(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2010-39895
(P2010-39895A)
(43)公開日　平成22年2月18日(2010.2.18)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 12/08	(2006.01)	G06F	12/08	541Z	5B005
G06F 9/46	(2006.01)	G06F	9/46	350	5B018
G06F 12/10	(2006.01)	G06F	12/10	553Z	
G06F 12/16	(2006.01)	G06F	12/08	501D	
		G06F	12/16	320D	

審査請求 未請求　請求項の数 7　OL　（全 21 頁）　最終頁に続く

(21)出願番号　特願2008-203968 (P2008-203968)
(22)出願日　平成20年8月7日 (2008.8.7)

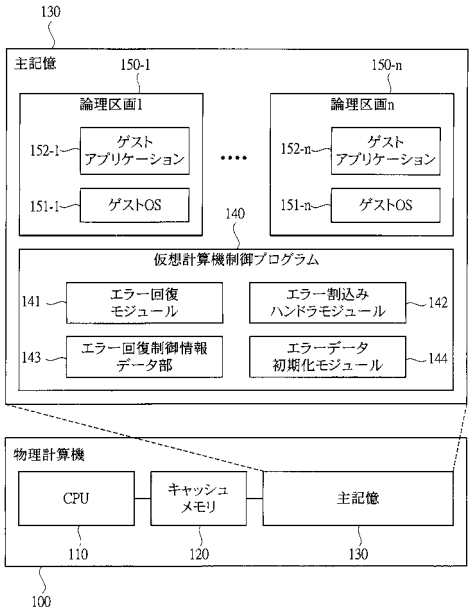
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(71)出願人　000233295
日立情報通信エンジニアリング株式会社
神奈川県横浜市戸塚区戸塚町３９３番地
(74)代理人　100080001
弁理士　筒井　大和
(72)発明者　小林　孝
神奈川県横浜市西区みなとみらい二丁目３番３号　日立情報通信エンジニアリング株式会社内

最終頁に続く

(54)【発明の名称】仮想計算機システムおよび仮想計算機システムにおけるエラー回復方法ならびに仮想計算機制御プログラム

(57)【要約】
【課題】ハードウェアの追加を必要とせず、かつ、ゲストプログラムにエラー回復手段を実装することを必要とせず、キャッシュメモリエラーを回復することを可能とする仮想計算機システムを提供する。
【解決手段】物理計算機１００上で、仮想計算機制御プログラム１４０を実行することにより、各論理区画１５０上でゲストプログラムを動作させる仮想計算機システムであって、仮想計算機制御プログラム１４０は、周期的にキャッシュメモリ１２０のエラーを回復するエラー回復モジュール１４１と、キャッシュメモリ１２０で発生したエラーによる割込み通知に対して、キャッシュメモリ１２０のエラーを回復するエラー割込みハンドラモジュール１４２と、論理区画１５０のシャットダウンもしくは再起動を契機に、キャッシュメモリ１２０のエラーを回復するエラーデータ初期化モジュール１４４とを有し、キャッシュメモリ１２０のエラーの回復処理を行う。
【選択図】図１

図 1



【特許請求の範囲】
【請求項１】
物理計算機上で、前記物理計算機を複数の論理区画に分割し、前記各論理区画に前記物理計算機の計算機資源を割当てて制御する仮想計算機制御プログラムを実行することにより、前記各論理区画上でそれぞれゲストＯＳを含むゲストプログラムを動作させる仮想計算機システムであって、
前記仮想計算機制御プログラムは、
周期的に前記物理計算機のキャッシュメモリのエラーを回復するエラー回復処理を行うエラー回復モジュールと、
前記キャッシュメモリで発生したエラーによる割込み通知に対して、前記キャッシュメモリのエラーを回復するエラー割込み処理を行うエラー割込みハンドラモジュールと、
前記論理区画のシャットダウンもしくは再起動を契機に、前記キャッシュメモリのエラーを回復するエラーデータ初期化処理を行うエラーデータ初期化モジュールとを有し、
前記各論理区画上で動作する前記ゲストプログラムの動作とは独立して、前記キャッシュメモリのエラーの回復処理を行うことを特徴とする仮想計算機システム。
【請求項２】
物理計算機上で、前記物理計算機を複数の論理区画に分割し、前記各論理区画に前記物理計算機の計算機資源を割当てて制御する仮想計算機制御プログラムを実行することにより、前記各論理区画上でそれぞれゲストＯＳを含むゲストプログラムを動作させる仮想計算機システムにおけるエラー回復方法であって、
前記仮想計算機制御プログラムにより、
前記論理区画上で動作する前記ゲストプログラムを停止させるステップと、
前記物理計算機のキャッシュメモリの各エントリに対して、前記エントリが有効である場合に前記エントリを無効とするステップと、
無効とされた前記エントリに前記物理計算機の主記憶からデータをロードし直すことで前記キャッシュメモリのエラーを回復するステップと、
前記論理区画上の前記ゲストプログラムの動作を再開させるステップとを含むエラー回復処理を実行することを特徴とする仮想計算機システムにおけるエラー回復方法。
【請求項３】
請求項２に記載の仮想計算機システムにおけるエラー回復方法において、
前記仮想計算機制御プログラムが、前記キャッシュメモリの対象の前記エントリに前記物理計算機の主記憶からデータをロードし直すことで前記キャッシュメモリのエラーを回復する際に、前記キャッシュメモリにおける訂正不可能なエラーに起因するエラー割込みの通知を受けた場合、
前記仮想計算機制御プログラムにより、
前記訂正不可能なエラーが発生した前記キャッシュメモリの前記エントリのアドレスであるエラーアドレスを求めるステップと、
前記エラーアドレスが、いずれかの前記論理区画に割当てられたアドレス領域に含まれるか否かを判定するステップと、
前記判定が真である場合に、該当する前記論理区画上で動作する前記ゲストＯＳへエラー割込みを通知するステップと、
前記判定が偽である場合に、前記エラーアドレスに対応する前記エントリに初期化データをストアすることで、前記キャッシュメモリのエラーを回復するステップ、および、
前記エラーアドレスに対応する前記エントリを無効とすることで、前記主記憶に前記キャッシュメモリのデータを反映させるステップとを含むエラー割込み処理を実行することを特徴とする仮想計算機システムにおけるエラー回復方法。
【請求項４】
請求項３に記載の仮想計算機システムにおけるエラー回復方法において、
前記エラーアドレスを含む前記論理区画のシャットダウンもしくは再起動を契機として、

前記仮想計算機制御プログラムにより、
前記エラーアドレスに対応する前記エントリに初期化データをストアすることで、前記キャッシュメモリのエラーを回復するステップと、
前記エラーアドレスに対応する前記エントリを無効とすることで、前記主記憶に前記キャッシュメモリのデータを反映させるステップとを含むエラーデータ初期化処理を実行することを特徴とする仮想計算機システムにおけるエラー回復方法。
【請求項５】
物理計算機上で実行され、前記物理計算機を複数の論理区画に分割し、前記各論理区画に前記物理計算機の計算機資源を割当てて制御することにより、前記物理計算機を、前記各論理区画上でそれぞれゲストＯＳを含むゲストプログラムを動作させる仮想計算機システムとして機能させる仮想計算機制御プログラムであって、
前記論理区画上で動作する前記ゲストプログラムを停止させるステップと、
前記物理計算機のキャッシュメモリの各エントリに対して、前記エントリが有効である場合に前記エントリを無効とするステップと、
無効とされた前記エントリに前記物理計算機の主記憶からデータをロードし直すことで前記キャッシュメモリのエラーを回復するステップと、
前記論理区画上の前記ゲストプログラムの動作を再開させるステップとを含むエラー回復処理を実行することを特徴とする仮想計算機制御プログラム。
【請求項６】
請求項５に記載の仮想計算機制御プログラムにおいて、
前記キャッシュメモリの対象の前記エントリに前記物理計算機の主記憶からデータをロードし直すことで前記キャッシュメモリのエラーを回復する際に、前記キャッシュメモリにおける訂正不可能なエラーに起因するエラー割込みの通知を受けた場合、
前記訂正不可能なエラーが発生した前記キャッシュメモリの前記エントリのアドレスであるエラーアドレスを求めるステップと、
前記エラーアドレスが、いずれかの前記論理区画に割当てられたアドレス領域に含まれるか否かを判定するステップと、
前記判定が真である場合に、対象の前記論理区画上で動作する前記ゲストＯＳへエラー割込みを通知するステップと、
前記判定が偽である場合に、前記エラーアドレスに対応する前記エントリに初期化データをストアすることで、前記キャッシュメモリのエラーを回復するステップ、および、
前記エラーアドレスに対応する前記エントリを無効とすることで、前記主記憶に前記キャッシュメモリのデータを反映させるステップとを含むエラー割込み処理を実行することを特徴とする仮想計算機制御プログラム。
【請求項７】
請求項６に記載の仮想計算機制御プログラムにおいて、
前記エラーアドレスを含む前記論理区画のシャットダウンもしくは再起動を契機として、
前記エラーアドレスに対応する前記エントリに初期化データをストアすることで、前記キャッシュメモリのエラーを回復するステップと、
前記エラーアドレスに対応する前記エントリを無効とすることで、前記主記憶に前記キャッシュメモリのデータを反映させるステップとを含むエラーデータ初期化処理を実行することを特徴とする仮想計算機制御プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、仮想計算機システムにおけるエラー回復技術に関し、特に、キャッシュメモリのエラー回復技術に適用して有効な技術に関するものである。
【背景技術】
【０００２】

従来、複数の物理計算機で別々に稼動していたＯＳ（オペレーティングシステム）およびＯＳ上で稼動するソフトウェアを、１台の物理計算機で稼動させることを可能にする技術として、仮想計算機技術がある。

【０００３】

仮想計算機技術では、例えば、ハイパバイザと呼ばれる仮想計算機制御プログラムが、１つの物理計算機を複数の論理区画に論理的に分割する。仮想計算機制御プログラムは、分割された各論理区画に計算機資源（ＣＰＵ（中央演算処理装置）、主記憶およびＩ／Ｏ（入出力装置））を割当てる。この論理区画上で仮想計算機制御プログラムの制御によりＯＳ（ゲストＯＳ）が動作する。

【０００４】

この仮想計算機技術は、従来は、汎用機（メインフレーム）等の大型計算機で用いられてきた技術である。しかし、近年のマイクロプロセッサの性能向上等によって、ローエンドのＰＣサーバにも適用されるようになってきている。このようなローエンドのＰＣサーバを、企業ビジネス等で使用するミッション・クリティカルなサーバに適用することは、コスト低減を図る上で有利であり、ニーズが高いといえる。

【０００５】

一方、企業ビジネスの国際化やインターネットを代表とするコンピュータネットワークのグローバル化を背景として、計算機システムの長時間連続稼動（２４時間、３６５日稼動）の必要性が高まってきている。この必要性は、ローエンドのＰＣサーバを用いた仮想計算機システムを用いる場合にも当然あてはまる。

【０００６】

従来、計算機システムにおいて大容量メモリといえば、主記憶が主流であり、主記憶の大容量化に比例して主記憶エラーの発生確率が高かった。しかし、近年は、ＣＰＵから主記憶のデータへのアクセス性能を向上させるために用いられるキャッシュメモリの大容量化に伴い、キャッシュメモリエラーの発生確率も高い傾向にある。

【０００７】

キャッシュメモリの大容量化に伴い、データがキャッシュメモリに長く滞在する確率が増え、キャッシュメモリにのみ最新データが存在するという場合も増えてくる。このため、仮想計算機システムにおいて、長時間連続稼動を実現するためには、主記憶エラーのみならず、キャッシュメモリエラーの発生時にシステムを継続稼動させる技術が非常に重要となる。

【０００８】

メモリのエラー回復に関しては、従来から様々な技術が提案されている。例えば、特開平６－５２０４９号公報（特許文献１）には、プロセッサで実行中の処理の開始から終了までにアクセスするデータを、キャッシュメモリで中間状態として管理し、書き換え前のブロックの内容をメインメモリに書き戻し、中間状態のブロックに対する書き換えはキャッシュメモリに格納されているブロックのみを書き換え、実行中の処理が中止する場合には、キャッシュメモリ上の書き換えたブロックのみを無効化することによって、メインメモリの内容の回復を行う技術が開示されている。

【０００９】

上記以外のメモリエラー回復技術として、プロセッサによるメモリへのアクセスとは別に、メモリに格納された全データを周期的にエラーチェックする装置が提案されている。すなわち、プロセッサからのメモリアクセスとは別に、ＲＡＭ（Random Access Memory）チップに対して、周期的に、全てのデータに対して順番にエラーチェックを行なうメモリスクラビング方法が用いられている。メモリスクラビング方法に関連する技術としては、例えば、特開平８－１９４６４８号公報（特許文献２）に記載されている技術がある。

【００１０】

このエラーチェックにてデータにエラーが発見された場合は、エラーの生じた行の全てのアドレスのデータコードをＲＡＭチップから一つずつ取り出してＥＣＣ（Error Correcting Code）チェックを行ない、エラーの訂正が可能であればデータの誤りを訂正するこ

とが行なわれている。なお、この種の技術に関連するものとしては、例えば、特開平１－１１２５９９号公報（特許文献３）、特開昭６３－２６９２３３号公報（特許文献４）などに開示されている技術が挙げられる。

【特許文献１】特開平６－５２０４９号公報
【特許文献２】特開平８－１９４６４８号公報
【特許文献３】特開平１－１１２５９９号公報
【特許文献４】特開昭６３－２６９２３３号公報

【発明の開示】
【発明が解決しようとする課題】
【００１１】
特許文献１に開示されている技術は、キャッシュメモリ上の書き換えたブロックのみを無効化することによって、メインメモリの内容の回復を行うという技術である。従って、キャッシュメモリエラーを回復するためにキャッシュメモリに対して適用することは容易ではないといえる。
【００１２】
また、特許文献２などで用いているメモリスクラビング技術は、一般に、プロセッサによるメモリアクセスとの競合時におけるメモリデータの整合性を保つために、専用のハードウェアを備える必要がある。または、ソフトウェアによるメモリアクセスの排他制御等の技術が必要である。また、キャッシュメモリを対象とする場合、キャッシュメモリをスクラビングするための手段が別途必要となる。
【００１３】
しかし、例えば、ローエンドのＰＣサーバにも適用可能な仮想計算機システムにおいて、キャッシュメモリエラーの回復を実現するためのハードウェアを追加することは、コスト面で望ましくない。
【００１４】
また、仮想計算機システムの場合は、複数の論理区画上で複数の様々な種類のゲストプログラム（ゲストＯＳ、ゲストアプリケーション）が稼動するシステムである。従って、ソフトウェアによるキャッシュメモリエラーの回復についても、これら全てのゲストプログラムに、長時間連続稼動を達成するほどのキャッシュメモリエラー回復機能の実装を要求することは、非現実的であり望ましくない。
【００１５】
そこで本発明の目的は、ハードウェアの追加を必要とせず、かつ、ゲストプログラムにエラー回復手段を実装することを必要とせず、キャッシュメモリエラーを回復することを可能とする仮想計算機システムおよびエラー回復方法、ならびにこの方法を実行する仮想計算機制御プログラムを提供することにある。本発明の前記ならびにその他の目的と新規な特徴は、本明細書の記述および添付図面から明らかになるであろう。

【課題を解決するための手段】
【００１６】
本願において開示される発明のうち、代表的なものの概要を簡単に説明すれば、以下のとおりである。
【００１７】
本発明の代表的な実施の形態による仮想計算機システムは、物理計算機上で、前記物理計算機を複数の論理区画に分割し、前記各論理区画に前記物理計算機の計算機資源を割当てて制御する仮想計算機制御プログラムを実行することにより、前記各論理区画上でそれぞれゲストＯＳを含むゲストプログラムを動作させる仮想計算機システムであって、前記仮想計算機制御プログラムは、周期的に前記物理計算機のキャッシュメモリのエラーを回復するエラー回復処理を行うエラー回復モジュールと、前記キャッシュメモリで発生したエラーによる割込み通知に対して、前記キャッシュメモリのエラーを回復するエラー割込み処理を行うエラー割込みハンドラモジュールと、前記論理区画のシャットダウンもしくは再起動を契機に、前記キャッシュメモリのエラーを回復するエラーデータ初期化処理を

行うエラーデータ初期化モジュールとを有し、前記各論理区画上で動作する前記ゲストプログラムの動作とは独立して、前記キャッシュメモリのエラーの回復処理を行うことを特徴とするものである。

【発明の効果】

【００１８】

本願において開示される発明のうち、代表的なものによって得られる効果を簡単に説明すれば以下のとおりである。

【００１９】

本発明の代表的な実施の形態によれば、仮想計算機システムにおいて、キャッシュメモリエラーが発生した場合であっても長時間連続稼動を可能とする高信頼性システムを提供することができる。また、仮想計算機制御プログラム内にキャッシュメモリのエラー回復機能を備えることにより、ハードウェアの追加を必要とせず、かつ、ゲストプログラムにエラー回復手段を実装することを必要とせず、低コストで確実なキャッシュメモリエラー回復機能を提供することができる。

【発明を実施するための最良の形態】

【００２０】

以下、本発明の実施の形態を図面に基づいて詳細に説明する。なお、実施の形態を説明するための全図において、同一部には原則として同一の符号を付し、その繰り返しの説明は省略する。

【００２１】

＜キャッシュメモリ＞

キャッシュメモリは、高速なプロセッサ（ＣＰＵ）においてＣＰＵから主記憶のデータへのアクセス性能を向上させるために用いられる高速なメモリである。以下では、一般的なキャッシュメモリの動作の一例について説明する。

【００２２】

図２は、一般的なキャッシュメモリの動作の一例を説明するためのブロック図である。図示しないＣＰＵの命令制御部からデータロードの要求が発行されると、キャッシュメモリは、リクエストとロードアドレス２００を受信する。ロードアドレス２００は、タグアドレス、インデックスアドレス、オフセットアドレスの３つの部分に分類される。

【００２３】

キャッシュメモリは、まず、インデックスアドレスを参照してディレクトリアレイ２１０の対応するエントリを読み出す。ディレクトリアレイ２１０のエントリには、アドレスタグ２１１とともに有効ビット２１２が格納されている。ディレクトリアレイ２１０から読み出したエントリの有効ビット２１２がＯＮ（有効）の場合、キャッシュメモリは、ディレクトリアレイ２１０から読み出したエントリのアドレスタグ２１１（以下、登録タグと呼ぶ）と、ＣＰＵの命令制御部から受信したタグアドレス（以下、受信タグと呼ぶ）とを比較する。

【００２４】

受信タグがディレクトリアレイ２１０の該当エントリに登録されている登録タグと一致する場合、キャッシュメモリは、ＣＰＵの命令制御部から受信したインデックスアドレスとオフセットアドレスとを加算により連結して、データアレイ２２０のエントリアドレスを生成する。その後、データアレイ２２０から、エントリアドレスによって参照されるデータ（主記憶と同一のデータ）を読み出して、図示しないＣＰＵの演算部に渡す。

【００２５】

また、受信タグがディレクトリアレイ２１０に登録されている登録タグと不一致である場合、キャッシュメモリは、主記憶の対応するデータ（最新データ）のコピーを保持していないと判断し、図示しない主記憶の制御部へデータロード要求を送出する。主記憶の制御部から要求データが渡されると、キャッシュメモリは、ディレクトリアレイ２１０の該当エントリの有効ビット２１２をＯＮにするとともに、アドレスタグ２１１を書き込む。また、データアレイ２２０の対応するエントリに主記憶から渡されたデータを書き込むと

ともに、データをＣＰＵの演算部へ渡す。ディレクトリアレイ２１０から読み出したエントリの有効ビット２１２がＯＦＦ（無効）の場合も、主記憶の対応するデータ（最新データ）のコピーを保持していないと判断し、前記と同様の動作を行う。

【００２６】

なお、キャッシュメモリは、あらかじめ定められた大きさの連続アドレス領域を、データ管理単位としている。ディレクトリアレイ２１０の各エントリには、データ管理単位ごとの情報が登録される。以下では、便宜上、キャッシュメモリでのデータ管理単位をラインと記載する場合がある。一般的に、キャッシュメモリと主記憶との間のデータ転送は、ライン単位に行われる。

【００２７】

以上が、一般的なキャッシュメモリの動作の一例であるが、上記以外の動作により実現されるキャッシュメモリも一般に存在する。

【００２８】

＜実施の形態＞

以下、本発明の一実施の形態である仮想計算機システムについて説明する。図１は、本発明の一実施の形態である仮想計算機システムの例を示すブロック図である。

【００２９】

本実施の形態の仮想計算機システムでは、１つの物理計算機１００において、仮想計算機制御プログラム１４０が複数の論理区画１５０（１５０－１～１５０－ｎ）を構成する。この論理区画１５０では、それぞれ、ゲストＯＳ１５１（１５１－１～１５１－ｎ）およびゲストアプリケーション１５２（１５２－１～１５２－ｎ）が動作することが可能である（以下、ゲストＯＳ１５１、ゲストアプリケーション１５２を総称してゲストプログラムと記載する場合がある）。すなわち、物理計算機１００内において、仮想計算機制御プログラム１４０の制御により、複数の論理区画１５０においてゲストプログラムが動作可能である仮想計算機システムが構成される。

【００３０】

物理計算機１００は、ＣＰＵ１１０、主記憶１３０およびキャッシュメモリ１２０を有する。ＣＰＵ１１０は、物理計算機１００の種々の処理を実行する。また、主記憶１３０に格納されているプログラムを読み出して、そのプログラムに規定された処理を実行する。主記憶１３０は、後述する仮想計算機制御プログラム１４０等の各種プログラムやデータを格納する。

【００３１】

キャッシュメモリ１２０は、主記憶１３０のデータの写しを記憶するメモリであり、主記憶１３０から読み込まれたデータが格納されている。キャッシュメモリ１２０は、例えば、前述した図２に示すような、ディレクトリアレイ２１０とデータアレイ２２０とを備えて構成される。なお、キャッシュメモリ１２０の一般的な動作の例については、図２を用いて前述した通りである。

【００３２】

また、キャッシュメモリ１２０内のディレクトリアレイ２１０のデータの読み出し手段、および、有効ビット２１２をＯＦＦにする手段（無効化手段）がソフトウェアに対して提供されているものとする。これにより、ソフトウェアからディレクトリアレイ２１０内のデータを読み出したり、有効ビット２１２をＯＦＦ（無効）にしたりすることが可能である。

【００３３】

次に、仮想計算機制御プログラム１４０について説明する。仮想計算機制御プログラム１４０は、物理計算機１００のハードウェアリソース（計算機資源）を論理的に分割して論理区画１５０として管理するハイパバイザ（一般的に、ファームウェアと呼ばれる）である。すなわち、仮想計算機制御プログラム１４０は、物理計算機１００のＣＰＵ１１０や主記憶１３０等の計算機資源を論理的に分割して、これらを論理区画１５０として割当てる。

【００３４】
各論理区画１５０では、論理的に分割されたＣＰＵ１１０が仮想的なＣＰＵとして動作し、ＯＳ（ゲストＯＳ１５１）やプログラム（ゲストアプリケーション１５２）等を実行する。また、仮想計算機制御プログラム１４０は、これらＯＳやプログラムの制御を可能とするため、ゲストプログラムの動作を停止させたり再開させたりする手段を有する。なお、論理区画１５０の数の設定方法は様々だが、例えば、管理者等によって任意の数が設定されるといった方法がある。
【００３５】
この仮想計算機制御プログラム１４０は、エラー回復モジュール１４１とエラー割込みハンドラモジュール１４２とエラーデータ初期化モジュール１４４とを備える。また、これらのモジュールが使用するデータ群として、エラー回復制御情報データ部１４３を備える。
【００３６】
エラー回復モジュール１４１は、周期的にキャッシュメモリ１２０のエラーを回復するエラー回復処理を行うモジュールである。エラー割込みハンドラモジュール１４２は、キャッシュメモリ１２０で発生したエラーによる割込み通知に対して、キャッシュメモリ１２０のエラーを回復するエラー割込み処理を行ってハンドリングするモジュールである。エラーデータ初期化モジュール１４４は、論理区画１５０のシャットダウンもしくは再起動時に、キャッシュメモリ１２０のデータを初期化してエラーを回復するエラーデータ初期化処理を行うモジュールである。
【００３７】
図３は、エラー回復制御情報データ部１４３に保持する情報の例を示した図である。インデックスアドレス情報３０１は、エラー回復モジュール１４１等がキャッシュメモリ１２０のアクセスに用いるインデックスアドレスの値を保持する。このインデックスアドレス情報３０１は、仮想計算機制御プログラム１４０の初期化処理において、ディレクトリアレイ２１０の先頭エントリに対応するインデックスアドレスの値で初期化される。
【００３８】
エラー回復処理フラグ３０２は、エラー回復モジュール１４１が処理中であることを示すフラグである。このエラー回復処理フラグ３０２は、仮想計算機制御プログラム１４０の初期化処理においてＯＦＦで初期化される。エラーアドレス情報３０３は、エラー割込みハンドラモジュール１４２等がキャッシュメモリ１２０上のエラーアドレスを保持するために用いる。エラーアドレス有効フラグ３０４は、エラーアドレス情報３０３が有効か否かを示すフラグである。このエラーアドレス有効フラグ３０４は、仮想計算機制御プログラム１４０の初期化処理においてＯＦＦで初期化される。
【００３９】
図４は、エラー回復モジュール１４１が行うエラー回復処理の例を示すフローチャートである。エラー回復処理では、最初に、全てのゲストプログラムの動作を停止させる（Ｓ４０１）。これは、ゲストプログラムからのキャッシュメモリ１２０へのアクセスを停止し、その間にエラー回復モジュール１４１がエラー回復を行うことを可能にするための処理である。
【００４０】
次に、エラー回復処理フラグ３０２をＯＮにする（Ｓ４０２）。このフラグは、後述するエラー割込みハンドラモジュール１４２の処理において、対象のエラー割込みが、エラー回復モジュール１４１の処理中に発生したエラー割込みか否かを判定するために用いられる。次に、カウンタを０に初期化する（Ｓ４０３）。次に、インデックスアドレス情報３０１に保持されたインデックスアドレスを読み出し、キャッシュメモリ１２０において、読み出したインデックスアドレスに対応するディレクトリアレイ２１０のエントリから有効ビット２１２を読み出す（Ｓ４０４）。
【００４１】
次に、読み出した有効ビット２１２がＯＮ（有効）であるかどうかを判定する（Ｓ４０

５）。有効ビット２１２がＯＮの場合は、当該有効ビット２１２をＯＦＦ（無効）にして、ディレクトリアレイ２１０の該当エントリを無効化するようキャッシュメモリ１２０に指示する（Ｓ４０６）。なお、このような無効化を行った際のキャッシュメモリ１２０の一般的な動作として、無効化しようとするディレクトリアレイ２１０のエントリに対応するデータアレイ２２０のデータのみが最新のデータを保持した状態である場合は、当該最新データは次の階層のメモリ（図１の構成例の場合は主記憶１３０）へ転送され、最新データが失われないことが保証される。

【００４２】

次に、エラー回復モジュール１４１は、ステップＳ４０６で無効化したディレクトリアレイ２１０のエントリに対応するデータアレイ２２０のデータ部にデータをロードし直す（Ｓ４０７）。前述したように、キャッシュメモリ１２０と主記憶１３０との間のデータ転送はライン単位に行われるのが一般的である。従って、ここでのロードは一般的なロード命令を発行することで実現可能であり、このとき主記憶１３０のデータがキャッシュメモリ１２０に転送される。

【００４３】

なお、上述したステップＳ４０６およびステップＳ４０７の処理は、キャッシュメモリ１２０のデータアレイ２２０のデータがエラーデータを保持していた場合にエラー回復またはエラー検出を行う処理である。この具体的な内容については後述する。

【００４４】

ステップＳ４０５において、読み出した有効ビット２１２がＯＮではなかった場合は、ステップＳ４０６およびステップＳ４０７の処理は行わない。これは、ディレクトリアレイ２１０の対応するエントリが無効な状態であり、有効なデータが存在していないため、エラー回復が不要なためである。

【００４５】

次に、エラー回復モジュール１４１は、インデックスアドレス情報３０１の内容を更新する（Ｓ４０８）。更新後の値は、インデックスアドレス情報３０１のカレントの値に対し、対応するディレクトリアレイ２１０の１つ後方のエントリに対応するインデックスアドレスの値とする。ただし、カレントの値がディレクトリアレイ２１０の最後方（すなわち、最終）のエントリに対応するインデックスアドレスの値である場合は、ディレクトリアレイ２１０の先頭エントリに対応するインデックスアドレスの値にラップさせて更新後の値とする。この更新処理は、エラー回復モジュール１４１がキャッシュメモリ１２０のデータアレイ２２０のデータの先頭から最後までを周期的にエラー回復することを可能にする。

【００４６】

次に、エラー回復モジュール１４１は、カウンタを＋１インクリメントする（Ｓ４０９）。次に、カウンタの値が予め規定した回数よりも小さいか否かを判定する（Ｓ４１０）。カウンタが規定回数より小さい値であれば、ステップＳ４０４に戻り、ステップＳ４０４以降の処理を繰り返す。カウンタが規定回数より小さい値でない場合（すなわち、規定回数に達した場合）は、以降の処理に進む。

【００４７】

なお、前記規定回数の値は、仮想計算機システムにおけるシステム性能への影響を最小限にする値とする。この具体的な値は、本実施の形態の仮想計算機システムに要求される性能指標に依存するところがあるため、固定的な値は示さない。ただし、例えば、仮想計算機制御プログラム１４０に規定回数を設定する手段を設け、回数の変更を容易かつ柔軟に行えるものとする。

【００４８】

これ以降の処理はエンディング処理に相当するものである。まず、エラー回復モジュール１４１は、エラー回復処理フラグ３０２をＯＦＦにし（Ｓ４１１）、次に、ゲストプログラムの動作を再開させ（Ｓ４１２）、キャッシュメモリ１２０のエラー回復処理を終了する。

【００４９】
なお、上述の図４に示した、エラー回復モジュール１４１によるエラー回復処理は、本実施の形態では、仮想計算機システムにおけるキャッシュメモリ１２０を処理の対象としているが、仮想計算機の構成をとらない通常の物理計算機システムにおけるキャッシュメモリに対して適用することも可能である。
【００５０】
以下では、上述の図４に示した、エラー回復モジュール１４１によるキャッシュメモリ１２０のエラー回復処理の全体的な処理イメージについて説明する。
【００５１】
エラー回復モジュール１４１は、仮想計算機制御プログラム１４０の制御により、定期的に起動される。図５は、エラー回復モジュール１４１の処理の例の概要を示す図である。なお、ディレクトリアレイ２１０の有効ビット２１２（Ｖ）に関し、図５においては、例として、Ｖ＝１は有効な状態（ＯＮ）であることを示し、Ｖ＝０は無効な状態（ＯＦＦ）であることを示すものとする。
【００５２】
エラー回復モジュール１４１は、ディレクトリアレイ２１０の各エントリの有効ビット２１２を先頭エントリから順に読み出し、読み出した有効ビット２１２が示す状態に応じた処理を行う。図５において、処理５００－１は、ディレクトリアレイ２１０の先頭エントリに対する処理を示し、処理５００－２は、ディレクトリアレイ２１０の２番目のエントリに対する処理を示す。以下、処理５００－３は３番目のエントリ、処理５００－ｍは最終エントリに対する処理をそれぞれ示す。
【００５３】
なお、図５の各エントリの有効ビット２１２の値は、あくまで１つの例であり、図５の例では、先頭エントリ、３番目のエントリ、最終エントリがＶ＝１（有効）であり、２番目のエントリがＶ＝０（無効）である場合を示している。
【００５４】
エラー回復モジュール１４１は、ディレクトリアレイ２１０の１つのエントリの有効ビット２１２を読み出す（Ｓ４０４）。読み出した有効ビット２１２がＶ＝１（有効、ＯＮ）であれば、Ｖ＝０（無効、ＯＦＦ）の指示を行い（Ｓ４０６）、続いて、当該エントリに対応するデータアレイ２２０に１ライン分のデータをロードする（Ｓ４０７）。図５の例では、処理５００－１、処理５００－３、処理５００－ｍが該当する。読み出した有効ビット２１２がＶ＝０（無効、ＯＦＦ）であれば、当該エントリに対しては何も処理しない。図５の例では、処理５００－２が該当する。
【００５５】
エラー回復モジュール１４１は、ディレクトリアレイ２１０の最終エントリに対する処理（図５の例では処理５００－ｍ）を行った後、先頭エントリの有効ビット２１２を再び読み出し、読み出した有効ビット２１２が示す状態に応じた処理を行う。このように、エラー回復モジュール１４１は、ディレクトリアレイ２１０の先頭エントリから最終エントリまでの全てのエントリに対する処理を順次繰り返して行う。ただし、エラー回復モジュール１４１は、上記処理を無限に繰り返すことはなく、規定の回数を繰り返した時点で処理を終了し、仮想計算機制御プログラム１４０の起動元にリターンする。以上が、エラー回復モジュール１４１によるキャッシュメモリ１２０のエラー回復処理の概要である。
【００５６】
次に、前述した図４のエラー回復モジュール１４１の処理のフローチャートにおける、ステップＳ４０６およびステップＳ４０７の処理の詳細について図６～図９を用いて説明する。図４におけるステップＳ４０６およびステップＳ４０７の処理は、キャッシュメモリ１２０のデータアレイ２２０のデータがエラーデータを保持していた場合にエラー回復またはエラー検出を行う処理である。
【００５７】
図６は、キャッシュメモリ１２０のみ最新データを保持しており、キャッシュメモリ１

２０のデータがハードウェア訂正可能なビット反転データを含む場合の、エラー回復モジュール１４１によるエラー回復処理の一例を説明する図である。

【００５８】

なお、図６の例では、ハードウェア訂正可能なエラーが存在する場合の例であるため、データアレイ２２０にＥＣＣ部６０２が備わっており、データ部６０１に１ビット反転データ６１０が存在することを前提とするが、ＥＣＣ以外のハードウェア訂正可能エラーであってもよい。

【００５９】

図６（ａ）は、図４のフローチャートのステップＳ４０６の実行時の状態を示した図である。ステップＳ４０６では、データアレイ２２０内のエラーデータ（１ビット反転データ６１０を含む、データ部６０１で示されるデータ）に対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にする。この無効化により、データ部６０１の最新データが失われないことを保証するため、キャッシュメモリ１２０から主記憶１３０へデータ転送が行われる。これにより、対応する主記憶１３０の古いデータ（データ部６０３）が最新のデータに更新される。

【００６０】

この転送シーケンスにおいて、データアレイ２２０のデータ部６０１から取り出したデータをＥＣＣ部６０２の値を用いて訂正したデータが主記憶１３０へ転送される。従って、主記憶１３０のデータ部６０３には、ビット反転データが存在しない、エラー訂正されたデータが格納される。

【００６１】

しかし、この時点では、データアレイ２２０内のデータ部６０１には、１ビット反転データ６１０が残ったままである。この状態を長く放置しておくと、さらに別の１ビット反転データが発生した場合、２ビットエラーとなり、ハードウェア訂正不可能エラーとなってしまう。このようなハードウェア訂正不可能エラーを回避するために、図４のフローチャートのステップＳ４０７の処理が効果的である。

【００６２】

図６（ｂ）は、図４のフローチャートのステップＳ４０７の実行時の状態を示した図である。ステップＳ４０７では、ステップＳ４０６で無効化したディレクトリアレイ２１０のエントリに対応するデータアレイ２２０内のエラーデータ（１ビット反転データ６１０を含む、データ部６０１で示されるデータ）にデータをロードする。このロードにより、主記憶１３０のデータ（データ部６０３）がデータアレイ２２０へ転送され、データアレイ２２０のデータ部６０１の１ビット反転データ６１０が正しいデータ（回復データ６１１）に回復される。

【００６３】

図７は、キャッシュメモリ１２０および主記憶１３０ともに最新データを保持しており、キャッシュメモリ１２０のデータがハードウェア訂正可能なビット反転データを含む場合の、エラー回復モジュール１４１によるエラー回復処理の一例を説明する図である。

【００６４】

なお、図６の例と同様に、図７の例では、ハードウェア訂正可能なエラーが存在する場合の例であるため、データアレイ２２０にＥＣＣ部６０２が備わっており、１ビット反転データ６１０が存在することを前提とするが、ＥＣＣ以外のハードウェア訂正可能エラーであってもよい。

【００６５】

図７（ａ）は、図４のフローチャートのステップＳ４０６の実行時の状態を示した図である。ステップＳ４０６では、データアレイ２２０内のエラーデータ（１ビット反転データ６１０を含む、データ部６０１で示されるデータ）に対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にする。この無効化を行っても、対応する主記憶１３０のデータ（データ部６０３）が最新データであるため、キャッシュメモリ１２０から主記憶１３０へのデータ転送は行われない。従って、データアレイ２２０内

のエラーデータ（１ビット反転データ６１０を含む、データ部６０１で示されるデータ）は残ったままである。

【００６６】

図７（ｂ）は、図４のフローチャートのステップＳ４０７の実行時の状態を示した図である。ステップＳ４０７では、ステップＳ４０６で無効化したディレクトリアレイ２１０のエントリに対応するデータアレイ２２０内のエラーデータ（１ビット反転データ６１０を含む、データ部６０１で示されるデータ）にデータをロードする。このロードにより、主記憶１３０のデータ（データ部６０３）がデータアレイ２２０へ転送され、データアレイ２２０のデータ部６０１の１ビット反転データ６１０が正しいデータ（回復データ６１１）に回復される。

【００６７】

図８は、キャッシュメモリ１２０および主記憶１３０ともに最新データを保持しており、キャッシュメモリ１２０のデータがハードウェア訂正不可能なビット反転データを含む場合の、エラー回復モジュール１４１によるエラー回復処理の一例を説明する図である。

【００６８】

なお、図８の例では、ハードウェア訂正不可能なエラーが存在する場合の例であるため、データアレイ２２０にＥＣＣ部６０２が備わっており、２ビット反転データ８１０が存在することを前提とするが、ＥＣＣ以外のハードウェア訂正不可能エラーであってもよい。

【００６９】

図８（ａ）は、図４のフローチャートのステップＳ４０６の実行時の状態を示した図である。ステップＳ４０６では、データアレイ２２０内のエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）に対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にする。この無効化を行っても、対応する主記憶１３０のデータ（データ部６０３）が最新データであるため、キャッシュメモリ１２０から主記憶１３０へのデータ転送は行われない。従って、データアレイ２２０内のエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）は残ったままである。

【００７０】

図８（ｂ）は、図４のフローチャートのステップＳ４０７の実行時の状態を示した図である。ステップＳ４０７では、ステップＳ４０６で無効化したディレクトリアレイ２１０のエントリに対応するデータアレイ２２０内のエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）にデータをロードする。このロードにより、主記憶１３０のデータ（データ部６０３）がデータアレイ２２０へ転送され、データアレイ２２０のデータ部６０１の２ビット反転データ８１０が正しいデータ（回復データ８１１）に回復される。

【００７１】

図９は、キャッシュメモリ１２０のみ最新データを保持しており、キャッシュメモリ１２０のデータがハードウェア訂正不可能なビット反転データを含む場合の、エラー回復モジュール１４１によるエラー回復処理の一例を説明する図である。

【００７２】

なお、図８の例と同様に、図９の例では、ハードウェア訂正不可能なエラーが存在する場合の例であるため、データアレイ２２０にＥＣＣ部６０２が備わっており、２ビット反転データ８１０が存在することを前提とするが、ＥＣＣ以外のハードウェア訂正不可能エラーであってもよい。

【００７３】

図９（ａ）は、図４のフローチャートのステップＳ４０６の実行時の状態を示した図である。ステップＳ４０６では、データアレイ２２０内のエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）に対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にする。この無効化により、データ部

６０１の最新データが失われないことを保証するため、キャッシュメモリ１２０から主記憶１３０へデータ転送が行われる。これにより、対応する主記憶１３０の古いデータ（データ部６０３）が最新のデータに更新される。

【００７４】

この転送シーケンスにおいて、データアレイ２２０のデータ部６０１から取り出したデータをＥＣＣ部６０２の値を用いて訂正しようと試みるが、２ビット反転データ８１０が含まれるため訂正することは不可能である。このため、主記憶１３０には訂正されないままのエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）が転送される。従って、主記憶１３０のデータ部６０３には、ビット反転データが存在する、すなわち、エラー訂正されていないデータが格納される。

【００７５】

図９（ｂ）は、図４のフローチャートのステップＳ４０７の実行時の状態を示した図である。ステップＳ４０７では、ステップＳ４０６で無効化したディレクトリアレイ２１０のエントリに対応するデータアレイ２２０内のエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）にデータをロードしようと試みる。しかし、主記憶１３０のデータ（データ部６０３）に２ビット反転データ９１０が含まれるため、データ転送エラーとなる。

【００７６】

キャッシュメモリ１２０でのデータロードにおいて、このような訂正不可能なエラーに起因するデータ転送エラーが発生した場合は、一般的に、訂正不可能なエラーを示す割込みがソフトウェアに通知される。仮想計算機制御プログラム１４０は、仮想計算機システムにおける割込みハンドリング機能を備えており、訂正不可能なエラーを示す割込みに対して、エラー割込みハンドラモジュール１４２をコールする。

【００７７】

以下では、訂正不可能なエラーを示すエラー割込みを受けた場合の、エラー割込みハンドラモジュール１４２の処理について説明する。図１０は、エラー割込みハンドラモジュール１４２が行うエラー割込み処理の例を示すフローチャートである。

【００７８】

エラー割込みハンドラモジュール１４２の処理は、まず、エラー回復処理フラグ３０２がＯＮか否かを判定する（Ｓ１００１）。エラー回復処理フラグ３０２がＯＮでない場合は、エラー割込みハンドラモジュール１４２の処理を終了し、コール元である仮想計算機制御プログラム１４０の割込みハンドリング機能に戻る。エラー回復処理フラグ３０２がＯＮである場合は、キャッシュメモリ１２０におけるエラーアドレスを求める処理を行う（Ｓ１００２）。

【００７９】

図１１は、キャッシュメモリ１２０におけるエラーアドレスを求める方法の例を説明する図である。エラーアドレス１１０１のタグアドレスの値は、インデックスアドレス情報３０１に保持されたインデックスアドレスに対応する、ディレクトリアレイ２１０のエントリのアドレスタグ２１１の値と等しくなるはずである。従って、この値をエラーアドレス１１０１のタグアドレスの値とする。

【００８０】

また、エラーアドレス１１０１のインデックスアドレスの値は、インデックスアドレス情報３０１に保持されたインデックスアドレスの値と等しくなるはずである。従って、このインデックスアドレスをエラーアドレス１１０１のインデックスアドレスの値とする。エラーアドレス１１０１のオフセットアドレスの値は、ライン境界内の先頭のオフセットを指定するため、全て０とする。このように、専用のハードウェア等を必要とせず、仮想計算機制御プログラム１４０が、キャッシュメモリ１２０におけるエラーアドレス１１０１を求めることが可能である。

【００８１】

図１０のフローチャートにおいて、ステップＳ１００２でエラーアドレス１１０１を求

めた後は、求めたエラーアドレス１１０１がいずれかの論理区画１５０に割当てられたアドレス領域に含まれるアドレスか否かを判定する（Ｓ１００３）。エラーアドレス１１０１が論理区画１５０に含まれるアドレスである場合は、エラー割込みハンドラモジュール１４２は、エラーアドレス１１０１をエラーアドレス情報３０３に格納し（Ｓ１００４）、続いて、エラーアドレス有効フラグ３０４をＯＮにする（Ｓ１００５）。

【００８２】

次に、エラー割込みハンドラモジュール１４２は、エラーアドレス１１０１を含むアドレス領域が割当てられた論理区画１５０で稼動するゲストＯＳ１５１へエラー割込みを通知する（Ｓ１００６）。これにより、ゲストＯＳ１５１にエラー回復処理を委ねることになるが、当該ゲストＯＳ１５１がエラー回復を行わないＯＳである場合には、後述する、仮想計算機制御プログラム１４０のエラーデータ初期化モジュール１４４にてエラー回復を行うことが可能である。

【００８３】

ステップＳ１００３にて、エラーアドレス１１０１が論理区画１５０に含まれないアドレスである場合は、エラーアドレス１１０１が示すデータアレイ２２０にデータをストアする（Ｓ１００７）。このストアはエラー回復のためのストアであるため、ストアするデータの値は初期化データなどの任意の値でよい。次に、インデックスアドレス情報３０１に対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にするようキャッシュメモリ１２０に指示する（Ｓ１００８）。

【００８４】

上述したステップＳ１００７とステップＳ１００８の処理により、図９の例における、キャッシュメモリ１２０のデータアレイ２２０と主記憶１３０のハードウェア訂正不可能なエラーデータが回復される。この処理の詳細については後述する。

【００８５】

その後、エラーアドレス１１０１が論理区画１５０に含まれる場合も含まれない場合も、エラー回復処理フラグ３０２をＯＦＦにする（Ｓ１００９）。これにより、次にエラー割込みハンドラモジュール１４２がコールされた場合に、エラー回復モジュール１４１の処理中であるのか否かの判定を正しく行うための準備が整う。次に、エラー割込みハンドラモジュール１４２は、エラー回復モジュール１４１が停止させていたゲストプログラムの動作を再開させ（Ｓ１０１０）、エラー割込みハンドラモジュール１４２としての処理を終了する。以上が、エラー割込みハンドラモジュール１４２の処理の説明である。

【００８６】

次に、以下では、前述したエラーデータ初期化モジュール１４４にてエラー回復を行う処理について説明する。図１０のステップＳ１００６により、エラー割込みハンドラモジュール１４２が、エラーアドレス１１０１を含むアドレス領域が割当てられている論理区画１５０で稼動するゲストＯＳ１５１へエラー割込みを通知する。このとき、当該割込み通知を受けたゲストＯＳ１５１がエラー回復を行わないＯＳであった場合、エラーは残ったままとなる。多くのケースでは、このエラーが原因で、当該論理区画１５０で稼動するゲストプログラムは、当該論理区画１５０（論理計算機）のシャットダウンまたは再起動を選択することになる。

【００８７】

仮想計算機制御プログラム１４０は、前述したように、物理計算機１００のハードウェアリソース（計算機資源）を論理的に分割して論理区画１５０として管理するプログラムであり、物理計算機１００のＣＰＵ１１０や主記憶１３０等の計算機資源を論理的に分割して、これらを論理区画１５０として割当てている。

【００８８】

仮想計算機制御プログラム１４０は、論理区画１５０（論理計算機）のシャットダウンまたは再起動の際に、対象の論理区画１５０に分割して割当てていた主記憶１３０を開放する。この開放の後、仮想計算機制御プログラム１４０は、エラーデータ初期化モジュール１４４を実行する。なお、論理区画１５０（論理計算機）の再起動の場合に限り、仮想

計算機制御プログラム１４０は、エラーデータ初期化モジュール１４４の実行に続き、再起動処理にて主記憶１３０を分割して対象の論理区画１５０に割当て直す。

【００８９】

図１２は、エラーデータ初期化モジュール１４４が行うエラーデータ初期化処理の例を示すフローチャートである。エラーデータ初期化モジュール１４４は、まず、エラーアドレス有効フラグ３０４がＯＮであるか否かを判定する（Ｓ１２０１）。エラーアドレス有効フラグ３０４がＯＮでない場合は、エラーデータ初期化モジュール１４４の処理を終了する。

【００９０】

エラーアドレス有効フラグ３０４がＯＮである場合は、対象の論理区画１５０における開放されるアドレス領域が、エラーアドレス情報３０３のアドレスを含む領域か否かを判定する（Ｓ１２０２）。エラーアドレス情報３０３のアドレスを含まない領域である場合は、エラーデータ初期化モジュール１４４の処理を終了する。

【００９１】

エラーアドレス情報３０３のアドレスを含む領域である場合は、エラーアドレス情報３０３が示すデータアレイ２２０にデータをストアする（Ｓ１２０３）。このストアはエラー回復のためのストアであるため、ストアするデータの値は初期化データなどの任意の値でよい。次にエラーアドレス情報３０３のインデックスアドレスに対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にするようキャッシュメモリ１２０に指示する（Ｓ１２０４）。

【００９２】

上述したステップＳ１２０３とステップＳ１２０４の処理により、図９の例における、キャッシュメモリ１２０のデータアレイ２２０と主記憶１３０のハードウェア訂正不可能なエラーデータが回復される。この処理の詳細については後述する。その後、エラーアドレス有効フラグ３０４をＯＦＦにする（Ｓ１２０５）。これにより、次にエラーデータ初期化モジュール１４４が呼び出された場合に、エラーアドレス情報３０３の情報が有効であるか否かの判定を正しく行うための準備が整う。以上で、エラーデータ初期化モジュール１４４の処理は終了する。

【００９３】

次に、前述した図１０のフローチャートにおけるステップＳ１００７とステップＳ１００８の処理、もしくは前述した図１２のステップＳ１２０３とステップＳ１２０４の処理により、図９の例における、キャッシュメモリ１２０のデータアレイ２２０と主記憶１３０のハードウェア訂正不可能なエラーデータが回復される処理について図１３を用いて説明する。

【００９４】

図１３は、前述した図９における、キャッシュメモリ１２０および主記憶１３０の状態の例の続きであり、キャッシュメモリ１２０のみ最新データを保持しており、キャッシュメモリ１２０のデータがハードウェア訂正不可能なビット反転データを含む場合の、エラー回復モジュール１４１によるエラー回復処理の一例を説明する図である。

【００９５】

なお、図９の例と同様に、図１３の例では、ハードウェア訂正不可能なエラーが存在する場合の例であるため、データアレイ２２０にＥＣＣ部６０２が備わっており、２ビット反転データ８１０が存在することを前提とするが、ＥＣＣ以外のハードウェア訂正不可能エラーであってもよい。

【００９６】

図１３（ａ）は、図９（ｂ）と同じ図を再度示したものであり、図４のフローチャートのステップＳ４０７の実行時の状態を示した図である。

【００９７】

図１３（ｂ）は、図１０のフローチャートのステップＳ１００７、もしくは図１２のフローチャートのステップＳ１２０３の実行時の状態を示した図である。ステップＳ１００

７、もしくはステップＳ１２０３では、データストアが実行され、データアレイ２２０内のエラーデータ（２ビット反転データ８１０を含む、データ部６０１で示されるデータ）が初期化データ１３０１で上書きされる。また、このデータストアにより、初期化データ１３０１のＥＣＣが生成され、データアレイ２２０内のＥＣＣ部６０２に格納される。

【００９８】

これにより、データアレイ２２０内のデータ部６０１とＥＣＣ部６０２の値が、それぞれ初期化データ１３０１によってエラーのない値となり、訂正不可能エラーが取り除かれる。また、このデータストアにより、データアレイ２２０内のデータが最新データとなる。

【００９９】

図１３（ｃ）は、図１０のフローチャートのステップＳ１００８、もしくは図１２のフローチャートのステップＳ１２０４の実行時の状態を示した図である。ステップＳ１００８、もしくはステップＳ１２０４では、データアレイ２２０内の回復したデータ（回復初期化データ１３０２を含む、データ部６０１で示されるデータ）に対応するディレクトリアレイ２１０のエントリの有効ビット２１２をＯＦＦ（無効）にする。この無効化により、データ部６０１の最新データが失われないことを保証するため、キャッシュメモリ１２０から主記憶１３０へデータ転送が行われる。これにより、対応する主記憶１３０のデータが最新のデータに更新される。

【０１００】

この転送シーケンスにおいて、データアレイ２２０のビット反転データが存在しないデータが主記憶１３０へ転送されるため、主記憶１３０のデータ部６０３にはビット反転データが存在しないデータ（回復初期化データ１３０３を含むデータ）が格納され、主記憶１３０のデータ部６０３からも訂正不可能エラーが取り除かれる。

【０１０１】

以上に説明したように、本実施の形態の仮想計算機システムでは、図６～図８に示す例の場合は、キャッシュメモリエラーの影響が全く無い状態でエラーを回復し、システムの長時間連続稼動を可能にする。図９に示す例の場合は、ゲストＯＳ１５１によるエラー回復が期待できない場合は一部の論理区画１５０がダウンする可能性があるが、そのような場合でもエラー回復を行うため、一時的なダウンにとどめることが可能である。また、システム全体をダウンさせることはないため、図９に示す例のようなキャッシュメモリエラーが発生してもシステムダウンを防ぐことを可能にする。

【０１０２】

このように、本実施の形態の仮想計算機システムによれば、キャッシュメモリエラーが発生した場合であっても長時間連続稼動を可能とする高信頼性システムを提供することができる。また、仮想計算機制御プログラム１４０内にキャッシュメモリ１２０のエラー回復機能を備えることにより、ハードウェアの追加を必要とせず、かつ、ゲストプログラムにエラー回復手段を実装することを必要とせず、低コストで確実なキャッシュメモリエラー回復機能を提供することができる。

【０１０３】

以上、本発明者によってなされた発明を実施の形態に基づき具体的に説明したが、本発明は前記実施の形態に限定されるものではなく、その要旨を逸脱しない範囲で種々変更可能であることはいうまでもない。

【産業上の利用可能性】

【０１０４】

本発明は、１つの物理計算機において複数の論理区画を構成する仮想計算機制御プログラムを有する仮想計算機システムに利用可能である。

【図面の簡単な説明】

【０１０５】

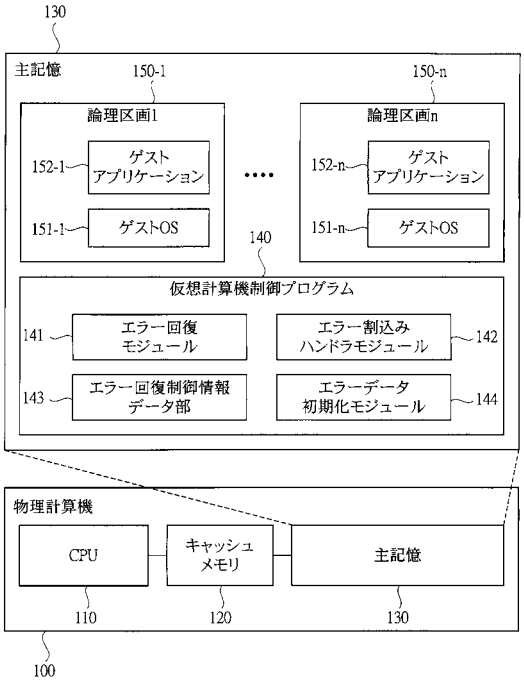
【図１】本発明の一実施の形態である仮想計算機システムの例を示すブロック図である。

【図２】一般的なキャッシュメモリの動作の一例を説明するためのブロック図である。

【図３】本発明の一実施の形態における、エラー回復制御情報データ部に保持する情報の例を示した図である。
【図４】本発明の一実施の形態における、エラー回復モジュールが行うエラー回復処理の例を示すフローチャートである。
【図５】本発明の一実施の形態における、エラー回復モジュールの処理の例の概要を示す図である。
【図６】本発明の一実施の形態における、エラー回復モジュールによるエラー回復処理の一例を説明する図である。
【図７】本発明の一実施の形態における、エラー回復モジュールによるエラー回復処理の一例を説明する図である。
【図８】本発明の一実施の形態における、エラー回復モジュールによるエラー回復処理の一例を説明する図である。
【図９】本発明の一実施の形態における、エラー回復モジュールによるエラー回復処理の一例を説明する図である。
【図１０】本発明の一実施の形態における、エラー割込みハンドラモジュールが行うエラー割込み処理の例を示すフローチャートである。
【図１１】本発明の一実施の形態における、キャッシュメモリにおけるエラーアドレスを求める方法の例を説明する図である。
【図１２】本発明の一実施の形態における、エラーデータ初期化モジュールが行うエラーデータ初期化処理の例を示すフローチャートである。
【図１３】本発明の一実施の形態における、エラー回復モジュールによるエラー回復処理の一例を説明する図である。
【符号の説明】
【０１０６】
１００…物理計算機、１１０…ＣＰＵ、１２０…キャッシュメモリ、１３０…主記憶、１４０…仮想計算機制御プログラム、１４１…エラー回復モジュール、１４２…エラー割込みハンドラモジュール、１４３…エラー回復制御情報データ部、１４４…エラーデータ初期化モジュール、１５０…論理区画、１５１…ゲストＯＳ、１５２…ゲストアプリケーション、
２００…ロードアドレス、２１０…ディレクトリアレイ、２１１…アドレスタグ、２１２…有効ビット、２２０…データアレイ、
３０１…インデックスアドレス情報、３０２…エラー回復処理フラグ、３０３…エラーアドレス情報、３０４…エラーアドレス有効フラグ、
５００…処理、
６０１、６０３…データ部、６０２…ＥＣＣ部、６１０…１ビット反転データ、６１１…回復データ、
８１０…２ビット反転データ、８１１…回復データ、
９１０…２ビット反転データ、
１１０１…エラーアドレス、
１３０１…初期化データ、１３０２、１３０３…回復初期化データ。

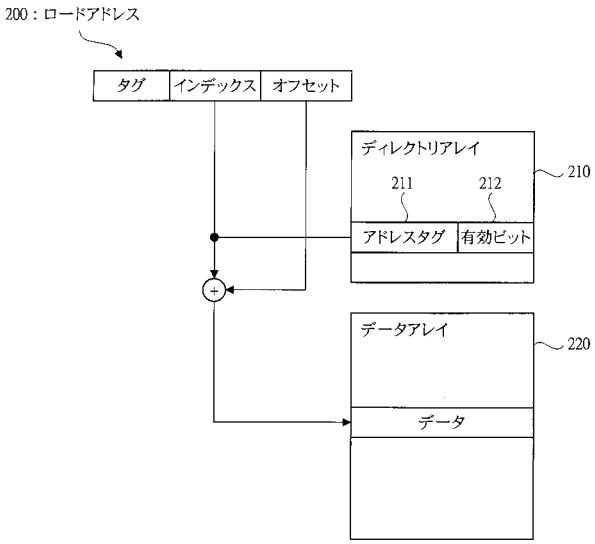
【図１】

図 1



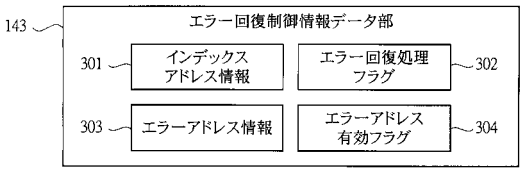
【図２】

図 2



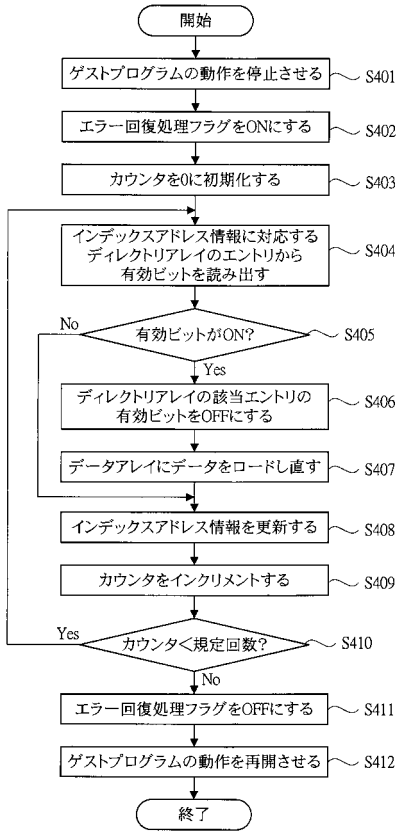
【図３】

図 3



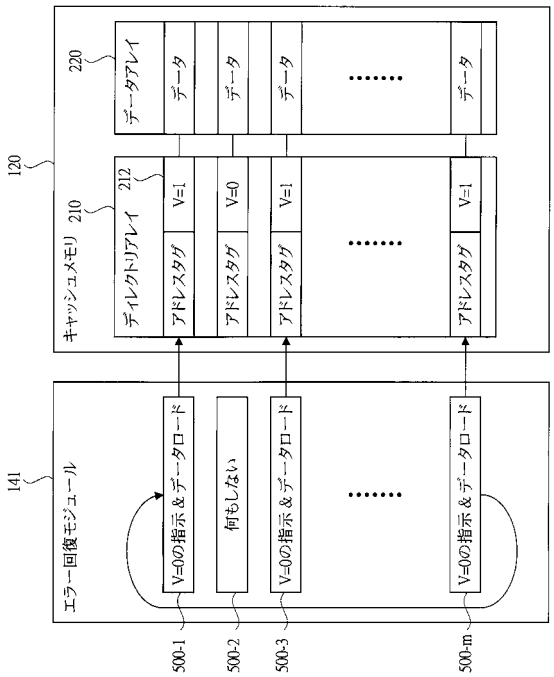
【図４】

図 4



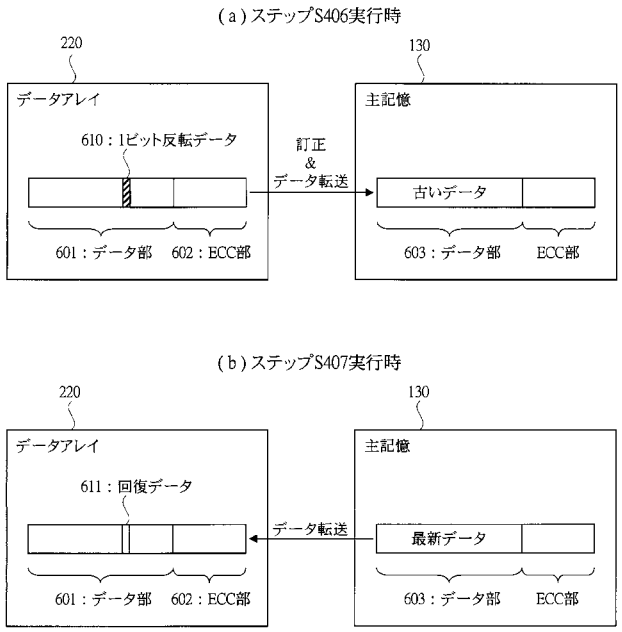
【図５】

図 5



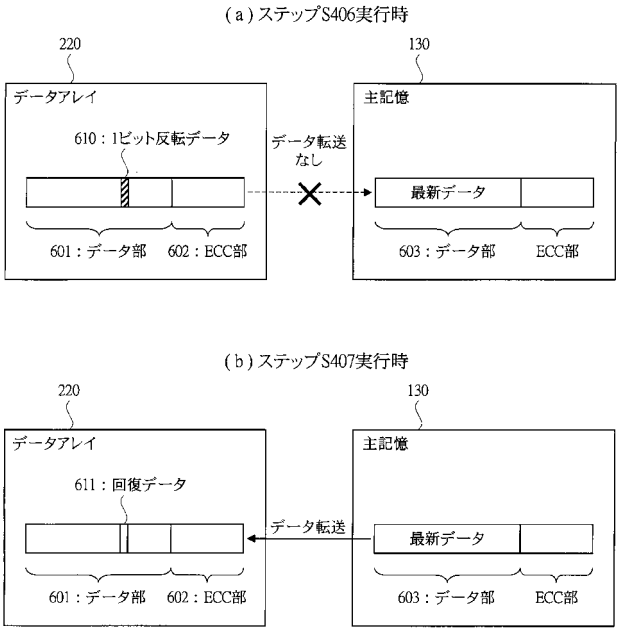
【図６】

図 6



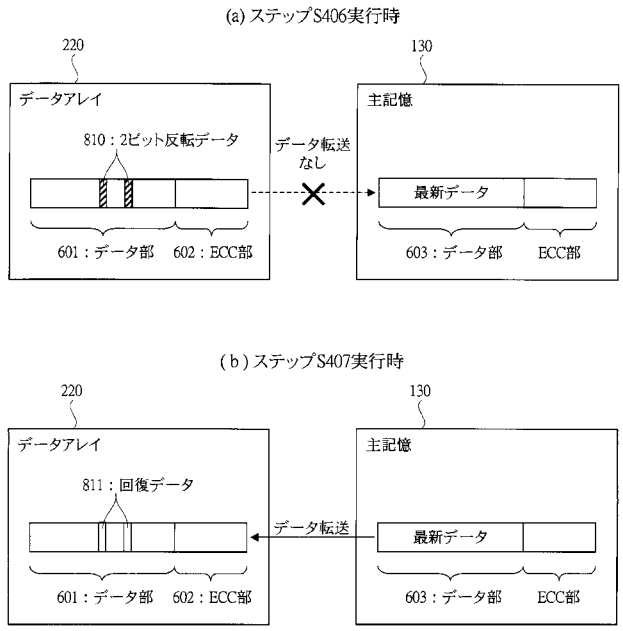
【図７】

図 7



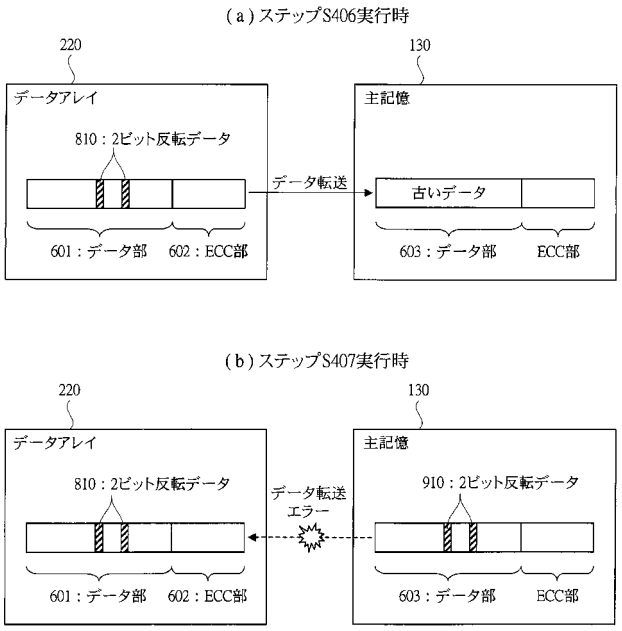
【図８】

図 8



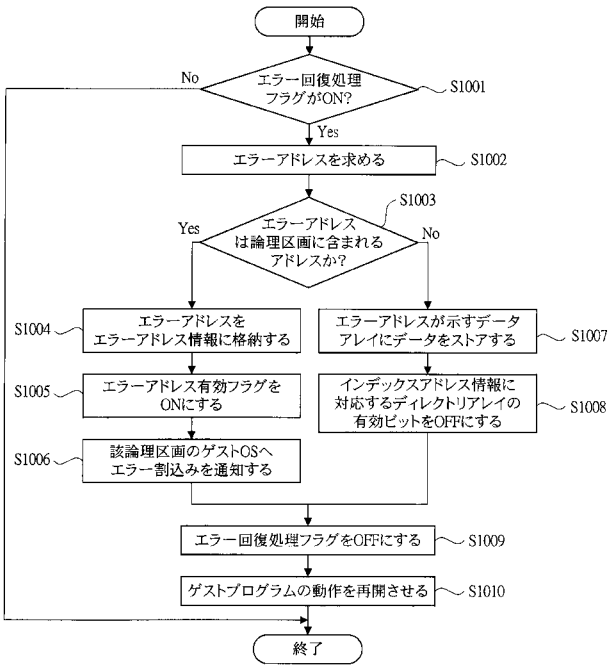
【図９】

図 9



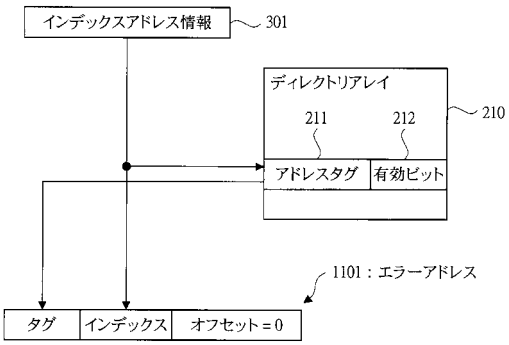
【図１０】

図 10



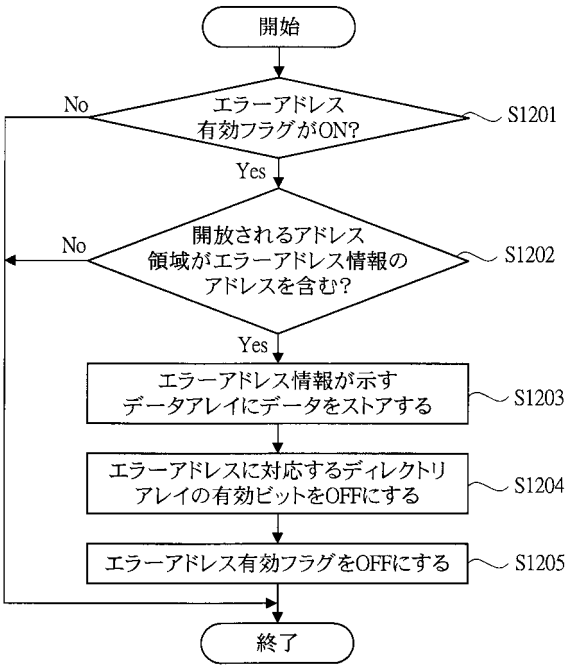
【図１１】

図 11



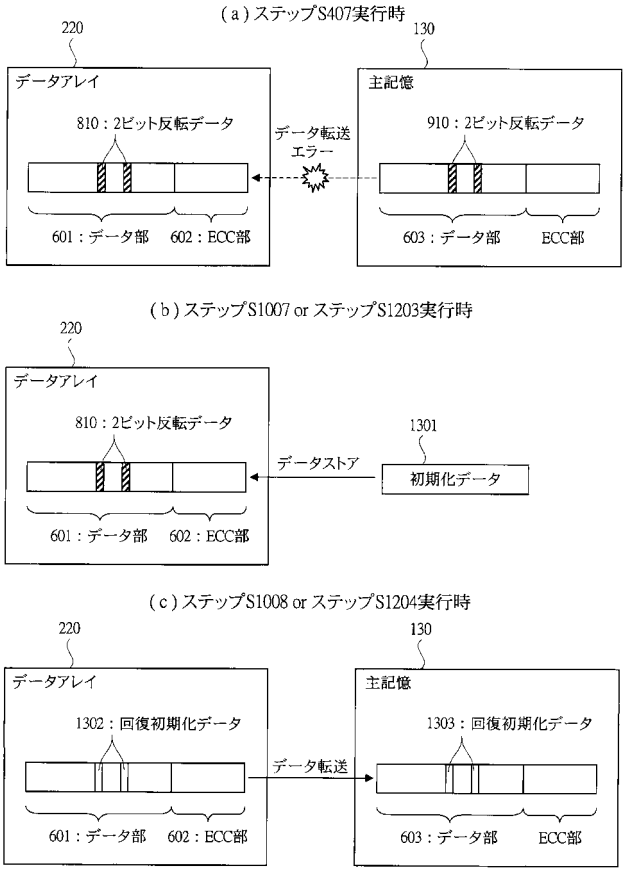
【図１２】

図 12



【図１３】

図 13



フロントページの続き

(51)Int.Cl.	FI	テーマコード（参考）
	G06F 12/16 320M	

(72)発明者 竹下 晃
神奈川県横浜市西区みなとみらい二丁目３番３号 日立情報通信エンジニアリング株式会社内
(72)発明者 山本 三雄
神奈川県秦野市堀山下１番地 株式会社日立製作所エンタープライズサーバ事業部内
(72)発明者 長島 広海
神奈川県横浜市西区みなとみらい二丁目３番３号 日立情報通信エンジニアリング株式会社内

Fターム(参考) 5B005 JJ01 MM01 MM36 RR01 VV13 WW02 WW17
5B018 GA02 HA14 KA22 MA03 QA20 RA11