出力装置（ｉｏｍｍｕ）用の入出力コントローラを含むコンピュータ・システム１００を表す。ｉｏｍｍｕは、アドレスａｄｄｒ１にあるシステム・バス（ｓバス）およびアドレスａｄｄｒ２にあるＰＣＩバスなどの高速バスに接続された入出力装置とＣＰＵの対話を管理する。それぞれのアドレスａｄｄｒ３およびａｄｄｒ４にある２つのＳＣＳＩコントローラ（ｅｓｐ１およびｅｓｐ２）は、アドレスａｄｄｒ５にある非同期転送モード（ＡＴＭ）コントローラと共にｓバスに結合される。第１のＳＣＳＩコントローラｅｓｐ１は、４つのＳＣＳＩ装置インスタンス（ｄｅｖ０、ｄｅｖ１、ｄｅｖ２、ｄｅｖ３）を管理するアドレス０（＠０として表される）にあるＳＣＳＩ装置ドライバ（ｓｄ）に関連付けられる。これらの装置インスタンスはそれぞれ、単一の物理装置１０６のそれぞれのスライスに対応する。第１のＳＣＳＩコントローラｅｓｐ１は、他の物理装置１０６の複数のＳＣＳＩ装置インスタンス（図示せず）を管理するアドレス１にあるＳＣＳＩ装置ドライバ（ｓｄ）にも関連付けられる。

【００１６】

コンピュータ・システム１００と共に使用できる各装置ドライバ・タイプには所定の固有のメジャー番号が割り当てられる。たとえば、ＳＣＳＩ装置ドライバｓｄにはメジャー番号３２が割り当てられる。各装置には、単一の装置ドライバによって管理される一群の装置内で固有のマイナー番号が割り当てられる。たとえば、アドレス０にあるドライバｓｄに関連付けられた装置ｄｅｖ０、ｄｅｖ１、ｄｅｖ２、ｄｅｖ３はそれぞれ、マイナー番号０、１、２、３とマイナー名ａ、ｂ、ｃ、ｄとを有する。同様に、アドレス１にあるドライバｓｄによって管理される装置は、装置ｄｅｖ０ないしｄｅｖ３に関連付けられたマイナー番号とは異なるマイナー番号（たとえば、マイナー番号４ないし７を有する４つ）を有する。マイナー番号およびマイナー名は、新しい各装置インスタンスごとに親装置ドライバ１４０（図１）によって割り当てられる（ＳＣＳＩインスタンスが、特定のＳＣＳＩドライブでよく、ＳＣＳＩ装置がそのドライブの特定のスライスでよいことを想起されたい）。これによって所与の装置ドライバによってエクスポートされる各装置は固有のマ

イナー番号とマイナー名とを有する。すなわち、ドライバはマイナー番号－名空間を管理する。

【００１７】

各マイナー番号は、親ドライバのメジャー番号と組み合わされると、各装置を固有に識別するｄｅｖ＿ｔ値を形成する。たとえばアドレス０にあるドライバｓｂによって管理される装置ｄｅｖ０、ｄｅｖ１、ｄｅｖ２、ｄｅｖ３は、それぞれのｄｅｖ＿ｔ値（３２，０）、（３２，１）、（３２，３）、（３２，３）を有する。ＳｐｅｃＦＳ１３８ｓはｄｅｖ＿ｔ値の対応する装置へのマッピングを維持する。そのため、ＳｐｅｃＦＳへのすべての装置オープン要求は、固有のｄｅｖ＿ｔ値を使用してオープンすべき装置を識別する。

【００１８】

装置へのＤｅｖＴｒｅｅパスはその装置の物理名を形成する。たとえば、装置ｄｅｖ０の物理名は文字列／ｄｅｖｉｃｅｓ／ｉｏｍｍｕ@ａｄｄｒ０／ｓｂｕｓ@ａｄｄｒ１／ｅｓｐ１@ａｄｄｒ３／ｓｄ@０：ａによって与えられ、この場合、ｓｄ@０：ａは、アドレス０にあるｓｄドライバによって管理されマイナー名がａ：である装置、すなわち装置ｄｅｖ０を指す。この物理名は、対応する装置にアクセスするのに必要なすべての情報を保持する（ｓノードに対応する）特殊ファイル１７０（図２に示されている）を識別する。特に、各特殊ファイル１７０の属性は、対応する装置に関連付けられたｄｅｖ＿ｔ値を保持する。

【００１９】

前述のように、リンク・ジェネレータ１４４は、リンク・ジェネレータによって管理される装置に適用できる１組の規則に従って装置の物理名から装置の論理名を生成する。たとえば、アドレス０にあるドライバｓｄによって管理される装置ｄｅｖ０の場合、ＳＣＳＩ装置のリンク・ジェネレータは論理名／ｄｅｖ／ｄｓｋ／ｃ０ｔ０ｄ０ｓ０を生成することができる。この場合、ｃ０はコントローラｅｓｐ１@ａｄｄｒ３を指し、ｔ０はｓｄ@０ドライバによって管理される物理ディスクのターゲットｉｄを指し、ｄ０はｓｄ@０ドライバを指し、ｓ０はマイナー名ａとマイナー番号０とを有するスライスを指定する。ｓｄ@１ドライバに関連付けられた装置ｄｅｖ０には、同じリンク・ジェネレータ１４４によって論理名ｄｅｖ／ｄｓｋ／ｃ０ｔ１ｄ１ｓ４が割り当てられる。２つのｄｅｖ０装置が、ターゲット値、ディスク値、スライス値の差によって区別される論理名を有することに留意されたい。次に、図４を参照して、現在Ｓｏｌａｒｉｓでこのインフラストラクチャをどのように使用して、コンピュータ１００上に存在する特定の装置をオープンできるようにしているかを説明する。

【００２０】

図４を参照すると、アプリケーション１５０の要求に応じて装置をオープンする際に様々なオペレーティング・システム構成要素によってコンピュータ１００のメモリ１０４で実行される動作の流れ図が示されている。メモリ１０４は、アプリケーション１５０が実行されるユーザ空間１０４Ｕと、オペレーティング・システム構成要素が実行されるカーネル空間１０４Ｋに分割される。この図では、動作が行われる順序と、各動作の送信元およびターゲットである装置が１組のラベル付き矢印で示されている。必要に応じて、破線は、参照が渡されるオブジェクトを示す。メモリ１０４の図の隣に、ラベル付き矢印に関連する各動作が定義されている。動作はメッセージまたは関数呼出しとして定義され、その場合、メッセージ名の後に、処理すべきデータまたは受信側エンティティによって返されるデータが続く。たとえば、メッセージ（４－１）「ｏｐｅｎ（ｌｏｇｉｃａｌ＿ｎａｍｅ）」は、「ｌｏｇｉｃａｌ＿ｎａｍｅ」によってユーザ空間１０４Ｕ内に表された装置をオープンすることをカーネル１３２に要求する、アプリケーション１５０から発行されるメッセージである。この特定の例では、アプリケーションは装置ｄｅｖ２をオープンすることを要求する。

【００２１】

オープン・メッセージ（４－１）を受信した後、カーネル１３２はメッセージ（４－２）「ｇｅｔ＿ｖｎｏｄｅ（ｌｏｇｉｃａｌ＿ｎａｍｅ）」をファイル・システム１３４に発

行する。このメッセージは、カーネル１３２がオープン動作を完了するのに必要な装置ｄｅｖ２のｖノードを返すようファイル・システム１３４に要求する。これに応答して、ファイル・システム１３４は、論理名空間１６４を使用して論理名１６６を対応する物理名１６８に変換する。ファイル・システム１３４は次いで、物理名によって指定されたファイルを見つけ、そのファイルの属性から対応する装置のｄｅｖ＿ｔ値を判定する。ファイル・システム１３４は、ｄｅｖ＿ｔ値を得た後、メッセージ（４－３）「ｇｅｔ＿ｖｎｏｄｅ（ｄｅｖ＿ｔ）」をＳｐｅｃＦＳ１３８ｓに発行する。このメッセージは、装置ｄｅｖ２にリンクされたｖノードの参照を返すことをＳｐｅｃＦＳ１３８ｓに要求する。ＳｐｅｃＦＳ１３８ｓは、メッセージ（４－３）を受信した後、要求されたｖノード１３６と、ｖノード１３６を装置ｄｅｖ２にリンクするｓノード１３６ｓとを作成し、ｖノード１３６の参照（４－４）をファイル・システム１３４に返す。ファイル・システム１３４は次いで、ｖノード参照をカーネルに返す（４－５）。

【００２２】

カーネル１３２は、ｖノード参照を得た後、ｖノード１３６に関連付けられた装置ｄｅｖ２をオープンすることを求める要求（４－６）をＳｐｅｃＦＳ１３８ｓに発行する。ＳｐｅｃＦＳ１３８ｓは、装置ｄｅｖ２を管理することが知られているドライバ２にオープン・コマンド（４－７）を発行することによってこの要求を満たそうとする。ドライバ２は、装置ｄｅｖ２をオープンできる場合、オープン動作が成功したことを示すｏｐｅｎ＿ｓｔａｔｕｓメッセージ（４－８）を返す。そうでない場合、ドライバ２は同じメッセージ（４－８）で失敗表示を返す。ＳｐｅｃＦＳ１３８ｓは次いで、同様な状況メッセージ（４－９）をカーネル１３２に直接返す。メッセージ（４－９）で「成功」が返されたと仮定した場合、カーネル１３２は、ファイル記述子、すなわち装置ｄｅｖ２にリンクされたｖノード１３６のユーザ空間表現をアプリケーション１５０に返す（４－１０）。アプリケーション１５０は、ファイル記述子を得た後、ファイル・システム動作を使用してカーネル１３２およびファイル・システム１３４を介して装置ｄｅｖ２にアクセスすることができる。たとえば、アプリケーション１５０は、返されたファイル記述子に関する読取り要求を発行することによって装置ｄｅｖ２からの入力データを実行する。このようなファイル・システム・コマンドは次いで、ＳｐｅｃＦＳ１３６ｓと、装置ｄｅｖ２を管理するｖノード・オブジェクト１３６およびｓノード・オブジェクト１３６ｓとによって実際の装置コマンドに変換される。

【００２３】

したがって、Ｓｏｌａｒｉｓでは、コンピュータ・システム１００のユーザは、そのシステム１００上の装置に比較的容易にアクセスすることができる。しかし、Ｓｏｌａｒｉｓによって使用される方法では、いくつかの異なるコンピュータがクラスタの一部として構成されているときでも、ユーザがコンピュータを介して装置に透過的にアクセスすることはできない。すなわち、第１のコンピュータ上で実行されているアプリケーションが、Ｓｏｌａｒｉｓを使用して第２のコンピュータ上の装置を透過的にオープンすることはできない。

【００２４】

Ｓｏｌａｒｉｓの現行のバージョンがマルチコンピュータの場合に透過的な装置アクセスを行うことができない理由は、ｄｅｖ＿ｔ番号およびマイナー番号が装置が接続されるときに割り当てられる現行の方法に関するものである。再び図３を参照するとわかるように、、コンピュータ１００に装置が接続されるたびに、装置の関連するドライバによって制御される１組の装置内で固有であり、したがって、ドライバのメジャー番号と組み合わされたときにコンピュータ１００の固有のｄｅｖ＿ｔ値にマップすることのできるマイナー番号をドライバがその装置に割り当てる。しかし、同じ装置およびドライバが第２のコンピュータ上に設けられている場合、そのドライバおよび装置には、同一ではないにしても同様な１組のメジャー番号およびマイナー番号ならびにｄｅｖ＿ｔ値が割り当てられる。たとえば、両方のコンピュータがＳＣＳＩドライバｓｄ（メジャー番号＝３２）と、ＳＣＳＩドライバｓｄによって管理される４つのＳＣＳＩ装置インスタンスを有する場合、各

ドライバｓｄはそのローカルな１組のＳＣＳＩ装置に同じ１組のマイナー番号を割り振る（たとえば、両方の組が０ないし３のマイナー番号を有する）。したがって、装置はそのｄｅｖ＿ｔ値に従ってアクセスされるので、第１のノード・アプリケーションが第２のノード上のＳＣＳＩディスクをオープンする必要がある場合、そのアプリケーションは、どちらのコンピュータ・システム上のＳｐｅｃＦＳに対してもＳＣＳＩディスクを明確に識別することができない。

【００２５】

【発明が解決しようとする課題】

したがって、アプリケーションが、どこで実行されているかにかかわらず、コンピュータ・クラスタのノード上に存在する装置に透過的にアクセスできるようにするファイル・ベースの装置アクセス・システムが必要である。

【００２６】

【課題を解決するための手段】

要約すると、本発明は、クラスタ内で使用され、ローカル装置名を、クラスタ・ノードのいずれかでランしているアプリケーションから他のクラスタ・ノードに位置する装置にアクセスできる、グローバルで一意の論理名にマッピングするシステムおよび方法にある。

【００２７】

特に、好ましい実施形態には、クラスタのグローバル・ファイル・システム、装置（デバイス）・ドライバ・インタフェース（ＤＤＩ）および装置（デバイス）情報ツリーが含まれている。装置ドライブ・インタフェースは、少なくとも予め定められた装置クラスのサブセットに関連付けられたインスタンスに対して、ローカル装置名がグローバルに一意であるかどうか判断する。ローカル装置名がグローバルに一意ではない場合、ＤＤＩはそれぞれのローカル・インスタンスに対してグローバルに一意な識別子を決める。装置ドライバ・インタフェースによって維持される装置情報ツリーは、クラスタ内の装置の物理的関連を表す。装置情報ツリーのルート・ノードとリーフ・ノードの間の各パスは、装置インスタンスのそれぞれ１つを表すグローバル・ファイル・システムによって維持される装置ファイルの物理名に対応する。リンク・ジェネレータは、各インスタンスに対して、そのインスタンスの対応する物理名にマッピングされるグローバルに一意の識別子に基づいて論理名を生成し、特定のインスタンスのグローバルに一意の識別子を、その特定のインスタンスを表す装置ファイルに関連付ける。

【００２８】

このフレームワークを与えられると、クラスタのユーザは、アクセスされる装置をその論理名で識別するアクセス要求をグローバル・ファイル・システムに出すことにより、任意の装置にアクセスできる。グローバル・ファイル・システムは、論理名を対応する物理名に変換し、物理名によって識別される装置ファイルを介して装置のグローバルに一意の識別子にアクセスし、そのグローバルに一意の識別子によって識別される装置にアクセスして、アクセス要求を解決する。

【００２９】

好ましい実施形態においては、グローバル・ファイル・システムは、１つのネットワーク・ノード上でしか稼動しない可能性があり、各ノードが、装置ドライバ・インタフェースと装置情報ツリーのインスタンスにそれぞれ対応する。本発明の実施形態においては、グローバル・ファイル・システムとそれぞれの装置ドライバ・インタフェースとの間のリンクは、プロキシ・ファイル・システムによって提供される。好ましい実施形態はまた、各々がローカル装置名をインスタンスに割り当てることのできる装置を管理するための複数の装置ドライバを含むこともある。これらの装置ドライバは、各種の装置ドライバ・インタフェースと共設される。

【００３０】

好ましい予め定められた装置クラスのセットは、次を含む。すたわち、

「ｄｅｖ＿ｅｎｕｍｅｒａｔｅ」・・特定のドライバによって管理される少なくとも１つのインスタンス（それぞれは、個別に列挙されている特定のノード上の特定のドライバに

よって管理される）で装置を指定。
「ｄｅｖ＿ｎｏｄｅｓｐｃｉｆｉｃ」・・ローカルにアクセスされ、各ノード上の管理ドライバと１対１の関係を有している、各ノード上で利用可能な装置を指定。
「ｄｅｖ＿ｇｌｏｂａｌ」・・任意のノードのドライバからアクセスできる装置を指定。
「ｄｅｖ＿ｎｏｄｅｂｏｕｎｄ」・・特定のノード上のドライバだけから通常アクセスされ、そのドライバと１対１の関係を有している装置を指定。

【００３１】

本発明は、それぞれの装置クラスのインスタンスのローカル装置名を、対応するグローバルに一意の識別子および論理名に変換するための一連の規則（ルール）を定義するものである。特に、ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置のローカル装置名は、装置名およびローカルに一意のローカル・マイナー番号の組合せを含み、グローバルに一意の識別子は、グローバル・マイナー番号を含み、論理名は、装置名とグローバル・マイナー番号の組合せを含んでいる。同様に、ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置のローカル装置名と論理名はそれぞれ、装置名から形成され、ｄｅｖ＿ｇｌｏｂａｌ装置のローカル装置名と論理名はそれぞれ装置名から形成される。

【００３２】

本発明はまた、クラスタ・ノードの１つのノードに装置構成システム（ＤＣＳ）が組み込まれており、その１つのノードには、クラスタ内の各装置に対して、装置名、論理装置を管理する装置ドライバのメジャー番号と、グローバル・マイナー番号、および論理装置のホストとなるノードのホストｉｄとを含んでいる永続的な装置構成システム（ＤＣＳ）データベースが維持されている。ＤＤＩは、ＤＣＳデータベースを使用してＤＤＩのために装置それぞれのグローバル・マイナー番号を生成するＤＣＳの支援によって、各論理装置のグローバルに一意の識別子、論理名および物理名を生成する。

【００３３】

本発明はまた、クラスタ上の装置をグローバルに可視的にするためのシステムでもあり、クラスタには装置が接続される複数のノードが含まれる。好ましいシステムは、少なくとも１つのグローバルに一意の識別子を各装置に設定して、その装置がどのノードからもアクセスできるようにする装置レジストラを含む。このシステムはクラスタ上でランするグローバル・ファイル・システムを含むことができ、少なくとも１つのグローバルに一意の識別子には、クラスタのユーザが装置を識別するグローバルに一意の論理名と、グローバル・ファイル・システムが装置を識別するグローバルに一意の物理名とが、備えられている。好ましい実施形態においては、装置レジストラによって設定される、各装置の論理名と物理名との間の１対１のマッピングがある。

【００３４】

本発明はまた、クラスタ内の装置の物理的関連を表す装置レジストラによって維持される装置情報（ｄｅｖ＿ｉｎｆｏ）データ構造も含み、当該物理的関連のそれぞれは、装置のそれぞれを表しグローバル・ファイル・システムによって維持されている装置ファイルの物理名に対応している。このフレームワークを与えられると、装置レジストラは好ましい実施形態では、接続された装置に対してグローバルに一意の装置タイプ（ｄｅｖ＿ｔ）値を決定し、ｄｅｖ＿ｉｎｆｏデータ構造にエントリおよび対応する物理名を接続された装置に対して作成し、接続された装置に対してｄｅｖ＿ｔ値に基づく論理名と、対応する物理名とを作成し、そして接続された装置のｄｅｖ＿ｔ値を、接続された装置を表す装置ファイルに関連付ける。

【００３５】

本発明の他の目的および特徴は、以下の詳細な説明および添付の特許請求の範囲を図面と共に検討したときにより容易に明らかになろう。

【００３６】

【発明の実施の形態】

図５を参照すると、本発明を実施することのできるコンピュータ・クラスタ２００のブロック図が示されている。クラスタ２０１は、関連する装置１０６とアプリケーション１５

０とを有する複数のノード２０２を含む。図１と同様に、装置１０６は高可用性装置１１２と、プリンタ１１４と、カーネル・メモリ１１６と、通信装置１１８と、記憶装置１２０とを含むことができる。本発明の議論の都合上、グローバル・ファイル・システム２０６は、クラスタ２０１上に記憶されているすべてのファイル用の単一のグローバル・ファイル空間を維持し、１つのノード２０２上で実行される。グローバル・ファイル・システム２０６は装置１０６の少なくとも２つの表現をサポートする。物理名空間（ＰＮＳ）表現３０５は、カーネル空間からアクセスできるものであり、それぞれのノード２０２上の装置１０６の物理的構成に対応する。論理名空間（ＬＮＳ）表現３０４は物理名空間３０５のユーザ空間バージョンであり、すなわち論理名空間３０４内の各エントリは物理名空間３０５内の対応するエントリにマップされる。本発明は、このグローバル・ファイル・システム２０６の多数の点を修正し、アプリケーション１５０による装置１０６への透過的なグローバル・アクセスを可能にする。クラスタ２０１は、本発明の重要な構成要素である装置構成システム（ＤＣＳ）２０８を備えるノード２０４も含む。

【００３７】

他の実施形態には、それぞれ、それ自体の物理名空間および論理名空間を維持する、任意の数のグローバル・ファイル・システム２０６がある。そのような環境では、特定の装置は、グローバル・ファイル・システム２０６およびそれに関連付けられた物理名空間および論理名空間のみを通してアクセスされる。

【００３８】

上記で図１ないし４を参照して説明したように、従来型のＳｏｌａｒｉｓ装置アクセス・システムでは、透過装置アクセスは単一のコンピュータ・システム内でしかできない。従来技術においてアクセスすべき装置のｄｅｖ＿ｔ値にファイル・システムによってマップされた論理名を生成する方法のある態様は、現行の装置アクセス・システムのクラスタへの拡張には適合しない。たとえば、数組の装置１０６－１、１０６－２がそれぞれ、４つのＳＣＳＩディスク・ドライブを含むと仮定すると、現在使用されている論理命名システムでは、それぞれの異なるノード１０６－１、１０６－２上のそれぞれの異なるドライブが同じｄｅｖ＿ｔ値を有する。この場合、アプリケーション１５０－１がノード２０２－２上の特定の１つのディスク・ドライブに透過的にアクセスすることは不可能である。次に、本発明がどのように、そのような透過的なグローバル装置アクセスを行うかについて説明する。

【００３９】

図６を参照すると、１つのノード２０２と、ＤＣＳ２０８を備えるノード２０４の他の詳細が示されている。この図では、ファイル・システム２０６は、１つの特定のノード２０２－２上に存在する場合と同様には示されていない。各ノード２０２は、内部にオペレーティング・システム（ＯＳ）ルーチン／オブジェクト２４０およびデータ構造３００が定義されているメモリ２３０を含む。ＯＳルーチン２４０は、オペレーティング・システム・カーネル２４２と、プロキシ・ファイル・システム（ＰｘＦＳ）２４４と、特殊ファイル・システム２５８と、装置ドライバ・フレームワーク（ＤＤＩ）２７０と、１組のサーバ・オブジェクト（ＤＳＯ）と、装置ドライバ２８０とを含む。

【００４０】

前述のように、カーネル２４２は、メモリ２３０、ファイル・システム２０６、または装置１０６へのアクセスを求める要求など、アプリケーション１５０からのシステム呼出しを処理する。カーネル２４２は、グローバル装置アクセスをサポートするように本発明によって修正されているのでカーネル１３２（図１）とは異なる。プロキシ・ファイル・システム（ＰｘＦＳ）２４４は、ＳｏｌａｒｉｓＰＸＦＳファイル・システムに基づくものであるが、カーネル２４２と同様に、本明細書ではグローバル装置アクセスをサポートするように修正される。ＰｘＦＳ２４４は、ノード２０２－ｉ内のアプリケーション１５０－ｉがいくつかの異なるノード２０２を介してファイル・システム２０６とシームレスに対話できるようにするオブジェクトの集合を含む。ＰｘＦＳオブジェクトには、ＰｘＦＳクラスタ２４６と、ＰｘＦＳサーバ２４８と、ｆ＿ｏｂｊｓ（ファイル・オブジェクト）

２５０と、ｖノード（仮想ファイル・ノード）２５２と、ｓノード（特殊ファイル・ノード）２５４と、ｐｘ＿ｖノード（プロキシｖノード）２５６とが含まれる。これらのオブジェクトはそれぞれ、ファイル・システム２０６の動作に応答してＰｘＦＳ２４４の必要に応じて作成されるので図６では任意選択（ｏｐｔ）と示されている。

【００４１】

ＤＤＩフレームワーク２７０（以下では「ＤＤＩ」と呼ぶ）も、従来技術（図１）を参照して説明したＤＤＩフレームワーク１４２に類似している。しかし、ＤＤＩフレームワーク２７０は本発明では、ＤＣＳ３６０と対話し、いくつかの異なるノード２０２上でそれらのノードからアクセスすることのできる装置１０６との互換性を有する物理名および論理名を生成する。ＤＤＩ２７０は、ローカル・ノード２０２に新しい装置が接続されるたびに呼び出される接続メソッド２７２を含む。従来技術の接続メソッドとは異なり、接続メソッド２７２は、ＤＣＳ３６０のサービスを使用して、接続された各装置ごとにグローバルに整合する物理名を作成するように構成される。ＤＤＩフレームワーク２７０は、対応する物理名から固有の論理名を生成するリンク・ジェネレータ２７４の集合も含む。それぞれの異なる種類の装置１０６ごとにそれぞれの異なる種類のリンク・ジェネレータがある。したがって、接続ルーチン２７２およびリンク・ジェネレータ２７４はそれぞれ、装置１０６をそれぞれ、カーネル・レベルおよびユーザ・レベルでグローバルに見えるようにする物理名空間及び論理名空間を構築する。

【００４２】

本発明は、それぞれ、装置１０６の特定のクラス３１２を管理する、１組のＤＳＯ２９０をクラスタ２００の各ノード上に含む。それぞれの装置クラスは、特定の装置１０６をオープンすることを求めるユーザの要求を一般的には透過的なグローバル装置アクセス・システムにより、具体的にはＤＣＳ３７２によって満たさなければならない場合の特殊性に対処する本発明の新しい形態である。好ましい実施形態には、ｄｅｖ＿ｅｎｕｍｅｒａｔｅ３１４、ｄｅｖ＿ｎｏｄｅ＿ｓｐｅｃｉｆｉｃ３１６、ｄｅｖ＿ｇｌｏｂａｌ３１８、ｄｅｖ＿ｎｏｄｅｂｏｕｎｄ３２０の４つの装置クラスと、ＤＳＯ＿ｅｎｕｍ２９２、ＤＳＯ＿ｎｏｄｅｓｐｅｃ２９４、ＤＳＯ＿ｇｌｏｂａｌ２９６、ＤＳＯ＿ｎｏｄｅｂｏｕｎｄ２９８の４つの対応するＤＳＯ２９０がある。

【００４３】

ｄｅｖ＿ｅｎｕｍｅｒａｔｅクラス３１４は、各装置が接続されるときに関連するドライバ２８０によって列挙される特定のノード２０２で複数のインスタンスを有することのできる装置１０６（たとえば、複数の記憶装置１２０）に関連付けられる。ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃクラス３１６は、ノード当たり１つのインスタンスしかなく、その結果ドライバ２８０によって列挙されることがない装置１０６（たとえば、カーネル・メモリ１１６）に関連付けられる。ｄｅｖ＿ｇｌｏｂａｌクラス３１８は、各ノード上に存在するドライバを使用してローカルまたはリモートにアクセスできる装置１０６（たとえば、通信装置１１８）のクラスである。ｄｅｖ＿ｎｏｄｅｂｏｕｎｄクラスは、特定のノード上のドライバを使用した場合にのみアクセスすることができる装置（たとえば、ＨＡ装置１１２）に使用される。

【００４４】

ドライバ２８０は、接続中の各オブジェクトについての装置クラス情報３１２が得られるときにはそれを含む追加の構成情報を報告することを除いてドライバ１４０に類似している。

【００４５】

データ構造３００はＤｅｖＩｎｆｏツリー３０２とｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓテーブル３０６とを含む。多数のＯＳルーチン２４０と同様に、データ構造３００は、従来技術（図１）で使用される同様な名称のデータ構造１６０に類似している。しかし、各データ構造は、本発明の動作を有効にする、従来技術との重要な違いを具体化する。具体的には、ＤｅｖＩｎｆｏツリー３０２は、クラスタ２００内の選択されたクラスの装置を見つけるために必要な追加の中間ノードを含む。ＤｅｖＩｎｆｏツリーで表される物理名空間

３０５を変更したため、論理名空間３０４も従来技術の論理名空間１６４とは異なる。最後に、ｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓテーブル３０６は、従来技術で使用されるｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓテーブルとは異なり追加のフィールドを含む。たとえば、本発明のｄｄｉ＿ｍｉｎｏｒ　ｎｏｄｅｓテーブルは、ｇｌｏｂａｌ＿ｍｉｎｏｒ＿ｎｕｍｂｅｒフィールド３０８と、ｌｏｃａｌ＿ｍｉｎｏｒ＿ｎｕｍｂｅｒフィールド３１０と、（ｄｅｖｉｃｅ）ｃｌａｓｓフィールド３１２（前述）を含む。従来技術のｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓテーブルには、フィールド３０８とフィールド３１２のどちらも含まれていない。

【００４６】

ノード２０４は、ＯＳルーチン／オブジェクト３４０およびデータ構造３７０が定義されているメモリ３３０を含む。ＯＳルーチン／オブジェクト３４０は、装置構成システム（ＤＣＳ）３６０と、ＤＣＳ上のｍａｐ＿ｍｉｎｏｒメソッドと、前述のＤＳＯと同一の１組のＤＳＯ２９０とを含む。データ構造３７０はＤＣＳデータベース３７２を含む。

【００４７】

ＤＣＳ３６０は、従来技術とはまったく異なるものであり、少なくとも２つの重要な機能を果たす。第１に、ＤＣＳ３６０はＤＤＩ２７０と共に働き、新たに接続された装置に、この装置にグローバルにかつ透過的にアクセスできるようにするグローバル・マイナー番号を割り当てる。第２に、ＤＣＳ３６０はファイル・システム２０６と共に働き、ＰＸＦＳ２４４アプリケーション１５０が、接続された装置１０６に透過的にアクセスできるようにする。ＤＣＳ＿ｄａｔａｂａｓｅ３７２は、ＤＣＳ３７２によって生成されたすべての重要な結果を常に保持する。ＤＣＳ３６０のこの２つの態様について以下でそれぞれ、図７Ａおよび７Ｂならびに図８Ａおよび８Ｂを参照して説明する。

【００４８】

図７Ａを参照すると、ノード２０２内のＤＤＩフレームワークおよびノード２０４内のＤＣＳ３６０が、ノード２０２に接続される装置３８０の適切なｄｅｖ＿ｔ値、論理名、物理名を規定する動作を示す流れ図が示されている。ＤＤＩ２７０、リンク・ジェネレータ２７４、ＤＣＳ３６０、それらの拡張機能は全体として、クラスタ２００用の装置レジストラとして働く。動作およびメッセージは図４Ａの場合と同様に示されている。流れ図に表した動作を説明する前に、本発明によって管理されるいくつかの名前空間について図７Ｂを参照して説明する。

【００４９】

図７Ｂを参照すると、２つのノード２０２－１、２０２－２を含む例示的なクラスタに本発明で使用されるマイナー名／番号空間３０７、物理名空間３０５、論理名空間３０４の概念図が示されている。後述のように、ノード２０２に装置１０６を接続するたびに、装置のドライバが装置にローカル・マイナー番号３０７＿ｎｕｍおよびローカル・マイナー名３０７＿ｎａｍｅを割り当てる。ＤＤＩ２７０はこの情報を使用して、グローバルに一意のマイナー番号を生成し、装置１０６のグローバルに一意の物理名３０５＿ｎａｍｅを形成する。物理名３０５＿ｎａｍｅはクラスタの装置階層内の装置を特定する。次いで、リンク・ジェネレータ２７４は物理名３０５＿ｎａｍｅをグローバルに一意の論理名３０４＿ｎａｍｅにマップする。ＤＤＩ２７０－１、２７０－２およびリンク・ジェネレータ２７４－１、２７４－２がそれぞれ、協働して共通のグローバル物理名空間３０５およびグローバル論理名空間３０４を生成することに留意されたい。これとは対照的に、各ドライバはそのノード２０２に対してマイナー名／番号空間しか生成しない。したがって、本発明はローカルのマイナー名／番号をグローバル物理名および論理名にマップする。これらのグローバル名前空間はファイル・システム２０６の一部である。したがって、任意のノード２０２上のアプリケーション１５０は、ファイル・システム２０６を使用して、クラスタ２００上のすべての装置１０６を、それらが単一のコンピュータ上に配置されている場合と同様に見てそれらにアクセスすることができる。フレームワークを形成する名前空間について説明したが、次に、本発明について図７Ｂを参照して説明する。

【００５０】

図７Ａを参照するとわかるように、ノード２２０に装置１０６を接続すると、ＤＤＩ２７０はドライバ２８０に接続メッセージ（７－１ａ）を発行する。これに対して、ドライバ２８０は、接続されたインスタンスに関連付けられた各装置のＤＤＩ２７０にｃｒｅａｔｅ＿ｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓメッセージ（７－１ｂ）を発行する。ｃｒｅａｔｅ＿ｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓメッセージ（７－１ｂ）は、装置３８０の構成を示し、適切な装置ドライバ２８０によって割り当てられたローカル・マイナー番号（ｍｉｎｏｒ＿ｎｕｍ）３８２およびｍｉｎｏｒ＿ｎａｍｅ３８４と、１つのクラス３１２から選択されたｄｅｖｉｃｅ＿ｃｌａｓｓ３８６とを含む。たとえば、装置がノード２０２に接続された第３のＳＣＳＩディスク・ドライブである場合、ｍｉｎｏｒ＿ｎｕｍ、ｍｉｎｏｒ＿ｎａｍｅ、ｃｌａｓｓはそれぞれ、「３」、「ａ」（それがその装置上の第１のスライスであることを示す）、「ｄｅｖ＿ｅｎｕｍｅｒａｔｅ」であってよい。

【００５１】

ｃｒｅａｔｅ＿ｍｉｎｏｒ＿ｎｏｄｅｓメッセージ（７－１ｂ）に応答して、ＤＤＩ２７０は、ｌｏｃａｌ＿ｍｉｎｏｒ＿ｎｕｍフィールド３１０をｍｉｎｏｒ＿ｎｕｍ値３８２（７－２）に等しく設定することによってｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓテーブル３８０を更新する。ＤＤＩ２７０は次いで、ＤＣＳ３６０にｄｃ＿ｍａｐ＿ｍｉｎｏｒメッセージ（７－３）を発行し、装置３８０の適切なグローバル・マイナー番号３８８を返すことをＤＣＳ３６０に要求する。前の文の「適切な」の意味は装置クラスに依存する。すなわち、ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置およびｄｅｖ＿ｎｏｄｅｂｏｕｎｄ装置は固有のグローバル・マイナー番号３８８を必要とするが、ｄｅｖ＿ｇｌｏｂａｌ装置およびｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置は必要としない。ｄｃ＿ｍａｐ＿ｍｉｎｏｒメッセージ（７－３）は、（１）「ｇｍｉｎｏｒ」、すなわちＤＣＳ３６０によって生成されたグローバル・マイナー番号３８８のリターン・フィールドと、（２）装置ドライバ２８０によって生成されたローカル・マイナー番号３８４を保持する「Ｉｍｉｎｏｒ」と、（３）装置ドライバ２８０によって生成された装置クラス３８６を保持する「ｃｌａｓｓ」の３つのフィールドを含む。ｍａｐ＿ｍｉｎｏｒメッセージ（７－３）に応答して、ＤＣＳ３６０は、メッセージ（７－３）で識別されたクラスのローカルＤＳＯ２９０に同様なｄｓ＿ｍａｐ＿ｍｉｎｏｒメッセージ（７－４）を発行する。

【００５２】

ＤＳＯ２９０は特に、接続中の装置に割り当てるべきグローバル・マイナー（ｇｍｉｎ）番号３８８を判定する。ｇｍｉｎ番号をどのように割り当てるかは装置のクラス３８６に依存する。たとえば、列挙される各装置は特定のノードでアクセスしなければならないので、ｄｅｖ＿ｅｎｕｍｅｒａｔｅクラス３１４のＤＳＯ２９２は、各ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置にクラスタ全体にわたって一意のｇｍｉｎ番号３８８を割り当てる。これに対して、どのノードでｄｅｖ＿ｇｌｏｂａｌ装置にアクセスするかは重要ではないので、ｄｅｖ＿ｇｌｏｂａｌクラス３１８のＤＳＯ２９６はこのような装置に同じｇｍｉｎ番号を割り当てる。他のクラスの場合、ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃクラス３１６のＤＳＯ２９４は、そのクラスの各装置に同じ非ヌルｇｍｉｎ番号を割り当て、ｄｅｖ＿ｎｏｄｅｂｏｕｎｄクラス３２０のＤＳＯ２９８は、そのクラスの各装置にクラスタ全体にわたって一意のｇｍｉｎ番号を割り当てる。

【００５３】

ＤＳＯ２９２、２９８は、まずＤＣＳデータベース３７２に問い合わせ、グローバル・マイナー番号がまだあるかどうかを判定することによってグローバル・マイナー番号を割り当てる。

【００５４】

ＤＣＳデータベース３７２は、常に保持され、クラスタ２００内のすべての装置１０６について、メジャー番号に関するフィールド３９０と、グローバル・マイナー番号に関するフィールド３８８と、内部（またはローカル）マイナー番号に関するフィールド３８２と、（サーバ・クラス３８６と数値３９４とを含む）装置サーバｉｄに関するフィールド３９２とを含む。マイナー名、メジャー番号、グローバル・マイナー番号、ローカル・マイ

ナー番号についてはすでに説明した。数値３９４は、接続中の装置のサーバであるノード２０２を識別する。第１のクラスのサーバのＩＤは無関係であり、第２のクラスの場合には、装置にアクセスする必要があるノードの位置と同じであるので、数値３９４の情報は、ｄｅｖ＿ｇｌｏｂａｌ装置およびｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置については任意選択である。ＤＣＳデータベース２７２の例を表１に示す。

【表１】

装置(フィールドでない)	メジャー 390	グローバル・マイナー 388	内部 マイナー 382	装置サーバ 392:	
				サーバ クラス 386	数値 394
tcp	42	0	0	dev_global	0
kmem	13	12	12	dev_node_spec	0,
disk c2t0d0s0	32	24	24	dev_enum	node id
kmem	13	1	12	dev_enum	node 0 id
kmem	13	2	12	dev_enum	node 1 id
kmem	13	3	12	dev_enum	node 2 id
kmem	13	4	12	dev_enum	node 3 id
HA devices	M	X1	X1	dev_nodebound	id

【００５５】

表１の第１行は、ｔｃｐインタフェースに関するエントリを示す。ｔｃｐインタフェースは、クラスタ２００内のあらゆるノード２０２からアクセスできるのでｄｅｖ＿ｇｌｏｂａｌ装置である。ｔｃｐ装置のメジャー番号は４２であり、この値はすべてのｔｃｐドライバに関連付けられる。ｔｃｐ装置のグローバル最小値３８８およびローカル最小値３８２ならびにサーバ数値３９４（すなわち、ｎｏｄｅ＿ｉｄ）は０に設定されることに留意されたい。これは、どのノードからｔｃｐインタフェースにアクセスするかは重要ではないからである。したがって、ＤＣＳデータベースにはクラスタ２００全体に関するｔｃｐエントリは１つしかない。表１内の第２のエントリは、省略時指定ではローカルにアクセスされるカーネル・メモリ装置に関するエントリである。このため、これはｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃクラスの装置である。メジャー番号１３は、ｋｍｅｍ装置ドライバに関連付けられる。ｋｍｅｍ装置は、特定のサーバでアクセスされるわけではないので、ヌル数値３９４を有し、かつ同一の非ヌルグローバル最小値および非ヌルローカル最小値（１２）を有する。これは、ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置では、ＤＣＳ３６０がローカル・マイナー番号と同じグローバル・マイナー番号を割り当てるに過ぎないからである。この例では、それぞれのノード２０２上に配置されたｋｍｅｍ装置どうしを区別する必要がないのでＤＣＳデータベース３７２にはｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃクラスのｋｍｅｍエントリは１つしかない。

【００５６】

第３のエントリは、ＳＣＳＩドライバがメジャー番号３２を有するＳＣＳＩディスクｃ０

ｔ０ｄ０ｔ０に関するエントリである。ＤＣＳ３６０は、ＳＣＳＩ装置にそのローカル・マイナー番号３８２（２４）と同じグローバル・マイナー番号３８８を割り当てている。これは、ＤＣＳデータベース３７２には他のＳＣＳＩ装置が表されていないからである。しかし、他のＳＣＳＩ装置ｃ０ｔ０ｄ０ｔ０を同じローカル番号（２４）で異なるノードに登録した場合、ＤＣＳ３６０はそのＳＣＳＩに異なるグローバル番号、おそらく２５を割り当てる。同じローカル番号を有するＳＣＳＩ装置どうしを区別するために、ＤＣＳデータベース３７２は完全なサーバ情報を含む。この場合、数値３９４はサーバ２０２のホストｉｄに設定される。

【００５７】

エントリ４ないし７は、ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置として登録された４つのカーネル・メモリ装置に関するエントリである。好ましい実施形態では、ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置を登録するたびに、カーネル内のすべてのノード２０２について、ＤＣＳデータベース３７２内に追加のエントリを作成することができ、これによって、ユーザはローカル・ノード以外のノード上のｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置にアクセスすることができる。したがって、４つのノード２０２－１、２０２－２、２０２－３、２０２－４があると仮定すると、ＤＣＳ２６０はこれらのノードのそれぞれごとにｄｅｖ＿ｅｎｕｍｅｒａｔｅクラスのカーネル・メモリ装置を登録することができる。他のｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置の場合と同様に、各ｋｍｅｍ装置には固有のグローバル番号が割り当てられる。ｄｅｖ＿ｅｎｕｍｅｒａｔｅ情報は、カーネル・メモリ装置をオープンすることを求める一般的な要求（たとえば、ｏｐｅｎ（／ｄｅｖｉｃｅｓ／ｋｍｅｍ））をユーザが発行するときには使用されない。ｄｅｖ＿ｅｎｕｍｅｒａｔｅ情報は、カーネル・メモリ装置をオープンすることを求める特定の要求をユーザが発行するときに使用される。たとえば、要求ｏｐｅｎ（／ｄｅｖｉｃｅｓ／ｋｍｅｍ０）によって、ユーザはノード０上のｋｍｅｍ装置をオープンすることができる。

【００５８】

最後のエントリは、一般的な高可用性（ＨＡ）装置がＤＣＳデータベース３７２内でどのように表されるかを示す。メジャー番号３９０、グローバル・マイナー番号、ローカル・マイナー番号は、ｍａｐ＿ｍｉｎｏｒ　ｎｏｄｅｓメッセージに与えられる値Ｍ、Ｘ１、Ｘ１から取り出される。数値３９４は、特定のノードに結合された装置のｉｄに設定される。この「ｉｄ」はノードｉｄではない。このｉｄは、各ＨＡサービスごとにクラスタ２００に対して固有に作成される。

【００５９】

装置３８０のグローバル・マイナー番号３８８が判定された後、適切なＤＳＯ２９０が、ＤＣＳデータベース３７２を新しい情報で更新し（７－５）、グローバル・マイナー番号３８８をＤＣＳ３６０に返す（７－６）。ＤＣＳ３７２は次いで、グローバル・マイナー番号３８８をＤＤＩ２７０に返し（７－７）、ＤＤＩ２７０がｄｄｉ＿ｍｉｎｏｒ＿ｎｏｄｅｓテーブル３０６（７－９）、論理名空間３０４、物理名空間３０５、ｄｅｖ＿ｉｎｆｏツリー３０２（７－９）を更新する。ＤＤＩ２７０は、新しいグローバル・マイナー番号３８８を書き込むことによってｄｄｉ＿ｍｉｎｏｒ　ｎｏｄｅｓテーブル３０６を更新する。名前空間３０４／３０５の更新はこれよりも複雑であり、この更新について次に説明する。

【００６０】

第１に、ＤＤＩ２７０はＤｅｖＩｎｆｏツリー３０２に新しいリーフ・ノードを追加し、ＤｅｖＩｎｆｏツリーの構造が、上記で図３を参照して説明した構造から、ｄｅｖ＿ｅｎｕｍｅｒａｔｅが接続されたクラスタ・サイトを表す追加の「／ｈｏｓｔｉｄ」ノード・レベルを「／ｄｅｖｉｃｅｓ」ノードのすぐ下に含めるように変更される。各ノード２０２が、そのノード上の装置を表すそれ自体のＤｅｖＩｎｆｏツリー３０２を有することに留意されたい。しかし、物理名空間で表されるように、ＤｅｖＩｎｆｏツリーの集合は、この追加の／ｈｏｓｔｉｄノードを含む単一の表現として組み合わされる。（たとえば、典型的な物理名は文字列／ｄｅｖｉｃｅｓ／ｈｏｓｔｉｄ／．．．から始まる）。各装

置はまた、リーフ・レベルで、ローカル・マイナー番号３８２ではなくグローバル・マイナー番号３８８に関連付けられる。（ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置の）必要に応じて、ＤｅｖＩｎｆｏツリー３０２の各リーフ・ノードのｄｅｖ＿ｔ値は、対応する装置のグローバル・マイナー番号３８８およびそのドライバのメジャー番号３９０から導かれる。たとえば、グローバル・マイナー番号ＧＮと、マイナー名ＭＮと、ドライバｓｄ@ａｄｄｒｙとを有するノード２０２－ｘ上のＳＣＳＩディスクへの物理パスは、本発明では以下のように表される。／ｄｅｖｉｃｅｓ／ｎｏｄｅ＿２０２－ｘ／ｉｏｍｍｕ@ａｄｄｒ／ｓｂｕｓ@ａｄｄｒ／ｅｓｐ@ａｄｄｒ／ｓｄ@ａｄｄｒｙ：ＭＮ

【００６１】

この物理名は、メジャー番号およびグローバル・マイナー番号から導かれたｄｅｖ＿ｔ値を属性として含む、所与の装置に関する構成情報を含むＵＦＳファイル１７０（図２Ｂ）の物理名に対応する。

【００６２】

本発明のリンク・ジェネレータ２７４は、ＤｅｖＩｎｆｏパスの少なくとも一部と、ＤＣＳから返されたグローバル・マイナー番号に従って修正されたドライバから与えられたマイナー名とから装置（および対応するＵＦＳ）の論理名を導く。

【００６３】

たとえば、ノード２０２－１が、最初にドライバ・マイナー名ａないしｄおよびドライバ・マイナー番号０ないし３で割り当てられた４つのスライスを含む１枚のＳＣＳＩディスクを有し、ノード２０２－２が、マイナー名ａないしｆおよびマイナー番号０ないし５で割り当てられた６つのスライスを含む１枚のＳＣＳＩディスクを有すると仮定する。これらの装置を接続したときに、ＤＣＳ３６０が第１のＳＣＳＩディスクに関してはグローバル・マイナー番号０ないし３を返し、第２のＳＣＳＩディスクに関してはグローバル・マイナー番号４ないし９を返したと仮定する。ＤＤＩ２７０は、これらのグローバル・マイナー番号を使用して物理名（後述）を作成し、リンク・ジェネレータ２７４はＤＤＩ２７０を使用して、以下のように物理名にマップされる論理名を作成する。

ドライバ２８０からのマイナー名	リンク・ジェネレータ２７４からの論理名
ａ（ノード２０２－１）	／ｄｅｖ／ｄｓｋ／ｃ０ｔ０ｄ０ｓ０
ｂ　〃	／ｄｅｖ／ｄｓｋ／ｃ０ｔ０ｄ０ｓ１
ｃ　〃	／ｄｅｖ／ｄｓｋ／ｃ０ｔ０ｄ０ｓ２
ｄ　〃	／ｄｅｖ／ｄｓｋ／ｃ０ｔ０ｄ０ｓ３
ａ（ノード２０２－２）	／ｄｅｖ／ｄｓｋ／ｃ１ｔ０ｄ０ｓ０
ｂ　〃	／ｄｅｖ／ｄｓｋ／ｃ１ｔ０ｄ０ｓ１
・・・	
ｆ　〃	／ｄｅｖ／ｄｓｋ／ｃ１ｔ０ｄ０ｓ５

【００６４】

ノード２０２－１および２０２－２に割り当てられた論理名はそれぞれの異なるクラスタ値（論理名文字列ｃｘｔ０ｄ０ｓｙのｃｘ部分。この場合、「ｘ」および「ｙ」は変数である）を有する。これは、論理名が装置の物理名にマップされ、クラスタ内で、それぞれの異なるノード上の装置がそれぞれの異なるコントローラに関連付けられるからである。たとえば、ノード２０２－１コントローラはｃ０として表され、ノード２０２－２はｃ１として表される。

【００６５】

ＤＤＩ２７０は、同じｇｍｉｎ情報を使用して物理名空間３０５を生成し、対応する装置のｄｅｖ＿ｔ値が属性として含まれるファイルを識別する論理名と物理名との間のマップを生成する。上記の例では、論理名空間３０４と、論理名空間と物理名空間とのマップは以下のように更新される（ａｄｄｒが任意のアドレスを代替することに留意されたい）。

論理名	ＤｅｖＩｎｆｏツリー３０２からの物理名
/dev/dsk/c0t0d0s0	/devices/node_202-1/iommu@addr/sbus@addr/espl@addr/sd@0:a
/dev/dsk/c0t0d0s1	〃 /espl@addr/sd@0:b
/dev/dsk/c0t0d0s2	〃 /espl@addr/sd@0:c
/dev/dsk/c0t0d0s3	〃 /espl@addr/sd@0:d
/dev/dsk/c1t0d0s0	/devices/node_202-2/iommu@addr/sbus@addr/espl@addr/sd@0:minor
/dev/dsk/c1t0d0s1	〃 /espl@addr/sd@0:e
/dev/dsk/c1t0d0s2	〃 /espl@addr/sd@0:f
...	
/dev/dsk/c0t0d0s5	〃 /espl@addr/sd@0:i

【００６６】

この例は、ＤＤＩ２７０が、ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置、たとえばＳＣＳＩ装置がそのクラスのメンバである論理名および物理名を生成することを示している。簡単に言えば、ｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置を命名する規則では、特定のドライバ（たとえば、ｓｄ）によって列挙される各インスタンスが固有のグローバル・マイナー番号を有し、そのグローバル・マイナー番号をドライバのメジャー番号と組み合わせたときに、対応する固有のｄｅｖ＿ｔ値が形成される必要がある。このような規則は、各インスタンスに関連付けられた物理名が、そのインスタンスのホストｉｄとグローバル・マイナー番号を、他の従来の物理パス情報と共に含まなければならないことも指定する。他のクラスの他の装置を命名する規則は、上記でｄｅｖ＿ｅｎｕｍｅｒａｔｅクラスに関して説明した規則に類似している。

【００６７】

特に、ＤＤＩ２７０は、ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置に形式／ｄｅｖ／ｄｅｖｉｃｅ＿ｎａｍｅの論理名および以下の形式の物理名を割り当てる。／ｄｅｖｉｃｅｓ／ｐｓｅｕｄｏ／ｄｒｉｖｅｒ@ｇｍｉｎ：ｄｅｖｉｃｅ＿ｎａｍｅ

この場合、ｄｅｖｉｃｅ＿ｎａｍｅは名前３８４であり、ｐｓｅｕｄｏは、この種の装置が擬似装置であることを示し、ｄｒｉｖｅｒは対応するドライバのｉｄであり、@ｇｍｉｎ：ｄｅｖｉｃｅ＿ｎａｍｅは、ｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃ装置のグローバル番号３８８および装置名３８４を示す。たとえば、カーネル・メモリ装置の論理名および物理名はそれぞれ、／ｄｅｖ／ｋｍｅｍおよびｄｅｖｉｃｅｓ／ｐｓｅｕｄｏ／ｍｍ@１２：ｋｍｅｍであってよい。前述のように、ｋｍｅｍ装置には、それに特定のノード上でアクセスできるようにする論理名を与えることができる。たとえば、ＤＤＩ２７０は、論理名／ｄｅｖ／ｋｍｅｍ０を物理名／ｄｅｖｉｃｅｓ／ｈｏｓｔｉｄ０／ｐｓｅｕｄｏ／ｍｍ@０：ｋｍｅｍにマップする。

【００６８】

ｄｅｖ＿ｇｌｏｂａｌクラスの場合、ＤＤＩによって生成される各論理名は、ファイル・システムによってクラスタ２００内の任意の装置に関連付けられる共通の物理パスを識別する。このような装置の論理名は、／ｄｅｖ／ｄｅｖｉｃｅ＿ｎａｍｅの形式であり、以下の形式の物理名にマップされる。

／ｄｅｖｉｃｅｓ／ｐｓｅｕｄｏ／ｃｌｏｎｅ@ｇｍｉｎ：ｄｅｖｉｃｅ＿ｎａｍｅ

この場合、ｄｅｖｉｃｅ＿ｎａｍｅはドライバに特有の名前３８４であり、ｐｓｅｕｄｏは、この種の装置が擬似装置であることを示し、ｃｌｏｎｅはこの装置がクローン可能で

あることを示し、@ｇｍｉｎ：ｄｅｖｉｃｅ＿ｎａｍｅは、ｄｅｖ＿ｇｌｏｂａｌ装置のグローバル番号３８８および装置名３８４を示す。たとえば、表１のｔｃｐ装置の論理名は／ｄｅｖ／ｔｃｐであり、物理名は／ｄｅｖｉｃｅｓ／ｐｓｅｕｄｏ／ｃｌｏｎｅ@０：ｔｃｐである。前述のように、本発明では、ｋｍｅｍ装置の場合とは異なり、ｄｅｖ＿ｇｌｏｂａｌ装置どうしを区別することができないことに留意されたい。すなわち、すべてのｄｅｖ＿ｇｌｏｂａｌ装置が区別不能である。

【００６９】

本発明のクラス・ベースの命名システムの利点は、それが、Ｓｏｌａｒｉｓの従来のバージョン向けに設計された従来型のソフトウェアとの互換性を有することである。たとえば、従来型のプログラムはｏｐｅｎ（／ｄｅｖ／ｋｍｅｍ）要求を発行することができ、この場合、本発明を具体化するＳｏｌａｒｉｓのバージョンは、ローカルｋｍｅｍ装置のハンドルを返す。本発明は、ｄｅｖ＿ｇｌｏｂａｌ装置とｄｅｖ＿ｅｎｕｍｅｒａｔｅ装置に対して同様な結果を与える。従来技術には、ｄｅｖ＿ｎｏｄｅｂｏｕｎｄ装置に関する概念はなかった。

【００７０】

それぞれの異なるクラスの装置がクラスタ２００のそれぞれの異なるノード上でアクセスできる一貫したグローバル名空間を、ＤＤＩ２７０およびＤＣＳ３６０がどのように形成するかについて説明したが、次に、他のノード上の装置に対するオープン要求に応答するために本発明によって使用されるステップについて図８Ａおよび８Ｂを参照して説明する。

【００７１】

図８Ａおよび８Ｂを参照すると、ノード２０２－３上に存在する装置１０６－２（図８Ｂ）にアクセス（オープン）するためにノード２０２－１上で実行されているアプリケーション１５０からの要求（８－１）に応答して本発明によって実行されるステップの流れ図が示されている。この例では、ファイル・システム２０６およびＤＣＳ３６０はそれぞれ、ノード２０２－２および２０４上に存在する。アプリケーション１５０は装置の論理名上のカーネル２４２にオープン要求を発行する。カーネル２４２は次いで、ファイル・システム２０６に問い合わせ装置のｄｅｖ＿ｔ値を判定する。ファイル・システムがカーネル２４２とは異なるノード上にあるので、これは、プロキシ・ファイル・システムＰｘＦＳを使用するマルチステップ・プロセスであり、プロキシ・ファイル・システムＰｘＦＳの大部分の態様はすでにＳｏｌａｒｉｓの現行のバージョンによって定義されている。しかし、本発明は、Ｓｏｌａｒｉｓの従来のバージョンとはまったく異なり、ＰｘＦＳクライアント２４６やＰｘＦＳサーバ２４８などのプロキシ・ファイル・システム要素をＤＣＳ３６０との対話をサポートするように修正する。ＰｘＦＳクライアント２４６とＰｘＦＳサーバ２４８とファイル・システム２０６との間の対話について簡単に説明する。

【００７２】

まずファイル・システム２０６にアクセスする必要のあるカーネル２４２などのオブジェクトは、そのローカルＰｘＦＳクライアント２４６にアクセス要求を発行する。ＰｘＦＳクライアントは、ファイル・システム２０６と共に配置されたＰｘＦＳサーバ２４８の参照を保持する。この参照によって、ＰｘＦＳクライアント２４６はカーネルの要求をＰｘＦＳサーバ２４８を介してファイル・システム２０６に伝達することができる。ファイル・システム２０６は、要求されたアクセスを実行し、要求されたファイルを表すｖノード・オブジェクト２５２を作成し、ｖノード・オブジェクト２５２の参照をＰｘＦＳサーバ２４８に返す。ノード２０２－１および２０２－２がそれぞれの異なるアドレス空間であるので、ｖノード２５２の参照は、ノード２０２－１内のＰｘＦＳクライアント２４６およびカーネル２４２には無用である。したがって、ＰｘＦＳサーバ２４８は、ｖノード２５２にリンクされたファイル転送オブジェクト（ｆ＿ｏｂｊ）２５０を作成し、ｆ＿ｏｂｊ２５０の参照をＰｘＦＳクライアント２４６に返す。ＰｘＦＳクライアント２４６は、ｆ＿ｏｂｊ参照を受信した後、ｆ＿ｏｂｊ２５０にリンクされたプロキシｖノード（ｐｘ＿ｖｎｏｄｅ）２５６を作成する。カーネル２４２は次いで、単にローカルｐｘ＿ｖノー

ド２５６にアクセスすることによって、ｖノード２５２で表されたファイル情報にアクセスすることができる。

【００７３】

カーネル２４２は、この機構を使用して、ＰｘＦＳクライアント２４６に対してオープンされる装置の論理名に関する参照メッセージ（８－２）を発行し、ＰｘＦＳクライアント２４６は同様な参照メッセージ（８－３）をＰｘＦＳサーバ２４８へ送る。ＰｘＦＳサーバ２４８は、ファイル・システム２０６にｌｏｏｋｕｐ（ｌｏｇｉｃａｌ＿ｎａｍｅ），ｇｅｔ＿ｖｎｏｄｅメッセージ（８－４）を発行し、論理シンボリック・リンクを介してｌｏｇｉｃａｌ＿ｎａｍｅを対応するｐｈｙｓｉｃａｌ＿ｎａｍｅにマップし、そのｐｈｙｓｉｃａｌ＿ｎａｍｅで識別されるＵＦＳファイルを表すｖ＿ｎｏｄｅ２５２の参照を返すことを要求する。ｐｈｙｓｉｃａｌ＿ｎａｍｅがこの例と同様に装置を参照するものであるとき、装置の属性は装置の固有のｄｅｖ＿ｔを含む。前述のように、ファイル・システム２０６は次いで、ＰｘＦＳサーバ２４８にｖノードを返し（８－５）、ＰｘＦＳサーバ２４８が対応するｆ＿ｏｂｊ２５０を作成し、ｆ＿ｏｂｊ２５０の参照をＰｘＦＳクライアント２４６に返す（８－６）。ＰｘＦＳクライアント２４６は次いで、要求された装置に関するｄｅｖ＿ｔ情報が属性に含まれるｐｘ＿ｖｎｏｄｅ２５６を作成し、そのｐｘ＿ｖｎｏｄｅ２５６の参照をカーネル２４２に渡す（８－７）。この点で、カーネル２４２はｐｘ＿ｖｎｏｄｅ２５６のＰｘＦＳクライアント２４６にオープン・メッセージ（８－８）を返す。ＰｘＦＳクライアント２４６は、このメッセージを受信した後、ｄｅｖ＿ｔ値を含むｐｘ＿ｖｎｏｄｅの属性から、対応するｖノード２５２が装置を表し、従って、ＤＣＳ３６０によってオープン・メッセージを処理しなければならないと判定する。ｐｘ＿ｖｎｏｄｅ２５６がｄｅｖ＿ｔ値を含まない場合、ＰｘＦＳクライアント２４６は他のチャネルを通してオープン要求（８－８）を満たす。Ｓｏｌａｒｉｓの従来のバージョンの場合と同様に、装置はローカルにしかアクセスできないので、ＰｘＦＳクライアントはｄｅｖ＿ｔ値に関する試験を実行しない。

【００７４】

ｐｘ＿ｖｎｏｄｅ２５６がｄｅｖ＿ｔ値４３０を含むので、ＰｘＦＳクライアント２４６は、ｄｅｖ＿ｔに対応する装置のＤＣＳ３６０にｒｅｓｏｌｖｅメッセージ（８－９）を発行する。次に、ＤＣＳ３６０がこの要求をどう扱うかについて図８Ｂを参照して説明する。

【００７５】

図８Ｂを参照するとわかるように、ｒｅｓｏｌｖｅ（ｄｅｖ＿ｔ）メッセージ（８－９）に応答して、ＤＣＳ３６０はＤＣＳデータベース３７２を参照し、ｄｅｖ＿ｔ値に対応する装置の位置およびＩＤを判定する。装置クラス３１２に関する前述の議論と同様に、ｄｅｖ＿ｅｎｕｍｅｒａｔｅクラスおよびｄｅｖ＿ｎｏｄｅｂｏｕｎｄクラスの装置は、ＤＣＳデータベース３７２の数値フィールド３９４にその位置が指定された特定のノード上でアクセスされる。これに対して、ｄｅｖ＿ｇｌｏｂａｌクラスまたはｄｅｖ＿ｎｏｄｅｓｐｅｃｉｆｉｃクラスの装置は、要求側アプリケーションのローカル・ノード上でアクセスされる。ＤＣＳ３６０は、オープンすべき装置の位置を判定した後、要求された装置が属する装置クラスを管理し、要求されたオブジェクトを備えるノードに対してローカルなＤＳＯ２９０の参照（ＤＳＯ＿ｒｅｆ）をＰｘＦＳクライアント２４６に返す（８－１０）。この例では、要求された装置１０６－２がｄｅｖ＿ｅｎｕｍｅｒａｔｅクラスの装置であり、ノード２０２－３上に存在すると仮定した場合、ＤＳＯ＿ｒｅｆはノード２０２－３上のＤＳＯ＿ｅｎｕｍオブジェクト２９２に返される。

【００７６】

ＰｘＦＳクライアント２４６は、メッセージ（８－１０）を受信した後、装置１０６－２に関するｇｅｔ＿ｄｅｖｉｃｅ＿ｆｏｂｊ要求を、参照されたＤＳＯ２９２（８－１１）に発行する。これに応答して、ＤＳＯ２９２はｃｒｅａｔｅ＿ｓｐｅｃｖｐ（）メッセージ（８－１２）を発行し、ノード２０２－３上のＳｐｅｃＦＳ４１０に装置１０６－２のｓノードを作成し返す（８－１３）ように要求する。ＤＳＯ２９２は次いで、ＰｘＦＳサ

ーバ２４８－２にｓノードのｆ＿ｏｂｊ参照を要求し（８－１４ａ）、ＰｘＦＳサーバ２４８－２は、要求されたｆ＿ｏｂｊを返す（８－１４ｂ）。ＤＳＯ２９２は次いで、ｓノードのｆｏｂｊ参照をＰｘＦＳクライアント２４６に返す（８－１５）。クライアント２４６は次いで、このｆｏｂｊに関するオープン要求（８－１６）を発行し、この要求がＰｘＦＳサーバ２４８－２を介してＳｐｅｃＦＳ４１０へ送られる（８－１７）。

【００７７】

ＳｐｅｃＦＳ４１０は次いで、装置１０６－２をオープンすることを試みる。オープン動作の結果に応じて、ＳｐｅｃＦＳ４１０は成功と失敗のどちらかを示す状況メッセージ（８－１８）を返す。オープンが成功した場合、状況メッセージ（８－１８）が、オープンされたｓノード４３２の参照も含む。ＰｘＦＳサーバ２４８－２は、状況メッセージ（８－１８）内の「成功」を受信した後、オープンされたｖ＿ｎｏｄｅ２５２－２に関するｆ＿ｏｂｊ２５０－２を作成し、ＰｘＦＳクライアント２４６に返す（８－１９）。ＰｘＦＳクライアント２４６は、ノードを介してｆ＿ｏｂｊ２５０－２にリンクされたｐｘ＿ｖｎｏｄｅ２５６－２を作成する。装置オープン動作の最後のステップとして、ＰｘＦＳクライアントはｐｘ＿ｖｎｏｄｅ２５６－２をカーネル２４２に返し（８－２０）、カーネル２４２が、対応するユーザ空間ファイル記述子（ｆｄ）４３４を作成する。カーネル２４２はこのファイル記述子をアプリケーション１５０－１に返し（８－２１）、次いでアプリケーション１５０－１は、ファイル記述子４３４を使用して装置１０６－２と直接（すなわち、カーネル２４２、ＰｘＦＳクライアント２４６、ｐｘ＿ｎｏｄｅを介して）対話することができる。

【００７８】

いくつかの特定の実施形態を参照して本発明について説明したが、この説明は本発明を例示するものであり、本発明を制限するものと解釈すべきではない。当業者には、添付の特許請求の範囲で定義される本発明の真の趣旨および範囲から逸脱せずに様々な修正が構想されよう。

【図面の簡単な説明】

【図１】単一のコンピュータ上の装置へのアクセスを可能にするために使用される構成要素を示す従来技術のコンピュータ・システムのブロック図である。

【図２】従来技術におけるアプリケーションとオペレーティング・システム・カーネルとファイル・システムと装置との間の関係を示すブロック図（Ａ）と従来技術における装置論理名と物理名とファイル・システムと装置タイプ識別子（ｄｅｖ＿ｔ）と装置との間の関係を示すブロック図（Ｂ）である。

【図３】従来技術で使用される装置情報ツリー（ＤｅｖＩｎｆｏ　Ｔｒｅｅ）に整合する例示的な装置情報ツリーの図である。

【図４】アプリケーション１５０の要求に応じて装置をオープンする際に従来技術のコンピュータ・システム１００のメモリ１０４で実行される動作の流れ図である。

【図５】本発明を実施することのできるコンピュータ・クラスタのブロック図である。

【図６】図５のクラスタの代表的なノード２０２および２０４で実施された本発明を構成するメモリ・プログラムおよびデータ構造のブロック図である。

【図７Ａ】装置ドライバ・インタフェース（ＤＤＩ）フレームワークおよび装置構成システム（ＤＣＳ）が、ノード２０２に接続中の装置の適切なｄｅｖ＿ｔ値、論理名、物理名を規定する動作を示す流れ図である。

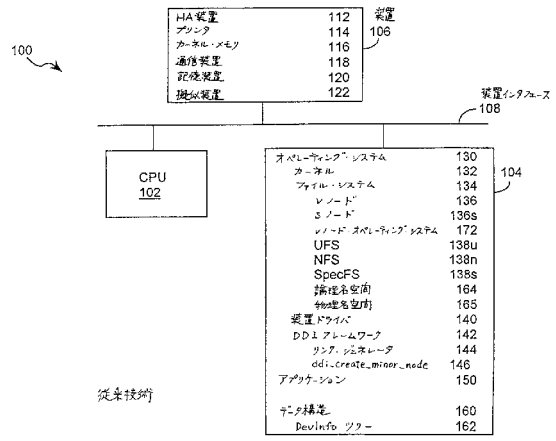
【図７Ｂ】本発明によって規定されたローカル・マイナー名／番号と物理名と論理名との間の関係を示す図である。

【図８Ａ】ノード２０２－３上に存在する装置にアクセス（オープン）するためにノード２０２－１上で実行されているアプリケーション１５０からの要求に応答して本発明によって実行される各ステップを示す流れ図である。

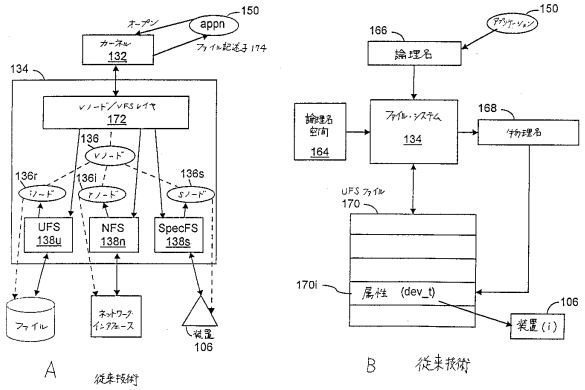
【図８Ｂ】ノード２０２－３上に存在する装置にアクセス（オープン）するためにノード２０２－１上で実行されているアプリケーション１５０からの要求に応答して本発明によって実行される各ステップを示す流れ図である。

【符号の説明】
１５０　アプリケーション
２００　クラスタ
２０１　クラスタ
２０２　ノード
２０４　ノード
２０６　グローバル・ファイル・システム
２０８　装置構成システム
２４０　オペレーティング・システム（ＯＳ）ルーチン／オブジェクト
２４２　カーネル
２４４　プロキシ・ファイル・システム（ＰｘＦＳ）
２５０　ファイル・オブジェクト
２５２　仮想ファイル・ノード
２５４　特殊ファイル・ノード
２５６　プロキシｖノード
２５８　特殊ファイル・システム
２７０　装置ドライバ・フレームワーク（ＤＤＩ）
２７２　接続メソッド
２７４　リンク・ジェネレータ
２８０　装置ドライバ

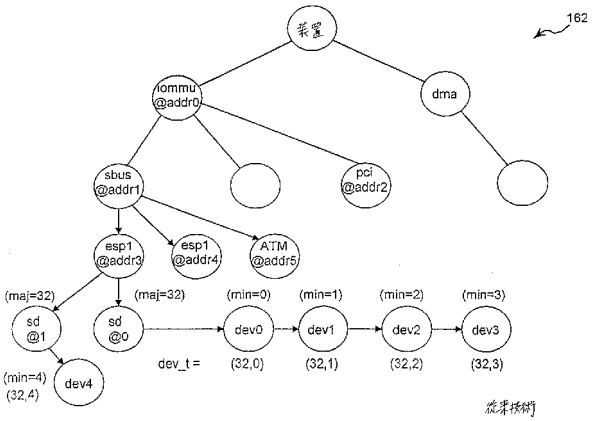
【図１】



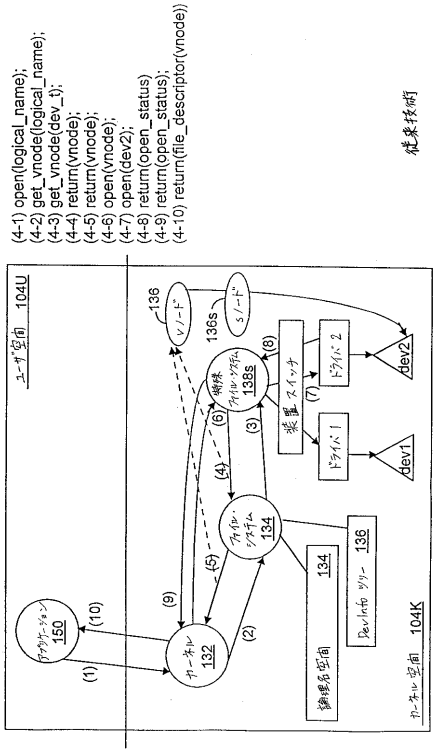
【図２】



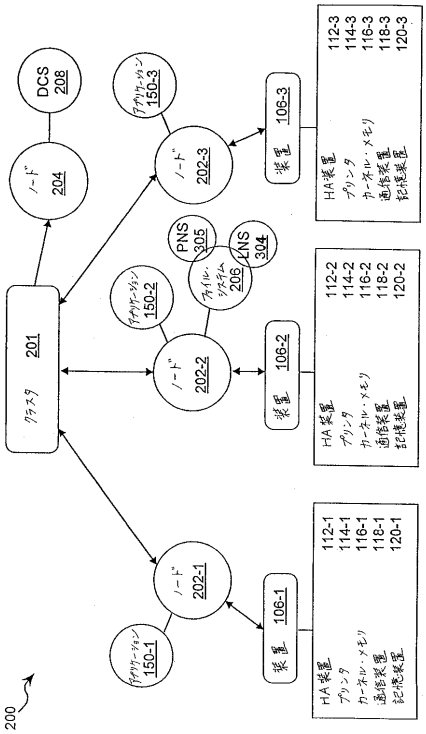
【図３】



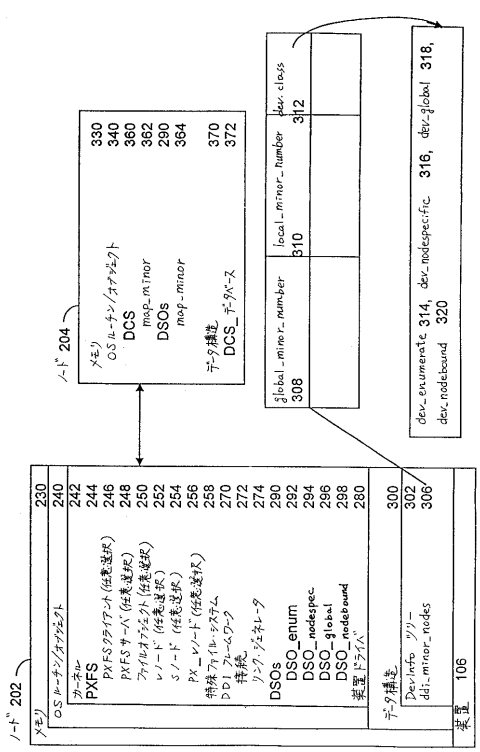
【図４】



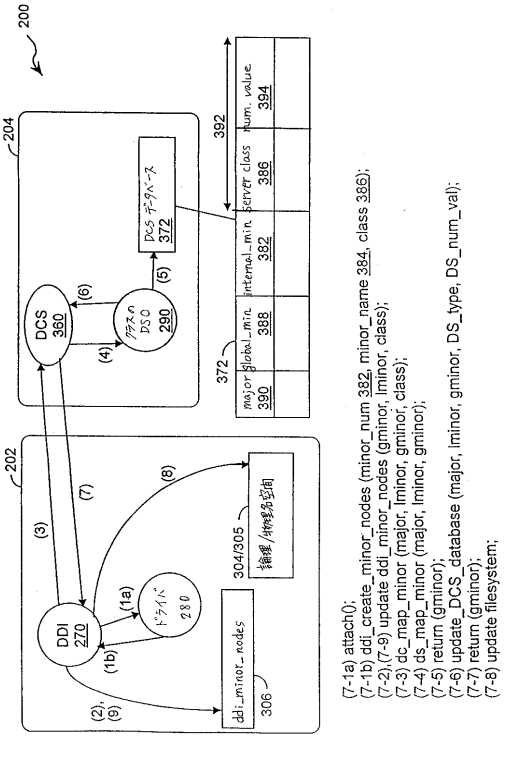
【図５】



【図６】

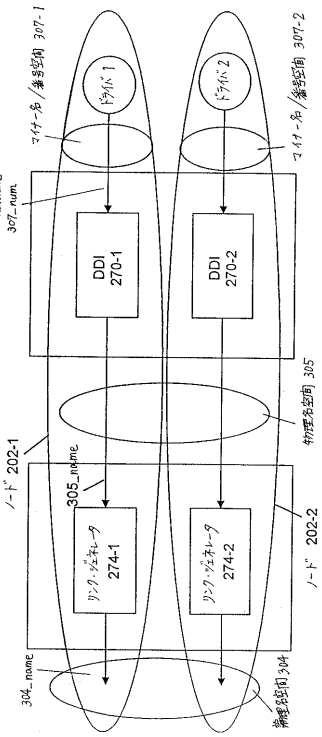


【図７Ａ】

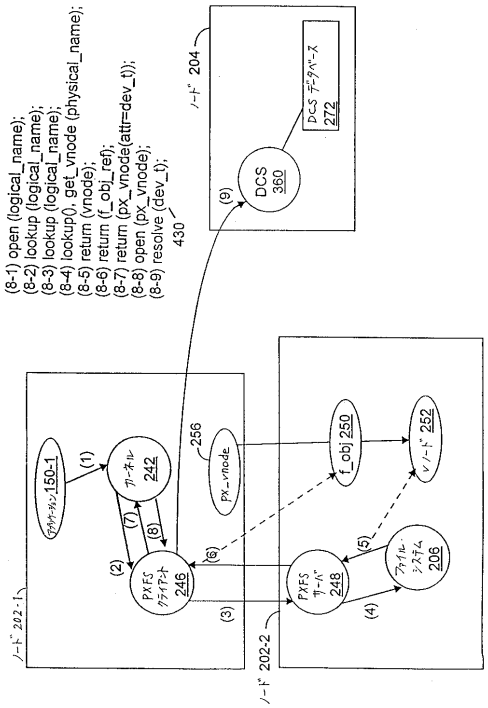


(7-1a) attach();
(7-1b) ddi_create_minor_nodes (minor_num 382, minor_name 384, class 386);
(7-2),(7-9) update ddi_minor_nodes (gminor, iminor, class);
(7-3) dc_map_minor (major, Iminor, gminor, class);
(7-4) ds_map_minor (major, Iminor, gminor);
(7-5) return (gminor);
(7-6) update_DCS_database (major, Iminor, gminor, DS_type, DS_num_val);
(7-7) return (gminor);
(7-8) update filesystem;

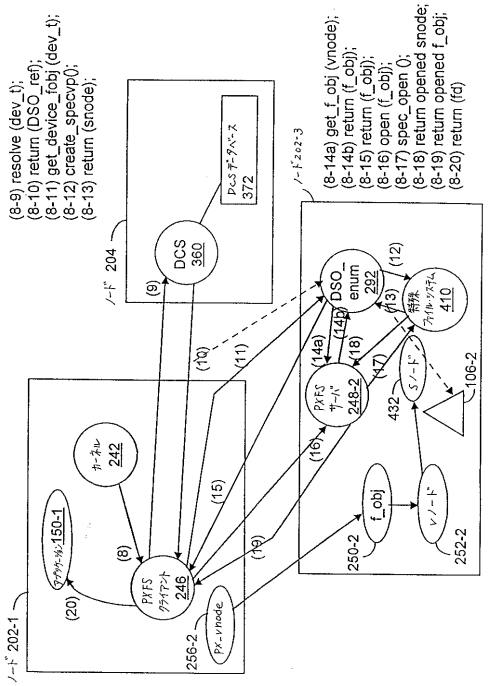
【図７Ｂ】


DDI 270-1
DDI 270-2
ノード 202-1
ノード 202-2
305_name
307_name
307_num
304_name
物理名空間 305
論理空間 304

【図８Ａ】


(8-1) open (logical_name);
(8-2) lookup (logical_name);
(8-3) lookup (logical_name);
(8-4) lookup(), get_vnode (physical_name);
(8-5) return (vnode);
(8-6) return (f_obj_ref);
(8-7) return (px_vnode(attr=dev_t));
(8-8) open (px_vnode);
(8-9) resolve (dev_t);
430
DCS 360
DCS データベース 272
カーネル 242
PXFS クライアント 246
PXFS サーバ 248
f_obj 250
px_vnode 256
vノード 252
ファイルシステム 206

【図８Ｂ】


(8-9) resolve (dev_t);
(8-10) return (DSO_ref);
(8-11) get_device_fobj (dev_t);
(8-12) create_specvp();
(8-13) return (snode);
(8-14a) get_f_obj (vnode);
(8-14b) return (f_obj);
(8-15) return (f_obj);
(8-16) open (f_obj);
(8-17) spec_open ();
(8-18) return opened snode;
(8-19) return opened f_obj;
(8-20) return (fd)
DCS 360
DCS データベース 372
カーネル 242
PXFS クライアント 246
PXFS サーバ 248-2
DSO_enum 282
ファイルシステム 410
f_obj 250-2
vノード 252-2
px_vnode 256-2
106-2
432

フロントページの続き

(72)発明者 シアマク・ナザリ
アメリカ合衆国・９１００６・カリフォルニア州・アルカディア・オークウッド アヴェニュ・１８６２
(72)発明者 アニル・スワループ
アメリカ合衆国・９２３５４・カリフォルニア州・ロマ リンダ・シエロ レーン・１１５６２
(72)発明者 ヨーセフ・カリディ
アメリカ合衆国・９４０８６・カリフォルニア州・サニーベイル・ウエスト ガーランド テラス・６３３

審査官 石井 茂和

(56)参考文献 米国特許第０５５５５４０１（ＵＳ，Ａ）
Brent Welch, A Comparison of Three Distrebuted File System Architectures: Vnode, Sprite, and Plan9, COMPUTING SYSTEMS, USENIX ASSOCIATION, 米国, the MIT Press, １９９４年 ７月 ８日, VOL.7 No.2, p175-199
Greg Thiel , LOCUS operating system, a transparent system, COMPUTER COMMUNICATIONS, NL, ELSEVIER SCIENCE PUBLISHERS, BV., １９９１年 ７月 １日, vol.14, no.6, p336-346
Jim-Min Lin & Shang Rong Tsai, Supporting user mobility in the Distributed Logical Machine System, MICROPROCESSING and MICROPROGRAMMING, NL, ELSEVIER SCIENCE PUBLISHERS, BV., １９９５年 ８月 １日, vol.41, no.4, p315-330

(58)調査した分野(Int.Cl., ＤＢ名)
G06F 13/14
WPI(DIALOG)
JSTPlus(JDream2)