(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3604176号
(P3604176)
(45)発行日　平成16年12月22日(2004.12.22)　(24)登録日　平成16年10月8日(2004.10.8)

(51)Int.Cl.[7]　F I
G06F 9/46　G06F 9/46　360C
G06F 12/08　G06F 12/08　525A

請求項の数 5　(全 18 頁)

(21)出願番号　特願平6-219655
(22)出願日　平成6年9月14日(1994.9.14)
(65)公開番号　特開平8-83253
(43)公開日　平成8年3月26日(1996.3.26)
審査請求日　平成12年9月13日(2000.9.13)

(73)特許権者　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(74)代理人　100083161
弁理士　外川　英明
(72)発明者　岡本　利夫
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝　研究開発センター内
(72)発明者　木村　哲郎
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝　研究開発センター内
(72)発明者　白川　健治
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝　研究開発センター内

最終頁に続く

(54)【発明の名称】仮想空間管理方法及び分散処理システム

(57)【特許請求の範囲】

【請求項１】

複数の計算機と、前記複数の計算機で共有される単一の仮想空間を管理する管理計算機と、前記複数の計算機と前記管理計算機とのそれぞれの間で相互に通信を行うための通信手段とから成るシステムにおける仮想空間管理方法であって、

前記管理計算機は、前記仮想空間から記憶領域を各計算機に割り当てる場合に、この割り当てる記憶領域が既に割り当て済の記憶領域と重複していないことを保証する重複回避機構を備えており、

前記複数の計算機のそれぞれは、記憶領域の確保が必要になったとき、必要量より大きい量の記憶領域を、前記管理計算機の前記重複回避機構を介して確保し、

前記複数の計算機のそれぞれは、確保した前記記憶領域を、他の計算機と通信することなく自計算機内で管理するようにし、

前記複数の計算機のそれぞれは、自計算機以外の計算機が管理する記憶領域を使用したい場合は、この前記記憶領域を管理する計算機と前記通信手段により通信を行い、使用の許可を得るようにしたことを特徴とする仮想空間管理方法。

【請求項２】

複数の計算機と、前記複数の計算機で共有される単一の仮想空間を管理する管理計算機と、前記複数の計算機と前記管理計算機とのそれぞれの間で相互に通信を行うための通信手段とを備える分散処理システムであって、

前記管理計算機は、

前記仮想空間から記憶領域を各計算機に割り当てる場合に、この割り当てる記憶領域が既に割り当て済の記憶領域と重複していないことを保証する重複回避機構を備え、
前記複数の計算機のそれぞれは、
前記通信手段を介して前記管理計算機へ記憶領域の確保の要求を行い、必要量より大きい量の記憶領域を前記重複回避機構を介して確保する確保手段と、
前記確保手段で確保した前記記憶領域を、他の計算機と通信することなく自計算機内で管理する管理手段と、
自計算機以外の計算機が管理する記憶領域を使用したい場合に、前記使用したい記憶領域を管理する計算機と前記通信手段により通信を行い、使用の許可を得るよう取得手段とを備えたことを特徴とする分散処理システム。

【請求項３】
前記複数の計算機のそれぞれは、割り当てられた前記記憶領域をさらに複数のサブ領域に分け、このサブ領域ごとに保護属性を設定して管理することを特徴とする請求項１記載の仮想空間管理方法。

【請求項４】
各計算機はそれぞれディスク装置を備えており、自計算機に割り当てられた前記記憶領域に配置したプログラムあるいはデータを、自計算機の備えるディスク装置に対して退避あるいは復旧することを特徴とする請求項１記載の仮想空間管理方法。

【請求項５】
前記重複回避機構は、既に割り当て済の記憶領域と割り当てた計算機を示す情報とを対応付けた管理テーブルを備えるようにしたことを特徴とする請求項１記載の仮想空間管理方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】
本発明は、仮想記憶空間管理機構を有する計算機システムが、相互にネットワーク等を介して接続され、分散システムを構築している場合の仮想空間を管理する方式に関する。

【０００２】

【従来の技術】
近年、計算機性能の向上とネットワーク技術の進歩により、複数の計算機上でプログラムを動作させ、通信しながら協調して処理を進めていく分散処理が一般化してきている。

【０００３】
従来からの計算機を用いての処理形態では、１台の計算機で、単独にすべての処理を行っていた。したがって計算機資源を管理し、アプリケーションプログラムにサービスを提供する基本プログラムであるオペレーティングシステム（ＯＳ）は、単独システム用のものが古くから開発され利用されてきた。たとえばＡＴ＆Ｔ社が開発したＵＮＩＸやＩＢＭ社のＭＶＳ　などのＯＳがその代表例である。

【０００４】
ここで、新しい分散処理の形態に拡張するために、従来からのＯＳを拡張して対応させることは、比較的容易である。つまり、従来からの機能を維持しつつ、新しい分散機能を追加提供していく。しかし、拡張には自ずと限度があり、将来の新しい機能の要望に対して柔軟に対応できない可能性がある。また、ＯＳの上ですべてのアプリケーションプログラムが動作するので、ＯＳ自身の実行効率、信頼性などの点が、拡張を繰り返し、ＯＳの内部が複雑になるにつれて問題になってくる。

【０００５】
従来の単独システム用のＯＳでは、アプリケーションプログラムの実行する場であるアドレス空間は、計算機ごとに管理されてきた。したがって、従来のＯＳを分散システムに拡張するには、専用のシステムコールを設けるのが常であった。たとえば、Ｕｎｉｘ系のＯＳである４．３ＢＳＤでは、ソケットインタフェースという新しいシステムコールを設けて、従来のアドレス空間上でアプリケーションプログラムを動かす。従来の（分散を使わ

ない）アプリケーションを変更無しに動作できるが、アプリケーションプログラマは、分散対応のアプリケーションを設計する際に、そのシステム構成を理解しなくてはならない欠点があった。また、ＯＳの種類によって、分散化への拡張方法が異なるので、アプリケーションプログラムを他のＯＳ上で動作するように変更しなければならないという欠点もあった。

【０００６】

以上まとめると、従来のＯＳを分散環境に対応するように拡張すると、分散対応のアプリケーションを実行する仮想空間の対応がばらばらになり、プログラムの再利用、共有データの利用などが非常に煩雑になる欠点があった。

【０００７】

【発明が解決しようとする課題】

本発明においては、上記の欠点を解決するために、単一仮想空間、つまり、すべてのアプリケーションを実行する仮想空間を同一にしたＯＳに関し、単一仮想空間の分散環境への拡張方法を単一仮想空間をネットワークワイドに広げることで、ネットワーク透過な仮想空間をＯＳが提供し、アプリケーションレベルに分散化を意識させないようにする。特に、本発明は、上記のようなネットワーク透過な単一仮想空間を実現する際に問題となる仮想空間の管理法について解決するものである。

【０００８】

即ち、本発明は、実際には単体で処理を行う計算機が複数で共通して１つの仮想空間を利用しても、相互の計算機がどのようにこの空間を利用しているか等の管理を簡単にでき、ＯＳやアプリケーションを効率よく動作させることのできる仮想空間管理装置及び方法の提供を目的とする。

【０００９】

【課題を解決するための手段】

本発明は、複数の計算機と、前記複数の計算機で共有される単一の仮想空間を管理する管理計算機と、前記複数の計算機と前記管理計算機とのそれぞれの間で相互に通信を行うための通信手段とから成るシステムにおける仮想空間管理方法であって、前記管理計算機は、前記仮想空間から記憶領域を各計算機に割り当てる場合に、この割り当てる記憶領域が既に割り当て済の記憶領域と重複していないことを保証する重複回避機構を備えており、前記複数の計算機のそれぞれは、記憶領域の確保が必要になったとき、必要量より大きい量の記憶領域を、前記管理計算機の前記重複回避機構を介して確保し、前記複数の計算機のそれぞれは、確保した前記記憶領域を、他の計算機と通信することなく自計算機内で管理するようにし、前記複数の計算機のそれぞれは、自計算機以外の計算機が管理する記憶領域を使用したい場合は、この前記記憶領域を管理する計算機と前記通信手段により通信を行い、使用の許可を得るようにした。

【００１０】

また、本発明は、複数の計算機と、前記複数の計算機で共有される単一の仮想空間を管理する管理計算機と、前記複数の計算機と前記管理計算機とのそれぞれの間で相互に通信を行うための通信手段とを備える分散処理システムであって、前記管理計算機は、前記仮想空間から記憶領域を各計算機に割り当てる場合に、この割り当てる記憶領域が既に割り当て済の記憶領域と重複していないことを保証する重複回避機構を備え、前記複数の計算機のそれぞれは、前記通信手段を介して前記管理計算機へ記憶領域の確保の要求を行い、必要量より大きい量の記憶領域を前記重複回避機構を介して確保する確保手段と、前記確保手段で確保した前記記憶領域を、他の計算機と通信することなく自計算機内で管理する管理手段と、自計算機以外の計算機が管理する記憶領域を使用したい場合に、前記使用したい記憶領域を管理する計算機と前記通信手段により通信を行い、使用の許可を得るよう取得手段とを備えた。

【００１４】

【作用】

本発明では、各計算機が処理のために必要とする仮想空間の一部を獲得する（あるいは解

放する）頻度は高いが、処理のために利用する部分は単一仮想空間の全てはなくごく一部である（仮想空間のローカリティ）という事実を利用して、仮想空間を管理する。つまり、仮想空間を比較的大きな単位で分割し、その分割された領域ごとに仮想空間の一部を各計算機に割り当てて管理する。

【００１５】

本発明によれば、全てのアプリケーションプログラムを一つの仮想空間に配置する単一仮想空間のＯＳを分散環境に拡張するにあたり、分散環境を構成する全ての計算機で同一の単一仮想空間を共有する分散単一仮想空間方式を取り、しかも上述した領域の割り当てによりこの分散単一仮想空間の管理が簡単化されているため、単一計算機上で動作していたプログラムを変更なしに動作させることが可能であり、アプリケーションプログラムを作成するプログラマに分散環境の複雑な仮想空間管理を意識させることなくプログラムを作成可能になる。

【００１６】

また、上述した各計算機への領域の割り当てを集中管理する仮想空間管理装置を分散システム内に設け、割り当ての管理と、各計算機ごとに独立に行う計算機資源の管理とを分離することにより、全体の動作を効率化できる。この仮想空間管理装置（実施例ではチャプターサーバと呼ぶものに相当）に対し、各計算機は新たに仮想空間上の領域を必要とする際にこれを要求し、割り当てられた領域を確保し、その領域にプログラムやデータを配置して実行することになる。

【００１７】

ここで、複数の計算機により共有された前記仮想空間の一部を各計算機に割り当てる単位である領域と、各計算機が個々に、割り当てられた領域内でアクセス保護を実現する単位であるサブ領域とを、独立にすることで、さらに全体の動作を効率化できる。

【００１８】

また、分散単一仮想空間の管理を効率よく実現するために、プログラムの実行上の性質を利用して、仮想空間の領域は割り当てられた計算機が一元的に管理するようにすると、多数の計算機が一つの仮想空間上のプログラムを実行しても管理が容易で、効率が低下しない。

【００１９】

ここで、仮想空間の領域に配置されたプログラムやデータのバッキングストレージを、その領域が割り当てられた計算機のディスク装置で行うようにすれば、領域内の管理が割り当てられた計算機内に閉じて行えるため、さらに効率化できる。

【００２０】

【実施例】

以下、図面を参照しながら本発明の一実施例について説明する。

（実施例１）

図１に本実施例に係る分散システムの全体構成を示す。高速ＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）に結合された複数のホストから分散システムが構成されている。ただし、ホストとは（１つ以上の）ＣＰＵとメモリから構成される計算機で、かつＬＡＮ上（ＯＳＩのネットワーク層）で識別のためのアドレス等のような識別子が与えられる単位である。さらに、多くの場合、ホストは、プログラム、データ、ファイルなどを記憶しておくディスク装置を持っている。ホストでの実行時は、ディスクの内容は、後述するＮＳＶＳに裏打ちされた物理メモリにロードされ実行される。

【００２１】

ホストには、分散システムを構成する為にオペレーティングシステム（ＯＳ）がのっている。このＯＳは、プログラムやデータなどを配置する仮想記憶空間の管理方法として単一仮想記憶空間管理方式を採用する。つまり、ＯＳの機能によって、ホスト上で動作するプログラムや、それを利用したりプログラムを開発したりするユーザからは、１つの仮想記憶空間しか存在していないように見せる。その仮想記憶空間を分散システム全体で共有している。

【００２２】
次に、分散システムで管理する仮想記憶空間の構成を図２に示す。本実施例では、分散システム全体で１つの巨大な仮想記憶空間を管理し、その上に本システムで扱うすべてのプログラム、データ、ファイルなどを配置する。ここで、この分散システム全体に渡って共有される単一の仮想記憶空間のことをネットワーク単一仮想記憶空間（ＮＳＶＳ）と呼ぶ。本実施例では空間を６４ビットで表現するものとする。したがって空間の大きさは（２の６４乗）ｂｙｔｅとなる。
【００２３】
この空間はＯＳが管理して作り出している仮想的なものであるので、実際には、各ホストの上での仮想記憶空間機構で実現する。つまり、各ホストは、それぞれに仮想記憶空間を別々に保持しており、その中のプログラム配置はすべてのホストから共通である。ただし、すべてのホストが、すべてのプログラムをもっているわけではなく、各ホストの保持する仮想記憶空間内にはそのホストにおける実行に必要なプログラムが配置されている。つまり、同じプログラムは同じ仮想記憶空間上の位置（　アドレス）　に存在することを各ホストのＯＳが管理している。
【００２４】
なお、上記のようにＯＳは、どのユーザに対しても同じ一つの空間しか提供しないが、実際には、ホストのハードウェア機能として複数の仮想記憶空間を管理できるのが通常である。したがって、図３に示すように、複数の仮想記憶空間を利用しながら、各仮想記憶空間の同じ位置（　アドレス）　に同じプログラムを割り当てる管理をＯＳが実現することによっても同等なことが実現可能である。
【００２５】
このように分散システム内の各ホストが協調して単一の仮想記憶空間を構築する場合、各ホストが個別に記憶領域（　仮想記憶空間の一部分）　を確保すると、領域が重複してしまう可能性があり、重複した部分の整合性がとれなくなり単一仮想記憶空間の管理を困難にする。よって、あるホストが記憶領域を確保する場合に、確保された領域が他のホストの確保している領域と重複していないことを保証する機構（重複回避機構）が必要となる。
【００２６】
本実施例では、各ホストに割り当てる記憶領域が重複しないように記憶領域の分配状況を管理する機構を担ったホスト（以下チャプターサーバと呼ぶ）を分散システム内に１台設ける。すなわち、分散システムは図１に示すように、多数のホストとチャプターサーバがＬＡＮ　で接続された構成をとる。
【００２７】
ホストが必要に応じて（　随時）　、必要量分だけの記憶領域の確保を行うと、領域の重複回避のための機構のオーバヘッドが増大し、性能に重大な影響を及ぼす。また、予め記憶領域を各ホストに分配する方式を採用すると、ホストの追加や削除といった分散システムの運用上頻繁に起こる事態に対し柔軟に対応できない。そこで、各ホストは、チャプターサーバが提供する重複回避機構を介して記憶領域の確保を行う時には、必要量より十分大きい量を一度に確保し（以下この領域をメモリチャプターと呼ぶ）、その領域から随時必要量だけ切り出し使用する。予め確保した領域を使い果たした場合、再度、重複回避機構を介してメモリチャプターの確保を行う。この方法により、重複回避のオーバヘッドの性能への影響を低減できる。
【００２８】
以上で本実施例の構成の概要を説明したので、以下に、チャプターサーバと各ホストの内部構造と動作を分けて詳しく説明する。
本実施例では仮想記憶空間の管理を効率良く行うために、メモリチャプターの大きさを固定とし（例えば（２の４８乗）ｂｙｔｅ）、仮想記憶空間アドレスの上位１６ｂｉｔ　をメモリチャプター番号とする。
［チャプターサーバ側の内部構造］

上記のチャプターサーバの構成を図４に示す。チャプターサーバは、分散システム内の他のホストと交信するための通信部４１を備える。また、チャプターサーバ内でも他のホストと同様にネットワーク単一仮想記憶空間をアプリケーションに対して提供するために、後述する仮想記憶空間管理部４２を備えても良い。

【００２９】

さらに、分散システム内の各ホストに対して、記憶領域の割当を行うＮＳＶＳ管理部４３を備える。ＮＳＶＳ管理部４３内には、使用中のメモリチャプターとそれが割り当てられたホスト（これをチャプターオーナーと呼ぶ）との対応関係を記録するＮＳＶＳ管理テーブル４４を持つ。ＮＳＶＳ管理テーブル４４の構造は、図５に示すように、メモリチャプター番号、ホスト番号、状態の組から成り、いずれかのホストに割り当てられたチャプターの状態は「使用中」となる。

ただしチャプターサーバは、各メモリチャプター内のメモリの利用状況に関する情報は保持せず、メモリチャプター内部についてはそれを確保したチャプターオーナーが個別に管理する。

［ホスト側内部構造］

分散システム内の各ホストの構成を図６に示す。各ホストは、他のホストやチャプターサーバと交信するための通信部６１を備える。また、ユーザプログラム（アプリケーション）からのサービス要求（システムコール）を受け付けるサービス要求受付部６４を備える。さらに、ネットワーク単一仮想記憶空間をアプリケーションに対して提供するために、仮想記憶空間管理部６２（４２と同じもの）を備える。

【００３０】

仮想記憶空間管理部６２内には、そのホストが確保した複数のメモリチャプターを管理するためにメモリチャプター管理リスト６３を持つ。メモリチャプター管理リスト６３は図７に示すような構造を有し、リスト構造の要素はメモリチャプター管理テーブルと呼ばれるテーブルである。メモリチャプター管理テーブルは、そのホストが確保した複数のメモリチャプターのうちの１つについて、そのメモリチャプター中のメモリの割り当て状況を保持する。メモリチャプター管理テーブルの構造を図８に示す。

【００３１】

以下、図７と図８について説明する。各メモリチャプター管理テーブルにはその属性として、そのテーブルがどのメモリチャプターを管理するものかを示すためのメモリチャプター番号と、他のメモリチャプター管理テーブルとをリストの形で結合するためのポインタが登録されている。テーブルの内部には、メモリチャプター内の利用状況を記録するために、領域（開始番地と大きさ）とその利用状況（使用中または未使用）が登録されている。

［動作］

以下、分散システム内のあるホストが記憶領域を確保する場合を例にとり、システム全体の動作を図９を用いながら順を追って説明する。

【００３２】

あるホスト上でユーザプログラムが動作し、システムコールによって、メモリ獲得要求が発行されると、まず、そのホストの仮想記憶空間管理部６２が、既に獲得しているメモリチャプター内の空き領域を、メモリチャプター管理テーブルを用いて検索する。メモリ獲得要求を満たす大きさの空き領域がある場合は、そこからメモリ領域を確保し割り当てを行い、メモリチャプター管理テーブルを更新する。要求を満たす空き領域が、ホストの保持するメモリチャプター内に存在しない場合は、通信部６１を介してチャプターサーバに対し新規メモリチャプターの獲得要求を発行する。

【００３３】

通信部４１を介して新規メモリチャプターの獲得要求を受けとった（Ｓ１及びＳ２Ｙｅｓ）チャプターサーバ内のＮＳＶＳ管理部４３は、空きメモリチャプターをＮＳＶＳ管理テーブル４４を用いて検索する（Ｓ３）。即ち、どのホストにも割り当てられていない（状態が「未使用」である）メモリチャプターを検索する。そして、これを要求を発行した

ホストに割り当てる。即ち、検索により得られた空きメモリチャプターの番号を通信部４１を介して要求元のホストへ返す（Ｓ４）。さらに、ＮＳＶＳ管理テーブル４４内の割り当てたメモリチャプターに該当する欄に、割り当て先のホスト番号を記入し状態を使用中に変更する（Ｓ５）。

【００３４】

チャプターサーバから新規メモリチャプターを受けとったホストの仮想記憶空間管理部６２は、受けとったメモリチャプターの管理のために、メモリチャプター管理テーブルを新たに作成し、メモリチャプター管理リスト６２に加える。そして、メモリ獲得要求を発行したユーザプログラムに対し、新規メモリチャプターの中から記憶領域を確保し割当を行い、その割当状況を表す情報を該当するメモリチャプター管理テーブルに記録する。最後に要求元へ処理が戻る。

【００３５】

チャプターサーバ側の処理は、メモリチャプター要求が同時に複数のホストから発行されることを考慮すると、上記検索・割り当ては排他的に行わなければならない。そこで、チャプターサーバは複数のホストからの割り当て要求を並行に処理することはせず、逐次的に処理を行う。

（実施例２）

ここでは、実施例１で説明したメモリチャプターに加え、新たにメモリセクションを加え、２種類の単位を用いてネットワーク単一仮想記憶空間（　以下、ＮＳＶＳと呼ぶ）　の管理を行う実施例について述べる。

【００３６】

実施例１で述べたように、空間全体が複数のメモリチャプターと呼ばれる領域に分割されている。このメモリチャプターは、仮想記憶空間を複数のホストで分担して管理する時の単位として用いられる。

【００３７】

各メモリチャプターはさらに複数のメモリセクションと呼ばれる領域に分割されている。このメモリセクションは、仮想記憶空間内に配置されるファイルやプログラムテキスト、データ等（以下プログラム等という）の記憶空間内の内容の違いによる割当の単位として用いらる。つまり、記憶領域の確保は、メモリセクション単位に行うため、ユーザプログラムは、新規のメモリセクションを確保するためのシステムコール（ＯＳに対するサービス要求）を発行することで獲得できる。

【００３８】

アクセス保護もメモリセクションを単位として行われる。アクセス保護とは、あるプログラム等へのアクセス（例えば読み出し、書き込み、実行）を、制限する機構のことをいう。つまり、このアクセスをどのスレッド（またはプロセス）が行おうとしているのかという情報や、このアクセスがどのプログラムから行われようとしているのかという情報に基づいて許可／禁止することを言う。スレッドとは、プログラムを実行する主体のことをいい、この実施例では、アクセス保護が許される範囲で、複数のメモリセクションをまたがってプログラムを実行し、また、複数のメモリセクション内のデータを読み書きすることができる。

【００３９】

このようなアクセス制御の具体的機構には、あるスレッド（またはプロセス）によるあるプログラム（またはメモリセクション）からのアクセスに対する許可／禁止を表す情報を、各メモリセクションに対応して登録し、プログラム実行時にこの情報を元にアクセス制御することにより、プログラム等に対するアクセス制御を実現する。このアクセス制御の機構は、特願平５－３９３７号に詳しい。

【００４０】

なお、仮想記憶空間の主記憶（物理メモリ）への写像は、一般的な仮想記憶空間の実現法であるページング方式を採用しているものとするが、他の方式であっても一般性を失わない。ページング方式の場合は、上記の各メモリセクションは複数のページから成る。つま

り、メモリセクションの大きさはページサイズの整数倍であるとする。
【００４１】
本実施例では仮想記憶空間の管理を効率良く行うために、メモリチャプター、メモリセクション、ページの各大きさを固定とし（それぞれ（２の４８乗）ｂｙｔｅ、（２の３２乗）ｂｙｔｅ、（２の１２乗）ｂｙｔｅ）、仮想記憶空間アドレスの上位１６ｂｉｔ　をメモリチャプターＩＤ、上位３２ｂｉｔ　をメモリセクションＩＤ、上位５２ｂｉｔ　をページＩＤとする。この様子を図１０に示す。
【００４２】
また、各ホスト上で管理されるスレッドＩＤは、ＮＳＶＳを形成するホストの範囲中でユニーク（お互いのスレッドＩＤが重複しない）であるとする。
まず、最初に本実施例の説明で必要となるデータ構造の管理に関して述べ、さらに、メモリ領域の管理手順について述べる。
［データ構造］
ＮＳＶＳが有効に機能するためには、メモリチャプターやメモリセクションがどのように割り当てられているかの管理が重要である。ここでは、メモリチャプターやメモリセクションの割り当て状況等の管理情報の配置について説明する。
【００４３】
ＮＳＶＳ内の各メモリチャプターがどのホストに割り当てられているかという情報の管理は、実施例１と同様、チャプターサーバと呼ばれるホストで集中管理している。この管理は図５に示すような表を用いて行っている。この表には、現在のチャプターＩＤとチャプターオーナーのホスト名および関連情報の対応が登録されている。
【００４４】
各ホストには、自ホストで現在利用しているメモリチャプターの情報をメモリチャプター管理リストで保持している。メモリチャプター管理リスト６３は、図１１に示す構造を採り、図８のメモリ領域（開始番地と大きさ）の代わりにメモリセクションＩＤを用い、利用状況（使用中または未使用）のフィールドを拡張して、このメモリセクションを所有するスレッドＩＤ　（またはプロセスＩＤ）　を用いるようになっている。この値が０の時は、空きセクションを表す。また、そのメモリチャプタを管理しているオーナーホスト名を表すフィールドを付加した。
【００４５】
あるメモリチャプター内の全てのメモリセクションに関する管理情報は、チャプターオーナーがこのメモリチャプター管理テーブルで保持する。
さらに、現在利用中のメモリセクションに関する各種の情報は、メモリセクション管理テーブルに記憶されている。メモリセクションの管理情報としては、メモリセクションＩＤ、そのメモリチャプターを管理するオーナースレッドのＩＤ、オーナーホスト名、アクセス保護情報、バッキングストア（そのメモリセクションの内容を持っているディスク装置等）との対応関係（対応するディスク内のアドレス等）、メモリセクションが他のホストと共有された場合（後述する）の内容のコピーの分配状況、およびコピー間の一貫性維持のための制御情報などがある。このメモリセクション管理テーブルの例（図１１のオーナーホストＨ－Ｂのもの）を図１２に示す。
［管理手順］
次に、ＮＳＶＳに基づく分散システムにおけるメモリ領域の管理手順について説明する。
【００４６】
管理手順は、大きく２つに分けられる。１つは、新規にメモリ領域を確保する場合であり、他方は、既に存在しているメモリ領域を利用する場合で他のホストと共有する場合である。
［新規メモリ領域確保］
新規にあるホスト上でメモリ領域を確保する場合、ユーザプログラムはメモリセクション確保のシステムコールを使ってＯＳに要求する。ＯＳでは、メモリチャプタ管理リストをたどりながら、メモリチャプタ管理テーブルのチャプターオーナが自ホスト名であるテー

ブルを探し、その中から空きメモリセクションを探す。あったら、要求したスレッドＩＤをオーナスレッドとして空きメモリセクションのオーナスレッドのところに登録する。要求元には、確保できたメモリセクションＩＤを返す。さらに、メモリセクション管理テーブルへも確保できたメモリセクションＩＤに関して新たなエントリを作成し、データを登録する。オーナスレッドは、新しいメモリセクションを作成すると同時に、以後、バッキングストアの管理、アクセス保護情報の管理、他ホストとのメモリ共有に関する管理などを行う。

【００４７】

もしそのホストに割り当てられたメモリチャプター内の全てのメモリセクションが使用中である場合には、そのホストは新たなメモリチャプターを確保する。メモリチャプターの確保は、実施例１と同様の方法である。したがって、各ホストに確保されたメモリチャプターの領域は、他のホストが確保した領域と重複しないことが保証されているので、このメモリチャプター内のすべてのメモリセクションは未使用であることが保証される。そして確保されたメモリチャプターに関して新規にメモリチャプタ管理テーブルを作成し、データを初期化し、登録する。そこから上記の手順でメモリセクションの割当を行う。

［既存メモリ領域利用］

次に、ＮＳＶＳに基づく分散システムにおける情報共有について考える。ＮＳＶＳに基づく分散システムでは、全てのホストから同じアドレス上に同じデータが存在するように見せているため、情報の共有は単に同じアドレスをアクセスすることをきっかけにして行われる。

【００４８】

本実施例では、メモリセクション毎にアクセス制御が行われているため、情報の共有に際しては、まずメモリセクションの共有を行わなければならない。メモリセクションの共有は、あるホストが上述の手続きにより確保したメモリセクションの内容を、他のホストが参照さらには更新できるようにすることにより実現される。

【００４９】

ここで、ホストＨ－ＡがメモリセクションＭＳ－Ａを確保し単独で利用している状態で、ホストＨ－Ｂとそのメモリセクションを共有するまでの具体的手順について説明する。メモリセクションＭＳ－Ａのチャプターは、ＭＣ－Ａであり、そのオーナーホスト名は、ホストＨ－Ａである。

【００５０】

まず、ホストＨ－Ｂ上で動いているプログラムがメモリセクションＭＳ－Ａ内のあるアドレスをアクセスする。アクセス手段は、ホストＨ－Ｂ上で動作しているスレッドがそのアドレスへコールやジャンプした場合や、そのアドレスのデータを参照した場合などである。しかしメモリセクションＭＳ－Ａはまだホスト Ｈ－Ｂにアタッチされていない、つまり、ホストＨ－Ｂで、そのメモリセクションはまだ利用実績がないので、ホストＨ－Ｂの仮想記憶管理部６２には、メモリセクションＭＳ－Ａが登録されていない。したがって、アクセス時にページフォルトが発生し、ＯＳのページフォルト処理手続きが呼び出される。

【００５１】

ページフォルトが生じると、ＯＳはまずフォルトを起こしたページがバッキングストアに退避されているページかどうかを調べる。

これは、メモリセクション管理テーブルを調べることで判明する。つまり、自ホストがチャプターオーナである利用中のメモリセクションの場合には、メモリセクション管理テーブルに登録されており、そのメモリセクションの内容が保管されているバッキングストアが登録されているからである。登録されている際は、バッキングストアに退避されているページを、通常のページング方式に従い主記憶（物理メモリ）にロードし、フォルトを起こしたスレッド（　またはプロセス）　を再開させるための手続きを行う。

【００５２】

また、他のホストがチャプターオーナであるメモリセクションの場合には、登録されている分配状況、一貫性制御情報などにしたがって、オーナホストからデータを転送し、主記

憶（物理メモリ）にロードし、フォルトを起こしたスレッド（　またはプロセス）　を再開させるための手続きを行う。

【００５３】

しかし、メモリセクション管理テーブルに登録されていない場合、他のホストが保持するページに対するアクセスと解釈され、チャプターサーバに対してページＩＤ（　または、メモリセクションＩＤ）　を送り、そのページが含まれるメモリチャプターのオーナーの検索を依頼する。

【００５４】

具体的には、Ｈ－ＢのＯＳはページフォルトを起こしたアドレスから、メモリチャプタＩＤを求める。これは、アドレス上位１６ビットがメモリチャプタＩＤなので求められる。

【００５５】

次に、自ホスト内のメモリチャプタ管理テーブルをサーチして、このチャプタＩＤが登録されているか調べる。あれば、そのテーブルのオーナーホスト名を調べる。なければ、自分の知らないメモリセクションであるとを判断し、他のホストが保持するメモリセクションへのアクセスであると見なして、そのメモリセクションが含まれるメモリチャプターのオーナーがどのホストであるかをチャプターサーバに問い合わせる。チャプターサーバはチャプターＩＤと、それを保持し管理するホストの識別子の対応を保持するＮＳＶＳ管理テーブル（図５）を持っており、その表からチャプターオーナーを調べてホストＨ－Ｂに伝達する。

【００５６】

検索の結果、そのメモリチャプターが未使用でオーナがいない場合、フォルトを起こしたメモリアクセスは不当として処理される。

チャプターオーナーＩＤが得られた場合には、そのホストに対してアクセス要求を送るとともに、アクセス保護のチェックを受けるために要求したスレッドＩＤ（　またはプロセスＩＤ）　などの情報を送る。チャプターオーナーでアクセスが許可された場合には、オーナーホストから該当ページ（またはそれを含むメモリセクション全体）のコピーが送られてくるので、それを受けとり、フォルトを起こした（要求した）スレッド（　またはプロセス）　を再開させるための手続きを行う。さらに、そのメモリチャプターやメモリセクションを管理するためのメモリチャプター管理テーブルを必要なら要求し、送られてたテーブルをメモリチャプター管理リストに加える。既に、存在している場合は、更新される。

【００５７】

また、そのメモリセクションに関する情報をメモリセクション管理テーブルに追加する。具体的には、ホストＨ－Ｂは得られたチャプターオーナー（この場合ホストＨ－Ａ）に対して、アクセスしたスレッドＩＤ（またはプロセスＩＤ）と共に、メモリセクションＭＳ－Ａの共有要求を送り、メモリセクションの内容（コピー）と必要に応じてそのメモリチャプター管理テーブルを受けとる。

【００５８】

一方、要求を受けた側のチャプターオーナーＨ－Ａは、要求元のスレッドＩＤをメモリセクション管理テーブルのアクセス保護情報に従ってアクセス許可を判断し、許可されると、該当メモリセクションのコピーをホストＨ－Ｂに送り、自分の保持するメモリセクションのコピーがホストＨ－Ｂにできたことを自ホスト内のメモリセクション管理テーブルの分配状況に登録するとともに、コピー間の一貫性維持のための一貫性制御情報に登録して、ロックの管理や、更新されたデータの配布、ホストＨ－Ｂに対するメモリセクションＭＳ－Ａのバッキングストアのサービス等を提供する。データの更新が生じた時やロック／　アンロックなどのイベントが生じた時に、コピーを持つホストＨ－Ｂに伝達する。要求に応じて、メモリチャプター管理テーブルの内容を送る。

【００５９】

オーナーホストにて使用中のメモリセクションが不要になり、メモリセクション解放要求が発行されると、ＯＳはメモリチャプター管理リスト（図１１）の中を検索し、解放され

たメモリセクションの状態を未使用に変更し、ページテーブルからそのメモリセクションに属するページを全て削除する。さらに、メモリセクション管理テーブルからも該当メモリセクションのエントリの開放作業を行う。他ホストと共有関係にある場合は、それを加味した開放作業を行う。

［チャプターサーバ］

上述のようにチャプターサーバは、新たなメモリチャプターを割り当てる処理と、アドレスまたはメモリセクションＩＤまたはメモリチャプターＩＤからチャプターオーナーがどのホストであるかを検索・応答する処理の２種類のサービスを提供する。

【００６０】

このサービスを提供するために、チャプターサーバはＮＳＶＳ管理テーブル（図５）を内部的に保持している。このテーブルはメモリチャプターＩＤをキーとするテーブルであり、チャプターオーナーの検索が高速に行えるように構成されている。さらに、このテーブルをリスト構造とすれば、未使用領域の検索を高速に行える。

【００６１】

つぎにチャプターサーバの処理手順について図１３を用いて説明する。チャプターサーバは分散システム内のホストが発行するサービス要求を受信し（Ｓ１１）、その要求が新規メモリチャプター要求である場合には（Ｓ１２Ｙｅｓ　）、ＮＳＶＳ管理テーブル４４から未使用メモリチャプターを検索し（Ｓ１５）、そのＩＤを要求を発行したホストに返す（Ｓ１６）。尚、空きメモリチャプターがない場合は、その旨を要求元のホストに伝える。この時、ＮＳＶＳ管理テーブル４４内の未使用メモリチャプターがリストを構成していると、この検索が高速に行える。

【００６２】

サービス要求がチャプターオーナーの問い合わせの場合には（Ｓ１３Ｙｅｓ　）、要求とともに送られてきたチャプターＩＤをキーとしてＮＳＶＳ管理テーブル４４を引くことにより、簡単かつ高速にチャプターオーナーのＩＤを得ることができ、それを要求を発行したホストに返す（Ｓ１４）。以上の処理を繰り返すことによりチャプターサーバの機能が実現される。また、サービス要求が発行されない間は、分散システムの他のホストと同様、普通の仕事もすることができる。

（実施例３）

本実施例に係るホストの構成を図１４に示す。本実施例で実施例１の構成と異なる点は、ディスク管理部６６及びその内部のディスク領域管理テーブル６７が付加される点である。

【００６３】

この実施例では、自己がそのチャプターオーナであるチャプター（即ち、ネットワーク単一仮想記憶空間中の自ホストが管理している領域）に割り当てられているデータを実際に格納する領域として、自ホストに接続されているディスクの領域を利用する。

【００６４】

以下、図１５のフローチャートに従って説明する。あるホスト上で、動作中のユーザ、アプリケーションプログラムから仮想空間の一部領域の獲得要求（システムコール）が発行されると、その要求は、サービス要求受付部６４に入る（Ｓ１）。つぎに、要求内容がメモリ領域の獲得要求かどうか調べ（Ｓ２）、そうであると、仮想記憶空間管理部６２に処理が進む。そうでないと、その他の処理実行部６５に処理が進む（Ｓ３）。

【００６５】

仮想記憶空間管理部６２では、まず、すでに獲得してあるメモリチャプター内の空き領域から、メモリチャプター管理リスト６３を用いて要求サイズを満たす大きさのものを検索する（Ｓ４）。この検索の結果を判定し（Ｓ５）、メモリ獲得要求を満たす大きさの空き領域がある場合は、そこからメモリ領域を確保し割り当てを行い、メモリチャプター管理テーブルを更新する（Ｓ６）。次に、確保した領域に対するディスク領域の割当を要求するために、ディスク管理部６６へ、ディスクの割当を要求する（Ｓ７）。ディスク管理部６６からの返答により、ディスクの割当が確保できたか判定し（Ｓ８）、確保できないと

エラー処理をして終える（Ｓ９）。確保できると、そのディスクの領域（ディスクアドレス）をメモリチャプター管理テーブルにあわせて登録する（Ｓ１０）。

【００６６】

この場合のメモリチャプター管理リスト６３の構造は図１６のようになり、ディスク内の記憶領域の位置（ディスクアドレス）など必要な情報を、仮想記憶空間上の領域と対応させて記憶するようになっている。そして最後に、メモリ獲得サービスを要求を発行元（スレッドまたはプロセス、タスク）に対して獲得できた仮想記憶空間上の領域の開始アドレス他必要な情報をサービス要求受付部６４を介して伝え、処理を終える（Ｓ１１）。

【００６７】

要求を満たす大きさの空き領域が、ホストの保持するメモリチャプター内に存在しない場合は、チャプターサーバー通信部６１を介してチャプターサーバに対し新規メモリチャプターの獲得要求を発行する（Ｓ１２）。チャプターサーバをもつホストでは、通信部４１を介して新規メモリチャプターの獲得要求を受けとったチャプターサーバ内のＮＳＶＳ管理部４３が、空きメモリチャプターをＮＳＶＳ管理テーブル４４を用いて検索し、要求を発行したホストに割り当てる。さらに、ＮＳＶＳ管理テーブル４４内の割り当てたメモリチャプターに該当する欄に、割り当てたホストＩＤを記入し状態を使用中に変更する。以上のチャプターサーバ内の処理は、実施例１と同様である。

【００６８】

要求元のホスト側では、チャプターサーバ通信部６１を介してチャプターサーバから返答がくると、この返答から、新規メモリチャプターが獲得できたかチェックする（Ｓ１３）。獲得できなかったときは、エラー処理をして終える（Ｓ１５）。新規メモリチャプターを受けとったホストは、受けとったメモリチャプターの管理のために、メモリチャプター管理テーブルを新たに作成し、メモリチャプター管理リスト６３に加える（Ｓ１４）。そして、メモリ獲得要求を発行したスレッド（またはプロセス、タスク）と、新規メモリチャプターに関して上記Ｓ６以降の処理を行う。

【００６９】

以下に、ディスク管理部６６の動作を図１７を用いながら説明する。ディスク管理部６６には、ディスクの利用状況を管理するディスク領域管理テーブル６７がある。その構成は図１８に示すように、ディスクのセクター単位に管理されたディスクアドレスとその利用状況を記憶するようになっている。テーブルの大きさは、全物理ディスク容量をカバーできる範囲である。

【００７０】

まずディスク管理部６６内にある要求受付部が要求を受け付ける（Ｓ２１）。要求内容は、ディスク領域の要求、解放などである。要求の場合は、パラメータとして容量を添える。また、解放の場合は、解放するディスクアドレスを添える。

【００７１】

以下の説明では、領域要求の場合（Ｓ２２Ｙｅｓ　）で説明する。要求された容量を、まず、テーブル６７の空き容量に格納されている値と比較する（Ｓ２４）。足りる場合は、要求量を引いて空き容量のデータを更新する（Ｓ２６）。不足する場合は、要求元（仮想記憶空間管理部６２）にその旨を伝えて処理を終了する（Ｓ２５）。

【００７２】

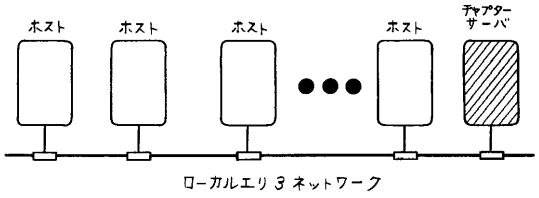
足りた場合、次に空きセクターを探して確保する。空きポインタによって空きセクターがリスト構造でつながっているので、必要な量のセクターをリストをたどって取り出す。つまり必要な数のセクター分、ポインタをたどりながらポイント先のアドレスを記憶し、このアドレスのエントリの状態フィールドを使用中にかえていく（Ｓ２７，Ｓ２８，Ｓ２９Ｎｏ，Ｓ３１）。

【００７３】

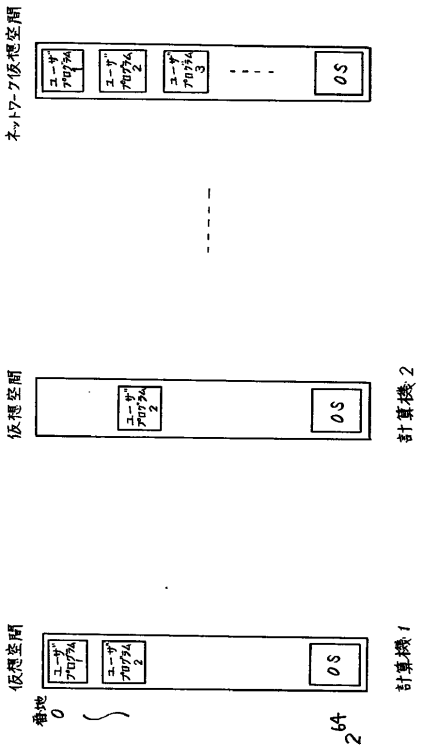
必要な数のセクタを確保したら（Ｓ２９Ｙｅｓ　）、記憶しておいたアドレスを要求元に返して処理を終了する（Ｓ３０）。尚、ディスクの解放の処理は、上記と逆の手順をふみ、返されたセクターの分だけ空き容量を増やし、空きポインタに返されたセクターのアド

レスをつなぎ、そのアドレスの状態を未使用にする（Ｓ２３の例）。
【００７４】
【発明の効果】
以上、説明したように、本発明によれば、複数の計算機をネットワークを介して結合した分散環境において、メモリーチャプターという仮想空間を比較的大きく分割する管理単位を導入し、この単位を用いて分散環境内の各計算機に記憶領域を割り当てる。
【００７５】
一度、計算機にメモリチャプターを割当てられると、その内の管理は、自計算機内だけで独占して行えるので、分散環境下でも、プログラム実行時のプログラムコードやデータなどを格納するところである仮想空間の領域の管理が他の計算機と交信して決めることなく独立して行えるので、通信量を減らせ、オーバヘッド少ない仮想空間管理が可能になる。
【００７６】
よって、分散環境下でのオペレーティングシステム（ＯＳ）が行っている仮想空間の管理機能において、どの計算機からでも同じ仮想空間を共有する仮想空間の管理法の効率化が実現できる。
【図面の簡単な説明】
【図１】本発明の一実施例のシステム構成を示す図。
【図２】本実施例システムで管理する仮想記憶空間の構成例を示す図。
【図３】図２とは別の構成例を示す図。
【図４】チャプターサーバの内部構成を示す図。
【図５】ＮＳＶＳ管理テーブルの構造例を示す図。
【図６】各ホストの内部構成を示す図。
【図７】メモリチャプター管理リストの構造例を示す図。
【図８】メモリチャプター管理テーブルの構造例を示す図。
【図９】実施例１におけるチャプターサーバの動作の流れを示すフローチャート。
【図１０】仮想記憶空間アドレスの使用法を表す図。
【図１１】メモリチャプター管理リストの別の構造例を示す図。
【図１２】メモリセクション管理テーブルの構造例を示す図。
【図１３】実施例２におけるチャプターサーバの動作の流れを示すフローチャート。
【図１４】実施例３における各ホストの内部構成を示す図。
【図１５】実施例３のシステムの動作の流れを示すフローチャート。
【図１６】メモリチャプター管理リストのさらに別の構造例を示す図。
【図１７】ディスク管理部の動作の流れを示すフローチャート。
【図１８】ディスク領域管理テーブルの構造例を示す図。
【符号の説明】
４１、６１…通信部
４２、６２…仮想記憶空間管理部
４３…ＮＳＶＳ管理部
４４…ＮＳＶＳ管理テーブル
６３…メモリチャプター管理リスト
６４…サービス要求受付部
６５…その他の処理実行部
６６…ディスク管理部
６７…ディスク領域管理テーブル

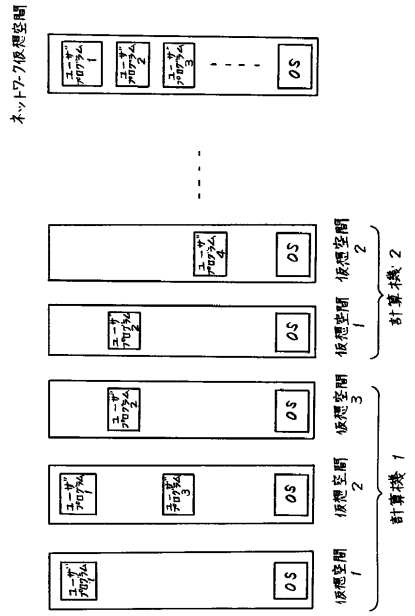
【図１】


ホスト
ホスト
ホスト
ホスト
チャプターサーバ
ローカルエリアネットワーク

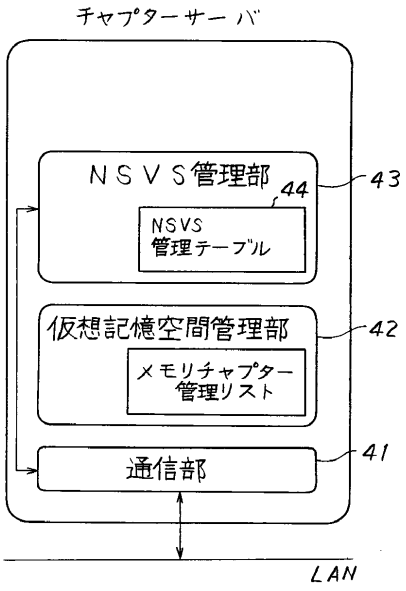
【図２】


ネットワーク仮想空間
ユーザプログラム1
ユーザプログラム2
ユーザプログラム3
OS
仮想空間
計算機2
仮想空間
計算機1
番地
0
2^64

【図３】


ネットワーク仮想空間
仮想空間1
仮想空間2
仮想空間3
計算機1
計算機2
OS

【図４】

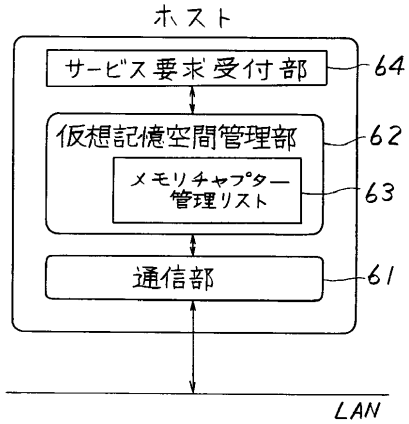

チャプターサーバ
NSVS管理部
44
43
NSVS
管理テーブル
仮想記憶空間管理部
42
メモリチャプター
管理リスト
通信部
41
LAN

【図５】

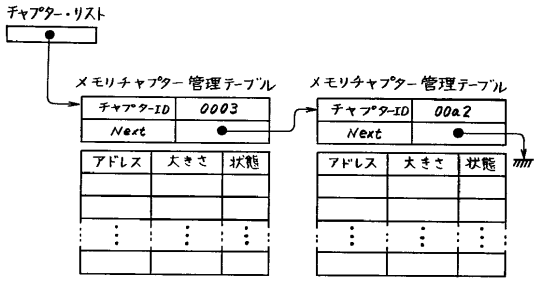
(各IDは16進数表記)

チャプター ID	ホスト ID	状態
0000	—	未使用
0001	1c02	使用中
0002	—	未使用
0003	930f	使用中
⋮	⋮	⋮
ffff	—	未使用

【図６】



【図７】



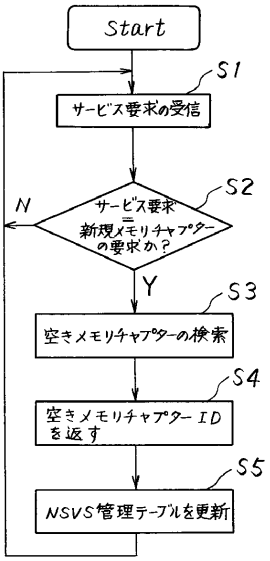
【図８】

(各IDは16進数表記)

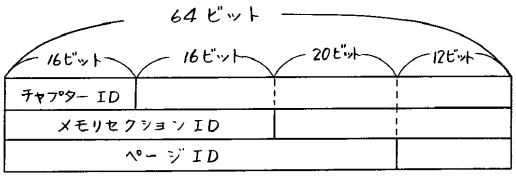
チャプター ID	0003
次のチャプターへのポインタ	0007

開始アドレス	大きさ	状態
0003 0000 0000 0000	1 0000 0000	未使用
0003 0001 0000 0000	1 0000 0000	使用中
0003 0002 0000 0000	3 5000 0000	使用中
0003 0005 5000 0000	5000 0000	未使用
⋮	⋮	⋮
0003 ffff 0000 0000	1 0000 0000	未使用

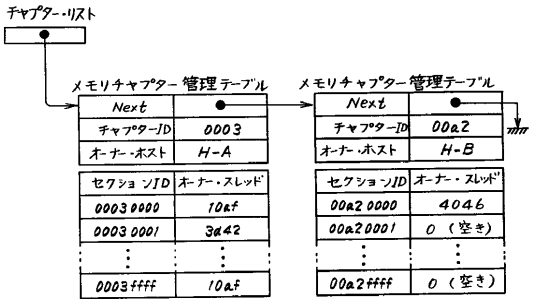
【図９】



【図１０】



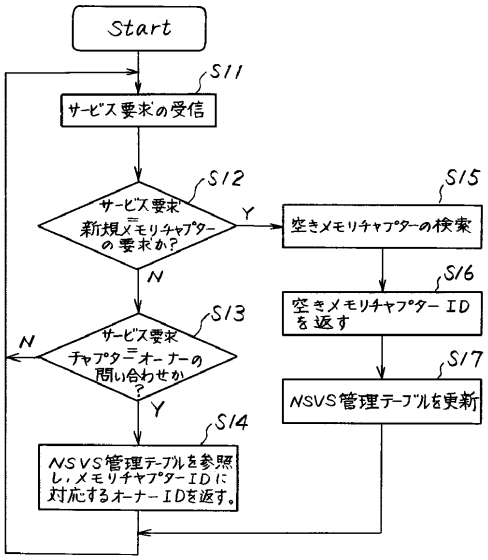
【図１１】



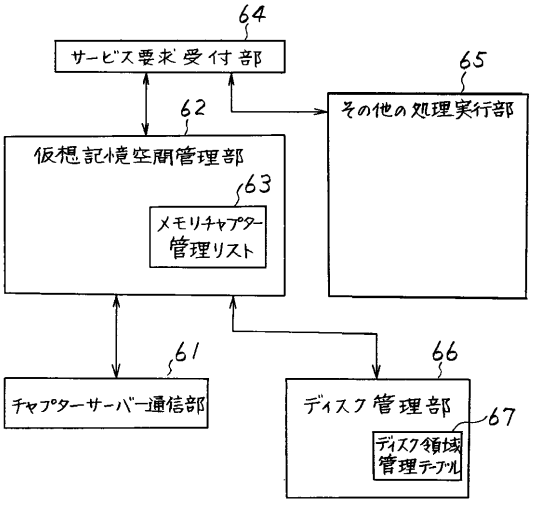
【図１２】

メモリゼクションID	オーナスレッド	オーナホスト	アクセス律速スレッド/バッファ	バッキングストア	分配状況	一貫性制御情報	その他
0003 0000	10af	H-A	aa2 / rw-	ディスクA 0100	H-B	Copy，バージョン3，…	
00a2 0000	4046	H-B	aa2 / r-x	—	H-A，H-D	バージョン11，…	

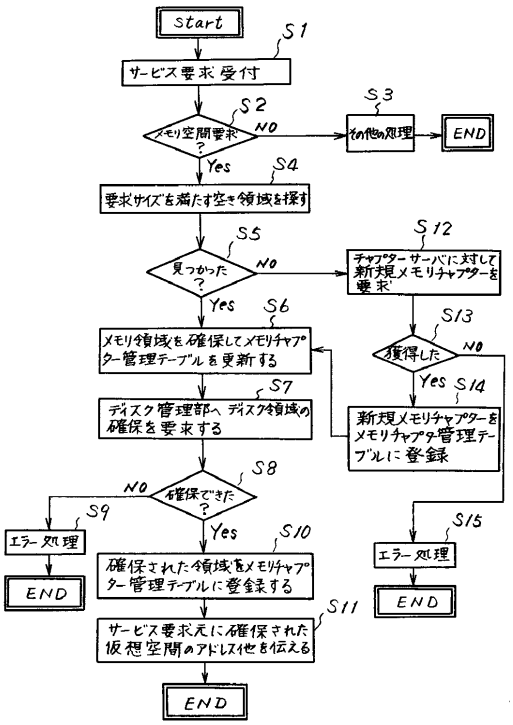
【図１３】



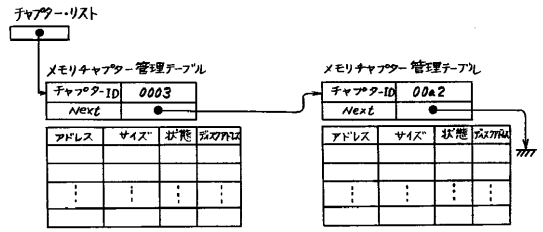
【図１４】



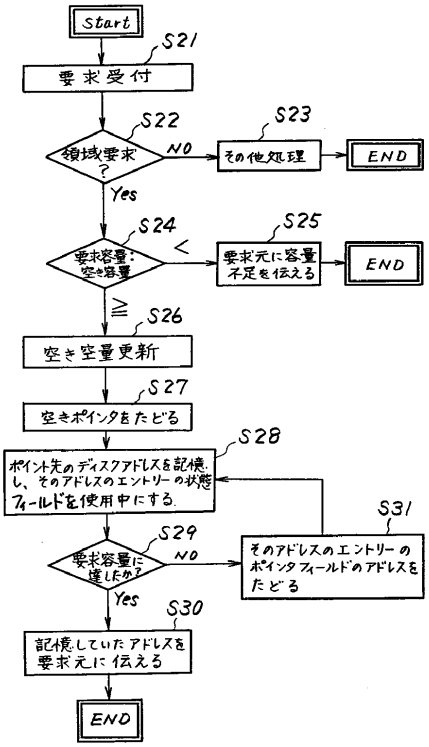
【図１５】



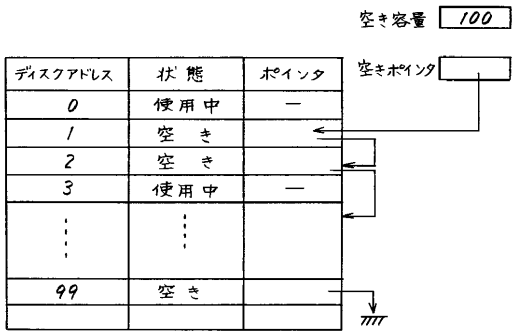
【図１６】


チャプター・リスト
メモリチャプター管理テーブル
チャプタ-ID
0003
Next
アドレス
サイズ
状態
ディスクアドレス
メモリチャプター管理テーブル
チャプタ-ID
0002
Next
アドレス
サイズ
状態
ディスクアドレス

【図１７】


Start
S21
要求受付
S22
領域要求?
NO
S23
その他処理
END
Yes
S24
要求容量:空き容量
<
S25
要求元に容量不足を伝える
END
≧
S26
空き空量更新
S27
空きポインタをたどる
S28
ポイント先のディスクアドレスを記憶し、そのアドレスのエントリーの状態フィールドを使用中にする
S29
要求容量に達したか?
NO
S31
そのアドレスのエントリーのポインタフィールドのアドレスをたどる
Yes
S30
記憶していたアドレスを要求元に伝える
END

【図１８】


空き容量
100
空きポインタ
ディスクアドレス
状態
ポインタ
0
使用中
—
1
空き
2
空き
3
使用中
—
99
空き

フロントページの続き

審査官　鳥居　稔

(56)参考文献　特開平０４－３６２７５１（ＪＰ，Ａ）
特開平０５－２５７８０３（ＪＰ，Ａ）
特開平０５－２５７８１１（ＪＰ，Ａ）
特開平０６－０５２０５４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.[7]，ＤＢ名)
G06F 9/46
G06F 12/08-10
G06F 15/16