

(19) 日本国特許庁 (JP)

(12) 特 許 公 報 (B2)

(11) 特許番号

特許第5035299号
(P5035299)

(45) 発行日 平成24年9月26日 (2012. 9. 26)

(24) 登録日 平成24年7月13日 (2012. 7. 13)

(51) Int. Cl. F I
G 0 6 F 11/20 (2006.01)
 G 0 6 F 11/20 3 1 O K
 G 0 6 F 11/20 3 1 O F

請求項の数 6 (全 23 頁)

(21) 出願番号	特願2009-141797 (P2009-141797)	(73) 特許権者	000005108
(22) 出願日	平成21年6月15日 (2009. 6. 15)		株式会社日立製作所
(62) 分割の表示	特願2006-340673 (P2006-340673) の分割		東京都千代田区丸の内一丁目6番6号
原出願日	平成18年12月19日 (2006. 12. 19)	(74) 代理人	100100310 弁理士 井上 学
(65) 公開番号	特開2009-205696 (P2009-205696A)	(72) 発明者	高本良史
(43) 公開日	平成21年9月10日 (2009. 9. 10)		神奈川県川崎市麻生区王禅寺1099番地
審査請求日	平成21年6月15日 (2009. 6. 15)		株式会社日立製作所システム開発研究所 内
		(72) 発明者	堀江亨
			神奈川県横浜市戸塚区戸塚町5030番地
			株式会社日立製作所ソフトウェア事業部 内
		審査官	三坂 敏夫

最終頁に続く

(54) 【発明の名称】 マルチコアプロセッサ制御方法、その計算機システム、及び管理計算機のプログラム

(57) 【特許請求の範囲】

【請求項 1】

複数の演算コアを有する複数のプロセッサを有するサーバであって、前記サーバでは仮想サーバを生成し制御する仮想化機構が稼働し、前記仮想化機構は管理サーバによって管理される仮想化環境において、

前記仮想化機構は前記プロセッサの障害予兆を検出する障害予兆検出機構を有し、

前記障害予兆検出機構がプロセッサあるいは前記演算コアの障害予兆を検出した場合に、

前記管理サーバは前記仮想化機構が前記障害予兆を検出したプロセッサあるいは前記演算コアで実行されないように実行スケジュールを変更し、

前記管理サーバは、障害予兆を検出した前記プロセッサの部位ごとに、当該プロセッサを交代するか、あるいは当該演算コアを交代するかを保持する対応テーブルを有し、

障害予兆を検出した場合に、前記対応テーブルに基づき前記プロセッサあるいは前記演算コアを交代し、

前記プロセッサが異なる種類の前記演算コアを有する場合に、前記障害予兆検出時に種類が少ない前記演算コアを優先して交代することを特徴とするマルチコアプロセッサ制御方法。

【請求項 2】

前記障害予兆検出時の対応テーブルには、前記障害予兆検出時の交代内容に加え、障害を検出した前記プロセッサの部位ごとに、当該プロセッサを交代するか、あるいは当該演算

10

20

コアを交代するかについても保持するものであり、前記プロセッサの部位には、汎用コアもしくは専用コアを含むものであることを特徴とした請求項1のマルチコアプロセッサ制御方法。

【請求項3】

複数の演算コアを有する複数のプロセッサを有する計算機を備えた計算機システムであって、

前記計算機システムでは、仮想サーバを生成し制御する仮想化機構が稼働し、前記仮想化機構は管理サーバによって管理され、

前記仮想化機構は、前記プロセッサの障害予兆を検出する障害予兆検出機構を有し、

前記障害予兆検出機構は、プロセッサあるいは前記演算コアの障害予兆を検出し、
前記管理サーバは、

前記仮想化機構が前記障害予兆を検出したプロセッサあるいは前記演算コアで実行されないように実行スケジュールを変更する手段と、

障害予兆を検出した前記プロセッサの部位ごとに、当該プロセッサを交代するか、あるいは当該演算コアを交代するかを保持する対応テーブルと、

前記障害予兆を検出した場合に、前記対応テーブルに基づき前記プロセッサあるいは前記演算コアを交代するとともに、前記プロセッサが異なる種類の前記演算コアを有する場合に、前記障害予兆検出時に種類が少ない前記演算コアを優先して交代する手段を有することを特徴とするマルチコアプロセッサを有する計算機システム。

【請求項4】

前記障害予兆検出時の対応テーブルには、前記障害予兆検出時の交代内容に加え、障害を検出した前記プロセッサの部位ごとに、当該プロセッサを交代するか、あるいは当該演算コアを交代するかについても保持するものであり、前記プロセッサの部位には、汎用コアもしくは専用コアを含むものであることを特徴とした請求項3のマルチコアプロセッサを有する計算機システム。

【請求項5】

複数の演算コアを有する複数のプロセッサを有する計算機は、仮想サーバを生成し制御する仮想化機構が稼働し、前記仮想化機構は管理計算機によって管理され、

前記仮想化機構は前記プロセッサの障害予兆を検出する障害予兆検出機構を有し、

前記管理計算機は、障害予兆を検出した前記プロセッサの部位ごとに、当該プロセッサを交代するか、あるいは当該演算コアを交代するかを保持する対応テーブルを有し、

前記障害予兆検出機構がプロセッサあるいは前記演算コアの障害予兆を検出した場合に、

前記管理計算機に、

前記仮想化機構が前記障害予兆を検出したプロセッサあるいは前記演算コアで実行されないように実行スケジュールを変更する機能と、

前記障害予兆を検出した場合に、前記対応テーブルに基づき前記プロセッサあるいは前記演算コアを交代するとともに、前記プロセッサが異なる種類の前記演算コアを有する場合に、前記障害予兆検出時に種類が少ない前記演算コアを優先して交代する機能とを実行させる

ことを特徴とする管理計算機のプログラム。

【請求項6】

前記障害予兆検出時の対応テーブルには、前記障害予兆検出時の交代内容に加え、障害を検出した前記プロセッサの部位ごとに、当該プロセッサを交代するか、あるいは当該演算コアを交代するかについても保持するものであり、前記プロセッサの部位には、汎用コアもしくは専用コアを含むものであることを特徴とした請求項5の管理計算機のプログラム

。

【発明の詳細な説明】

【技術分野】

【0001】

10

20

30

40

50

本発明は、マルチコアプロセッサで稼働する計算機の制御方法に関する。

【背景技術】

【0002】

企業の計算機システムやデータセンタにおいて、サーバの保有台数が増大した結果、運用管理コストも増大している。この問題を解決する技術としてプロセッサの高性能化がある。従来から、プロセッサの動作周波数の向上によりプロセッサの性能は向上してきた。プロセッサの性能向上に対するユーザの期待にはいくつかの異なる方向がある。一つは、より早く処理を行うための高性能化である。より複雑な処理を短時間で処理することで高度な処理結果を短い時間で得ることができるようになる。もう一つは、運用管理の容易化である。処理性能が高いプロセッサをサーバに用いることで、単一のサーバ上でより多くの処理を行うことができるため、管理すべきサーバの台数を少なくすることができる。その結果、運用管理対象となるサーバ数が少なくなり、運用管理が容易になる。現在は、特に運用管理を容易するためのプロセッサ高性能化に対する期待が大きくなっている。こういった背景の中、マルチコアプロセッサがプロセッサアーキテクチャの中心に位置づけられつつある。マルチコアプロセッサとは、単一の物理プロセッサ内に、並列に演算を行うことができるコアを複数内蔵するアーキテクチャである。従来のように、単一のコアの動作周波数を向上させる方法ではなく、コアを多数有することで性能を向上させる方法である。マルチコアプロセッサのメリットはいくつかあるが、多数の複雑な論理を必要とする従来のプロセッサに比べ、消費電力や全体性能を向上させることができる。こういったマルチコアプロセッサによって、サーバ単体の性能は従来よりも向上し、その結果運用管理の容易化を加速する技術として期待されている。

10

20

【先行技術文献】

【特許文献】

【0003】

【特許文献1】特開2006-260568

【特許文献2】特開2004-30363

【特許文献3】特開2003-330740

【発明の概要】

【発明が解決しようとする課題】

【0004】

マルチコアプロセッサにより性能向上と共に運用管理は容易化されることが期待できる。しかし、一方で可用性が低下する問題がある。特定の部位に障害が発生した場合でもシステムの処理を継続できる事が高可用性の条件となる。マルチコアプロセッサは、多数のコアを単一のプロセッサに内蔵することで性能を向上させるが、可用性の観点で見ると単一プロセッサの障害によって影響を受ける範囲が大きくなる事を意味している。つまり、マルチコアプロセッサによって、少ないプロセッサでシステムを構築することができる代わりに、可用性が低下する。例えば、2GHzのコアを1つ搭載したシングルコアプロセッサを使用して、合計20GHzのシステムを構築する場合は20個のプロセッサを要する。20個のプロセッサ上でシステムを構築した場合、単一プロセッサの障害によって影響を受ける範囲は1/20である。一方、2GHzのコアを単一のプロセッサに4つ搭載したマルチコアプロセッサは、同性能を得るのに5つのプロセッサで構築する事ができる。しかし、この場合、単一プロセッサの障害によって影響を受ける範囲は最大で4/20となり、マルチコアプロセッサの方が障害時の影響が大きくなる課題がある。

30

40

【0005】

また、別の課題としてマルチコアプロセッサを搭載したサーバの仮想化における可用性の低下がある。運用管理を容易化するもう一つの方法として、サーバの仮想化技術がある。サーバの仮想化技術は、単一の物理サーバ上で複数の仮想サーバを稼働することができる技術である。物理サーバには、プロセッサやメモリといったリソースがあり、サーバの仮想化技術はリソースを分割し、それぞれを異なる仮想的なサーバに割り当てることで単一の物理サーバ上で複数の仮想サーバを同時に実行する。マルチコアプロセッサ等による

50

性能向上とメモリなどのリソースが低コスト化したことで、サーバの仮想化技術に関するニーズが増大している。また、サーバの仮想化技術は、複数の仮想サーバを単一の物理サーバ上で稼働させることによる統合のメリット以外にも、ワークロード制御により一層のサーバリソースの有効利用を図るメリットがある。ここで、ワークロード制御とは、物理サーバのCPUやメモリといったリソースを状況に応じて仮想サーバに増減させる処理である。例えば、ある仮想サーバの負荷が高くなった場合に、同一の物理サーバ上で稼働する他の仮想サーバの中で負荷が低い仮想サーバからリソースを融通することで、物理サーバのリソースを有効に使用することができるようになる。

【0006】

マルチコアプロセッサは、サーバの仮想化環境においては、多くのプロセッサリソースを保有できることで、単一のプロセッサから多数の仮想サーバを生成する能力を有することができる。つまり、多数のサーバをより少ないサーバに統合することができるようになる。このようにマルチコアプロセッサと仮想化技術の相乗効果によって、運用管理の容易化が進むことが期待できる。実際、仮想化技術と複数のプロセッサを組み合わせることで、システムの信頼性を向上させることもできる。例えば、物理サーバに生成された複数の仮想サーバを用い、正副のホットスタンバイ環境を構築することで、従来は複数の物理サーバを要した高信頼環境を、単一の物理サーバで構築するなどの技術がある。従来技術では、複数の仮想サーバ間でホットスタンバイを構築する技術や、障害の予兆を検出することで新たな仮想サーバを構築し障害に備えるなどの技術はある。しかし、これらの技術は、マルチコアプロセッサを対象にした技術ではなく、シングルコアプロセッサ環境における仮想サーバを用いた高信頼化技術である。また、障害の影響範囲の大きさを考慮しておらず、マルチコアプロセッサに対応した技術ではない。従って、マルチコアプロセッサ環境における仮想サーバの信頼性や可用性を向上することが課題である。

【課題を解決するための手段】

【0007】

複数のマルチコアプロセッサからなるシステムにおいて、プロセッサとプロセッサが有するコアを管理するテーブルを有し、仮想サーバの生成時において異なるプロセッサのコアを用いて単一の仮想サーバを構成する。

【0008】

また、プロセッサが有するコアの数に応じて、プロセッサの数を可変にする。また、プロセッサの障害予兆を検出した場合に、前記障害の予兆を検出したプロセッサには仮想化機構の実行スケジュールを渡さないように制御する。

【0009】

また、複数のマルチコアプロセッサからなるシステムにおいて、オペレーティングシステムのプロセス又はスレッドの処理に、複数の異なるプロセッサが有する演算コアを割り当てる。

【0010】

また、複数のマルチコアプロセッサからなるシステムにおいて、オペレーティングシステムに、複数の異なるプロセッサが有する演算コアを割り当てる。

【発明の効果】

【0011】

マルチコアプロセッサ環境において、可用性を向上することができるようになる。

【図面の簡単な説明】

【0012】

【図1】本発明の全体構成図を示す。

【図2】プロセッサの構成図を示す。

【図3】仮想化機構の構成図を示す。

【図4】仮想サーバへのプロセッサ割り当て方法を示す。

【図5】サーバ構成テーブルを示す。

【図6】仮想化機構管理テーブルを示す。

【図 7】プロセッサ管理テーブルを示す。

【図 8】障害対応テーブルを示す。

【図 9】ユーザインターフェースを示す。

【図 10】コア割り当てプログラムのフローチャートを示す。

【図 11】信頼性重視型コア割り当てのフローチャートを示す。

【図 12】性能重視型コア割り当てのフローチャートを示す。

【図 13】一般型コア割り当てのフローチャートを示す。

【図 14】コア割り当て例を示す。

【図 15】障害予兆検出機構のフローチャートを示す。

【図 16】障害管理プログラムのフローチャートを示す。

10

【図 17】障害対策プログラムのフローチャートを示す。

【図 18】障害予兆対策プログラムのフローチャートを示す。

【図 19】スケジュール制御機構のフローチャートを示す。

【図 20】割り込み制御テーブルを示す。

【図 21】マイグレーション制御プログラムのフローチャートを示す。

【図 22】プロセッサ制御機構のフローチャートを示す。

【図 23】実施例 2 における異なるコアを有するプロセッサの構成を示す。

【図 24】実施例 2 における障害対応テーブルを示す。

【図 25】実施例 2 における障害対応プログラムのフローチャートを示す。

【図 26】実施例 2 における専用コア交代プログラムのフローチャートを示す。

20

【図 27】実施例 2 におけるアプリケーションプログラムのリカバリ構成を示す。

【図 28】仮想化機構の連携時のサーバ構成テーブル。

【図 29】管理サーバの構成を示す。

【発明を実施するための形態】

【0013】

以下、図面を用いて本願発明の実施の形態を説明する。

【実施例 1】

【0014】

図 1 は、本発明における実施例の全体図を示している。本実施例における制御の中心は、管理サーバ 101 である。仮想化機構管理プログラム 102 は、管理対象となるサーバ装置 112 で稼働する仮想化機構 115 を管理するプログラムである。仮想化機構管理プログラム 102 は、コア割り当てプログラム 103 と障害管理プログラム 104 とスケジュール制御プログラム 105 とマイグレーション制御プログラム 106 から構成される。コア割り当てプログラム 103 は、仮想サーバ 114 に対しサーバ装置 112 が有するプロセッサ 119 やプロセッサ 119 内のコア 116 を割り当てる制御を行う。障害管理プログラム 104 は、プロセッサ 119 やコア 116 に障害や障害の予兆を検出した際に実行され、プロセッサ 119 やコア 116 の交代などの管理を行う。スケジュール制御プログラム 105 は、仮想化機構 115 に対して、プロセッサ 119 やコア 116 の実行スケジュールを変更する指示を発行する。マイグレーション制御プログラム 106 は、プロセッサ 119 やコア 116 の交代を仮想化機構 115 に対して指示するプログラムである。これらは、コア管理テーブル 107、サーバ構成テーブル 108、仮想化機構管理テーブル 109、プロセッサ管理テーブル 110、障害対応テーブル 111 を更新・参照しながら実行される。管理対象であるサーバ装置 112 は、メモリ 113、プロセッサ 119、プロセッサ 119 に内蔵されるコア 116、プロセッサコントローラ 117、I/O デバイス 118 から構成される。メモリ 113 内には、仮想化機構 115 が格納および実行され、これにより仮想サーバ 114 が生成される。仮想サーバ 114 は、仮想化機構 115 によってサーバ装置 112 内のメモリ 113 やプロセッサ 119 や I/O デバイス 118 等のリソースを分配することで、サーバと同等の機能を有することができる。

30

40

【0015】

なお、図 29 は、管理サーバ 101 のハードウェアとプログラムの関係を記載しており

50

、管理サーバ１０１は、プログラムやテーブルなどのデータを格納するメモリ２９０１、プログラムを実行するプロセッサ２９０２、ディスクアクセスのインターフェースであるHBA（Host Bus Adaptor）、ネットワークインターフェース２９０４から構成され、本発明における仮想化機構管理プログラム１０２や各テーブルはメモリ内に格納されている。仮想化機構管理プログラム１０２はプロセッサ２９０２で実行することにより実現するが、説明を簡単にするため、以下、プロセッサ２９０２の処理に関してはプログラムを実行主体として説明する。また、コア割当てプログラム１０３、スケジュール制御プログラム１０５、障害管理プログラム１０４、マイグレーション制御プログラム１０６は、プロセッサ２９０２が実行することで、仮想化機構管理処理を実現するが、これらはコア割当て部、スケジュール制御部、障害管理部、マイグレーション制御部など、各処理を行う処理部として集積回路化するなどしてハードウェアで実現することもできる。

10

【００１６】

図２は、プロセッサ１１９の構成例を示している。プロセッサ１１９内には複数のコア１１６が内蔵されている。コア１１６は、メモリ１１３から命令を読み込み、命令の実行やメモリ１１３とデータ読み込みや書き込みを行うことができる。コア１１６内の１次キャッシュ２０２は、命令やデータを高速に読み書きするための高速なメモリである。複数のコア１１６は、１次キャッシュ２０１よりも容量の大きな２次キャッシュ２０３を有する、２次キャッシュ２０３は複数のコア１１６から共有される。また、外部のメモリ１１３とのデータや命令を入出力するためのメモリインターフェースを有する。複数のコア１１６は、それぞれ独立に動作することができる。従って、コア１１６の数が多いほど処理性能が向上する。また、複数のコア１１６は、マルチプロセッサの形態をとることができる。つまり、コア１１６は独立に動作するが、メモリ１１３上のプログラムからは単一システムのような使い勝手を得ることができる。例えば、メモリ１１３上に複数のプログラムが同時に稼働する場合、一般的にはオペレーティングシステムと呼ばれるシステムプログラムが自動的に複数のコア１１６へ実行の制御を割当てるため、プログラムからはあたかも複数のプログラムが同時に実行されたように見ることができる。

20

【００１７】

図３は仮想化機構１１５の詳細を示している。サーバ装置１１２内のメモリ１１３内には仮想化機構１１５が稼働する。仮想化機構１１５内には、スケジュール制御機構３０１とプロセッサ制御機構３０３と障害予兆検出機構３０２と割り込み制御テーブル３０４と制御インターフェース（I/F）３０５から構成される。スケジュール制御機構３０１は、仮想化機構１１５に関連するプログラムの実行スケジュール制御を行う。プロセッサ制御機構３０３はプロセッサ１１９の動作モードの制御を行う。障害予兆検出機構３０２は、定期的あるいは、プロセッサ１１９が空き状態になると動作し、ハードウェアのチェックを行う機構である。これらの動作は、外部から制御I/F ３０５を介して制御することができる。

30

【００１８】

図４は、仮想化機構１１５によって、プロセッサ１１９が仮想サーバ１１４に割り当てられている状態を示している。仮想サーバ１１４への割り当て方にはいくつかの方法がある。４０１の割り当て方は、単一のプロセッサ１１９を、複数の仮想サーバ１１４に割り当てる方法である。共有割当て方法とも呼ばれ、通常は複数の仮想サーバ１１４で同一のプロセッサ１１９リソースを共有し、プロセッサ１１９よりも多い仮想サーバ１１４を構築する際に使用される割当て方法である。４０２と４０３の割り当て方は、占有割当て方法とも呼ばれ、仮想サーバ１１４は専用のプロセッサ１１９が割り当てられ他の仮想サーバ１１４から共有されることはない。割り当て方には、いくつかのパターンがあり、４０２のように複数のプロセッサ１１９を単一の仮想サーバ１１４に割り当てる。高性能な仮想サーバ１１４を要求される場合にはこういった割り当て方を行う。このように、仮想化機構１１５は、プロセッサ１１９などのサーバ装置１１２のリソースを複数の仮想サーバ１１４に分配する事で、複数の仮想サーバ１１４を生成し、かつ複数の仮想サーバ１１４は

40

50

並列に動作することもできる。また、ここではプロセッサ 119 の割当て方について述べたが、I/O デバイス 118 やメモリ 113 などのリソースも同様に分割して割り当てることができる。

【0019】

図 5 はサーバ構成テーブル 108 を示している。サーバ構成テーブル 108 は、主にサーバ装置 112 のハードウェアの構成を管理するために設けられたテーブルである。コラム 501 は、システム識別子 (ID) を示している。システム ID 501 とは、例えば単一のサーバ構成を単位とした識別子である。一例として、マルチプロセッサ構成などが考えられる。単一のシステム ID 501 には、コラム 502 に示すように単一あるいは複数のプロセッサから構成される。さらに、プロセッサには、コラム 503 に示すように複数のコアが搭載される。コラム 503 では、プロセッサ内に有するコアの数が記載される。コラム 504 では、システムが有するメモリの総容量が格納される。コラム 505 では、システムが有する I/O デバイスの種類と数が格納される。例えば、I/O デバイスとは、ネットワークと接続するためのネットワークアダプタや、ディスクと接続するためのディスクアダプタなどである。コラム 506 には、仮想化機構の識別子 (ID) が格納される。これらのコラムにより、システム当たりのハードウェアリソースを把握することができ、また仮想化機構と対応づけることで仮想化機構が使用することができるハードウェアリソースを管理することができるようになる。

【0020】

図 6 は、仮想化機構管理テーブル 109 の構成を示している。コラム 601 は、仮想化機構 ID を示す。これは、図 5 のコラム 506 に対応する仮想化機構 ID である。コラム 602 は、生成された仮想サーバ ID が格納される。仮想化機構によって生成された全ての仮想サーバ ID が格納される。また、生成された仮想サーバだけではなく、仮想化機構が有する空きハードウェアリソースであるリソースプールに関する情報も記載される。これは、新たに仮想サーバを生成する場合や既存の仮想サーバにリソースを追加する場合に、システムが有する空きリソースから割り当てるため、リソースの残りを管理する必要があるためである。コラム 603 は、仮想サーバに割り当てられているプロセッサ ID が格納される。複数のプロセッサが割り当てられている場合には複数のプロセッサ ID が記載される。コラム 604 は、コラム 603 に記載されたプロセッサが有するコア ID が格納される。これにより、仮想サーバへの割当てを、コア単位で管理することができるようになる。コラム 605 は、仮想サーバに割り当てられたメモリ容量が格納される。コラム 606 には、仮想サーバに割り当てられた I/O デバイスが格納される。仮想化機構管理テーブル 109 は、新たな仮想サーバの生成や、削除、あるいは割り当てリソースに変更が生じた場合に更新される。

【0021】

図 7 はプロセッサ管理テーブル 110 の構成を示している。プロセッサ管理テーブル 110 は主に、プロセッサの障害に関する情報が格納される。コラム 701 は、プロセッサ ID が格納される。コラム 702 には、プロセッサが有するコア ID が格納される。コラム 703 には各プロセッサやコアの状態が格納される。例えば、障害の予兆を検出した場合や障害が発生した場合には部位が格納される。プロセッサ管理テーブル 110 によって、複数のプロセッサや複数のコアを有するサーバ環境において、障害の予兆検出時や障害時にプロセッサ内の部位を特定することができる。

【0022】

図 8 は障害対応テーブル 111 の構成を示している。障害対応テーブル 111 は、プロセッサに障害が発生した場合や障害の予兆を検出した場合の対策が記載される。対策はプロセッサの種類 801 毎に記載される。プロセッサ種類 801 によって、該プロセッサのアーキテクチャを特定でき、またアーキテクチャの違いから生じる障害時の対策を管理することができる。コラム 802 は、プロセッサ部位が格納される。コラム 803 は、障害予兆を検出した場合の対応が格納される。例えば、コア交代や仮想化機構スケジュール禁止といった内容である。コア交代とは、障害予兆を検出したことで、交代コアを割り当て

10

20

30

40

50

る事を意味する。また、仮想化機構スケジュール禁止は、当該コアあるいはプロセッサに対し、仮想化機構のスケジュールを禁止することを意味する。プロセッサ交代とは、当該プロセッサの全てのコアを他のプロセッサのコアと交代することを意味する。対応は、部位によって異なる。例えば、コアと1次キャッシュが同一コア内に内蔵され他のコアとは独立している場合には、コア単位の交代で良い。しかし、2次キャッシュが複数のコアから共有されている場合には、障害発生時の影響が大きいためプロセッサ単位で交代した方が良い。カラム804は障害時の対応が記載される。内容は、カラム803と同様である。このように、プロセッサのアーキテクチャによって、障害予兆検出時の優先度が異なる。これらをアーキテクチャ毎に管理するために障害対応テーブル111を設けている。

【0023】

図9は、仮想サーバコンソールの例を示している。図9では、新たに仮想サーバを生成する場合のユーザインターフェースを記述している。901はユーザインターフェースの全体図を示している。902～904は、仮想サーバに対するアクションを選択するメニューである。ここで選択されたメニューの内容が管理サーバコンソール901の右に表示される。図9では、仮想サーバ追加の内容が表示されている。905は、新たに生成する仮想サーバのIDを入力する。仮想サーバをどのシステムIDに生成するかは、ユーザが選択することができる。906は、生成する仮想サーバの特性を入力する。これは、本発明の特徴となる内容である。仮想サーバの特性には、例えば、信頼性重視や性能重視、あるいは一般といった特性が考えられる。信頼性重視とは、同じ量のリソースを割り当てる場合であっても、障害に対して強くなる割り当て方をする事を意味する。また、性能重視とは、信頼性よりも性能が高いことを重視する割り当て方を意味する。一般とは、信頼性も性能も重視せず、空きリソースを割り当てることを意味する。907はプロセッサの割り当て方を選択する。例えば、占有割り当て／共有割り当てといった割り当て方や、割り当ての単位としてコア単位やプロセッサ単位、またその数を記載する。908は、割り当てメモリ量を入力する。909は、割り当てるI/Oデバイスを入力する。リソースの入力の際には、仮想化機構管理テーブル109のリソースプールに記載された内容をユーザインターフェースに反映すると入力ミスを少なくでき、また入力が容易になる。全ての項目を入力し終わると、910のボタンを押すことで入力されたパラメータで新たな仮想サーバを生成することができる。

【0024】

図10は、コア割り当てプログラム103のフローチャートを示している。コア割り当てプログラム103は、図9におけるユーザインターフェースの906の入力結果に応じて動作が変わる。コア割り当てプログラム103の各ステップについて説明する。ステップ1001は信頼性重視が入力された場合に、ステップ1003に移行し信頼性重視型コア割り当て処理を行う。ステップ1002は、性能重視が入力された場合に、ステップ1004に移行し性能重視型コア割り当て処理を行う。いずれでもない場合は、ステップ1005に移行し、一般型コア割り当て処理を行う。

【0025】

次に、各割り当て処理について詳細なフローチャートで説明する。信頼性重視型コア割り当て処理1003は、仮想サーバ生成時に、信頼性が高くなるコアの割り当て方を行う。ここで、信頼性が高いコアの割り当てとは、単一の仮想サーバに複数のコアを割り当てる場合に、異なるプロセッサのコアを選択して割り当てることを意味する。ステップ1101は、サーバ構成テーブル107から、プロセッサ当たりのコア数を取得する。ステップ1102は、コア数から、分散するプロセッサ数を決定する。例えば、プロセッサ内のコア数が多いほど、できるだけ多くの異なるプロセッサのコアを選択することを意味する。これは、コア数が多いプロセッサほど、障害が発生した場合の影響が大きくなるためである。そのため、できるだけ多くの異なるプロセッサのコアを選択して割り当てることで、プロセッサ単位で障害が発生した場合でも、仮想サーバの全てのコアが障害によって停止することを防ぐことができる。ステップ1103は、仮想化機構管理テーブル109のリソースプールから、プロセッサを選択する。ステップ1104は、選択されたプロセッ

サが、全て同一プロセッサかどうか判定する。ステップ1105は、リソースプールから同タイプのプロセッサを選択する。これは、単一の仮想サーバに割り当てるコアを決定する場合に、できるだけ同じコア数を有するプロセッサを選択した方が、信頼性のばらつきを少なくすることができるためである。同一種類の空きプロセッサが存在しない場合は、できるだけコア数が近いプロセッサを選択するなどの方法も考えられる。ステップ1106は、仮想化機構管理テーブル109を更新する。ステップ1107は、選択されたコアに基づいて、仮想化機構115に対して、新たな仮想サーバの生成を指示する。このように、単一の仮想サーバに対して、異なるプロセッサのコアを選択することで、単一のプロセッサの障害によって、仮想サーバの全てのコアが同時に障害になることを防ぐことができる。

10

【0026】

図12は、性能重視型のコア割り当て処理1004のフローチャートを示している。ここで性能重視型とは、仮想サーバに割り当てるコアをできるだけ同一プロセッサのコアを割り当てることを意味する。これは、例えば複数のコアから共有される2次キャッシュを効率的に使用するためである。単一の仮想サーバ内で実行されるプログラムは、同じ命令やデータを参照／更新する可能性が高い。こういったケースでは、できるだけキャッシュを共有した方が、キャッシュのヒット率を向上させることができる。従って、キャッシュを効率よく使用し高い性能を得るためには、できるだけ同一プロセッサのコアを割り当てた方がよい。性能重視型のコア割り当て処理1004におけるステップ1201は、仮想化機構管理テーブル109のリソースプールから、プロセッサ当たりのコア数が多く、空

20

【0027】

きも多いコアを選択する。ステップ1203は、予定のコア数に達したかどうか判定する。予定のコア数に達していない場合は、ステップ1204にて、仮想化機構管理テーブル109のリソースプールから、プロセッサ当たりのコア数が多く、空きも多いコアを選択し、ステップ1203を繰り返す。ステップ1205は、仮想化機構管理テーブル109を更新する。ステップ1206は、選択されたコアに基づいて、仮想化機構115に対して、新たな仮想サーバの生成を指示する。これらの処理により、仮想サーバの性能を向上させる割り当て方を提供することができるようになる。

30

【0028】

図14は、信頼性重視型コア割り当て処理と、性能重視型コア割り当て処理の結果の一例を示している。1401は、信頼性重視型コア割り当て処理の結果である。このように、複数の異なるプロセッサのコアを、単一の仮想サーバに割り当てる事が特徴である。これにより、プロセッサに障害が発生しても、仮想サーバの全てを停止させることがない。一方、1402は、性能重視型コア割り当て結果の一例である。このように、できるだけ同一のプロセッサのコアを仮想サーバに割り当てる事が特徴である。これにより、共有される2次キャッシュの命令やデータを有効に使用することで、性能を向上させることができる。図では述べていないが、性能と信頼性の両方を重視する割り当て方法もある。例えば、異なるプロセッサの数を少なくし、その代わりに同一プロセッサのコアを優先して割り当てるといった2つの割り当て方の混在である。こうすることで、信頼性と性能を満たす仮想サーバを生成することも可能である。

40

【0029】

50

図15は、障害予兆検出機構302のフローチャートを示している。障害予兆検出機構302は、仮想化機構115内で実行され、プロセッサの空き（アイドル）時や、所定の時間を経過した場合にハードウェアのチェックを行う。障害予兆検出機構302におけるステップ1501は、プロセッサがアイドルかどうかを判定する。ステップ1502は、前回のチェックから所定の時間を経過したかを判定する。ステップ1501あるいはステップ1502のいずれかの条件が一致すれば、ステップ1503以降のハードウェアチェックを実行する。ステップ1503は、プロセッサコアチェックを実行する。プロセッサのコアに障害の予兆がないかチェックする処理を行う。ステップ1504はプロセッサ内の1次キャッシュに障害の予兆がないかチェックする。ステップ1505は、プロセッサ内の2次キャッシュに障害の予兆がないかチェックする。ステップ1506は、プロセッサ内のメモリバスに障害の予兆がないかチェックする。これらプロセッサ内の障害予兆チェックには、例えば、プロセッサに対しチェックのための専用命令を発効するなどの方法が考えられる。ステップ1507は、メモリに障害の予兆がないかチェックする。メモリのチェックは、例えば、パリティやECC（Error Check and Correct memory）等のメモリエラー検出機構を使用する等が考えられる。ステップ1508は、I/Oデバイスチェックを行う。ステップ1509は、プロセッサコントローラチェックを行う。ステップ1510では、ステップ1503～ステップ1509までのチェック結果をプロセッサ管理テーブル110に書き込む。ステップ1511では、チェックした現在の時間を記録する。これは、次回のチェック判定（ステップ1502）に使用される。また、チェック時に障害の予兆ではなく、障害が発生していた場合にも、障害予兆と同様にプロセッサ管理テーブル110に記録する。これらの処理により、ハードウェアの障害を事前に把握することができるようになる。

10

20

【0030】

図16は、障害管理プログラム104のフローチャートを示している。障害管理プログラム104におけるステップ1601は、障害が発生したかどうかを判定する。これは、プロセッサ管理テーブル110を参照することで判定することができる。ステップ1602は、障害の予兆を検出したかどうかを判定する。これもステップ1601と同様に、プロセッサ管理テーブル110を参照することで判定することができる。障害が発生していた場合には、ステップ1603に移行し、障害対策プログラムを実行する。また、障害の予兆が検出された場合には、ステップ1604に移行し、障害予兆対策プログラムを実行する。

30

【0031】

図17は、障害対策プログラム1603のフローチャートを示している。障害が発生したプロセッサやコアは、基本的には他のプロセッサやコアと交代する処理を行う。障害対策プログラム1603におけるステップ1701は、プロセッサ管理テーブル110から障害発生プロセッサを検索する。ステップ1702は、プロセッサ管理テーブル110から障害発生部位を検索する。ステップ1703は障害対応テーブル111を参照し、障害部位に対応した処理を検索する。ステップ1704は、障害対応がコア交代かどうかを判定する。コア交代の場合は、ステップ1605に移行し、仮想化機構管理テーブル109を検索し、障害発生プロセッサとは異なるプロセッサのコアを交代コアとして選択する。ステップ1706は、障害コアを仮想サーバから切り離し、ステップ1705にて選択した交代コアを割り当てる。ステップ1707は、障害対応がプロセッサ交代かどうかを判定する。プロセッサ交代の場合は、ステップ1708に移行し、仮想化機構管理テーブル109を検索し、異なるプロセッサを交代プロセッサとして選択する。ステップ1709では、障害プロセッサを切り離し、交代プロセッサを割り当てる。これらの処理により、障害が発生したコア、あるいはプロセッサに割り当てられていた仮想サーバを正常な状態に戻すことができる。

40

【0032】

図18は、障害予兆対策プログラム1604のフローチャートを示している。障害予兆が検出したプロセッサやコアは、基本的には他のプロセッサやコアと交代する処理を行う

50

。障害予兆対策プログラム1604におけるステップ1801は、プロセッサ管理テーブル110から障害予兆が検出されたプロセッサを検索する。ステップ1802は、プロセッサ管理テーブル110から障害予兆の発生部位を検索する。ステップ1803は障害対応テーブル111を参照し、障害予兆部位に対応した処理を検索する。ステップ1804は、スケジュール制御機構301に対し、概要部位を仮想化機構115の実行スケジュール対象から外す指示を発行する。これは、仮想化機構115を将来発生する可能性がある障害から保護することを目的としている。仮想化機構115は、生成された仮想サーバを含むシステム全体に影響を及ぼす可能性があるため、システム内において優先して保護すべき対象である。このステップ1804によって、障害予兆を検出したプロセッサあるいはコアをスケジュール対象から外すことで、システム全体の信頼性を向上することができる。ステップ1805は、障害予兆対応がコア交代かどうかを判定する。コア交代の場合は、ステップ1806に移行し、仮想化機構管理テーブル109を検索し、障害予兆を検出したプロセッサとは異なるプロセッサのコアを交代コアとして選択する。ステップ1807は、マイグレーション制御プログラム呼び出しを行う。ここで、マイグレーションとは、コアの交代時にシステムに影響を与えないようにする処理である。ステップ1808は、障害予兆対応がプロセッサ交代かどうかを判定する。プロセッサ交代の場合は、ステップ1809に移行し、仮想化機構管理テーブル109を検索し、異なるプロセッサを交代プロセッサとして選択する。ステップ1810では、マイグレーション制御プログラムの呼び出しを行う。これらの処理により、障害予兆が検出されたコア、あるいはプロセッサをシステムに影響を及ぼすことなく仮想サーバを正常な状態に戻すことができる。

10

20

【0033】

図19は、スケジュール制御機構301のフローチャートを示す。スケジュール制御機構301におけるステップ1901は、コア障害の予兆を検出したかどうかを検出する。コア障害の予兆を検出した場合、ステップ1902に移行し、該当コアをスケジュール対象から外すように、割り込み制御テーブル304を変更する。ステップ1903は、プロセッサ全体に影響する障害予兆を検出したかどうかを判定する。例えば、複数のコアから共有される2次キャッシュやメモリバスなどの障害予兆である。プロセッサ全体に影響を及ぼす障害の予兆を検出した場合には、該当プロセッサの全コアをスケジュール対象から外すように、割り込み制御テーブル304を変更する。

【0034】

図20は、割り込み制御テーブル304を示している。カラム2001は、プロセッサIDが格納される。カラム2002は、コアIDが格納される。カラム2003は、割り込みレベルに応じた割り込み可／不可が格納される。割り込みレベルは、プロセッサアーキテクチャによって、レベル数や意味が異なるが、障害予兆を検出した場合には、プログラムの保護を目的として割り込みを不可に変更する。これにより、仮想化機構115のようなシステム全体に影響を及ぼすプログラムを保護することができる。一般に、プロセッサによって実行されるプログラムは、割り込みを受け付けたプロセッサ内のコアで実行される。コアやプロセッサに障害が発生していない場合は、発生した割り込みは複数のコアで均等にプログラムが実行されるように分散される。しかし、障害予兆を検出した場合は、近い将来に障害が発生する可能性を有するため、特に仮想化機構115のような信頼性を重視されるプログラムは保護する必要がある。そのため、障害予兆を検出したコアやプロセッサには、割り込みを不可にすることで、実行スケジュールから外す処理を行う。図20では、例えばコアの障害予兆を検出したプロセッサ1のコア1は割り込みが不可に設定される。また、プロセッサ全体に影響する障害予兆を検出したプロセッサ2は全てのコアに対して、割り込みが不可となるように設定される。

30

40

【0035】

図21は、マイグレーション制御プログラム106のフローチャートを示している。マイグレーション制御プログラム106におけるステップ2101は、コア障害予兆を検出したかどうかを判定する。コア障害予兆を検出した場合は、ステップ2102に移行し、プロセッサ制御機構303に対し、障害予兆検出コアと交代コアの制御モードをステップ

50

実行に切替指示を行う。これは、プロセッサの実行モードをステップ実行に変更することで、命令がどこまで実行されたかを正確に把握するためである。通常の実行モードでは、コア内で複数の命令が同時に実行されるため、どこまでの命令を実行したか判定することができない。そのため、プロセッサを一時的にステップ実行モードに移行し、外部からプロセッサの命令レベルの動作を管理する。交代コアをステップ実行モードに移行するのは、実行を外部から制御するためである。ステップ2103は、プロセッサ制御機構303から、該当コアの状態を取得する。ここで状態とは、交代コアに制御をスムーズに移行するために必要な情報である。例えば、プログラムカウンタのように命令の実行アドレスが格納された制御レジスタの内容などである。ステップ2104は、交代コアに対し、コアの状態を設定する。ステップ2105では、コア交代のために、障害予兆検出コアを停止し、交代コアの実行を指示する。これにより、システムに影響を与えることなく、交代コアに移行することができる。ステップ2106は、プロセッサ障害予兆を検出したかどうかを判定する。プロセッサ障害予兆を検出した場合には、ステップ2107に移行し、プロセッサ制御機構303に対し、障害予兆検出プロセッサと交代プロセッサの全コアの制御モードをステップ実行に切替指示を行う。ステップ2108は、プロセッサ制御機構303から、該当する全コアの状態を取得する。ステップ2109は、交代プロセッサのコアに対して、ステップ2108で取得したコアの状態を設定する。ステップ2110は、障害予兆検出プロセッサを停止し、交代プロセッサの実行を指示する。これにより、システムに影響を与えることなく、プロセッサ全体のマイグレーションが可能になる。

10

【0036】

20

図22は、プロセッサ制御機構303のフローチャートを示す。プロセッサ制御機構303におけるステップ2201は、通常実行モードへの変更指示かどうかを判定する。通常実行モードへの変更指示の場合はステップ2202に移行し、指定コアを通常実行モードに切り替える。ステップ2203は、ステップ実行モードへの切替かどうかを判定する。ステップ実行モードへの切替の場合は、指定コアをステップ実行モードに切り替える。

【0037】

上記実施例1により、プロセッサの障害の予兆検出によって事前に障害を回避し、かつ障害が発生した場合でも、複数のプロセッサのコアを単一の仮想サーバに割り当てることで信頼性を高めることができる。また、ユーザインタフェースとして、高信頼重視と性能重視を設け、仮想サーバに対するコアの割り当てを制御することで、プロセッサを効率よく使用することができるようになる。

30

【0038】

なお、本実施例1では、仮想化機構管理プログラム102や各種テーブルは管理サーバ101のメモリ2901に格納されているが、サーバ装置112のメモリ113に格納されても良い。例えば、サーバ装置112に仮想化機構管理プログラム102や各種テーブルを配置する方法の一つとして、仮想サーバ114の一つに管理サーバ101の仮想化機構管理プログラム102や各種テーブルを配置しても良い。

【0039】

また、本実施例1では、仮想化機構115を用いてプロセッサコア116の割当てを行っているが、仮想化機構115は必ずしも必要ではない。例えば、単一のオペレーティングシステムがマルチコアの割当てを行う場合にも本発明を適用できる。例えば、オペレーティングシステムのプロセスやスレッドといったプログラム実行環境に対して本発明を適用できる。また他の例では、オペレーティングシステムが異なる複数のオペレーティングシステムを実行可能な環境を生成する場合にも本発明を適用できる。

40

【実施例2】

【0040】

図23は、本発明における実施例2のプロセッサの構成を示している。実施例1におけるコアは、実施例2における汎用コアであるが、実施例2では同種のコアだけでなく、異なる種類のコアが搭載されたプロセッサ119における管理方法について述べる。大きく異なる部分は、プロセッサ119に専用コア2301を搭載している点である。専用コア

50

2301とは、特定の処理を高速化することを目的としたコアを意味する。例えば、浮動小数点演算の高速化や、文字列処理の高速化などが考えられる。

【0041】

図24は、実施例2における障害対応テーブル111の構成を示している。実施例1とは異なり、カラム802のプロセッサ部位に専用コアの項目が追加される。実施例2では、専用コアの数が汎用コアに比べて少ないため、障害に対する優先順位は汎用コアよりも高く設定することを意味する。つまり、数が少ない種類のコアの障害はシステムに与える影響が大きいため交替を優先するように設定する。実施例2では、専用コアより汎用コアの数の多い例を示しているが、逆のケースも考えられる。本発明においては、専用コアか汎用コアかに関係なく、数が少ないコアの交替優先度を高めることが特徴となる。このよ

10

【0042】

図25は、実施例2における障害対策プログラム1603のフローチャートを示している。障害が発生したプロセッサやコアは、基本的には他のプロセッサやコアと交代する処理を行う。実施例2における障害対策プログラム1603におけるステップ1701は、プロセッサ管理テーブル110から障害発生プロセッサを検索する。ステップ1702は、プロセッサ管理テーブル110から障害発生部位を検索する。ステップ1703は障害対応テーブル111を参照し、障害部位に対応した処理を検索する。ステップ1704は、障害対応がコア交代かどうかを判定する。コア交代の場合は、ステップ1605に移行し、仮想化機構管理テーブル109を検索し、障害発生プロセッサとは異なるプロセッサ

20

【0043】

図26は、専用コア交替プログラム2502のフローチャートを示す。専用コア交替プログラム2502におけるステップ2601は、交替のための専用コアが存在するかどうかを判定する。交替コアがあれば、ステップ2602に移行し、仮想化機構管理テーブル109を検索し、障害発生プロセッサとは異なるプロセッサのコアを選択する。ここでコアとは専用コアを意味する。ステップ2603では、障害コアを切り離し、交替コアを割り当てる。一方、交替コアが存在しなかった場合は、ステップ2604に移行し、仮想化機構管理テーブル109を検索し、障害が発生したプロセッサとは異なるプロセッサの汎用コアを選択する。ステップ2605では、汎用コア上で専用コアの機能をエミュレーションするプログラムを実行し、専用コアの交替とする。ステップ2606は、当該汎用コアを専用コアとして割り当てる。これらの処理により、数が少ないコアを汎用コアでエミ

40

【0044】

図27は、プロセッサやプロセッサのコア障害時にアプリケーションプログラムのリカバリ例を示している。仮想サーバ114内では、オペレーティングシステム2704が稼働し、またその上でアプリケーションプログラム2701が実行されている。アプリケーションプログラムは、スナップショット取得プログラム2702とリカバリプログラム2703を有する。スナップショット取得プログラム2702は、定期的にアプリケーションの動作状態を記録する機能を有する。これは、障害が発生した際に、再実行可能にするためである。例えば、アプリケーションプログラムの実行結果等である。プロセッサやコ

50

アに障害が発生した場合には、スナップショット取得プログラム 2702 が取得した状態を、リカバリプログラム 2703 が読み込み、アプリケーションプログラム 2701 の実行を再開することで目的を果たすことができる。

【実施例 3】

【0045】

図 28 は、実施例 3 におけるサーバ構成テーブル 108 を示す。実施例 1 と異なる箇所はカラム 2801 が追加されたことである。カラム 2801 は、仮想化機構どうしが連携しあうことを意味している。システム ID は異なるが、仮想化機構がお互いに連携することで、複数のシステムを単一のシステムに見せることができる。図 28 では、仮想化機構 1 と仮想化機構 2 が連携している事を意味している。こういった場合、障害予兆や障害を検出した場合に、交代の対象を拡大することができる。例えば、仮想化機構 1 に属するプロセッサ ID 1 のコアに障害が発生した場合に、プロセッサ 3 が有するコアを交代の対象に選択することができる。この連携によって、システムの信頼性や可用性を向上することができるようになる。

10

【産業上の利用可能性】

【0046】

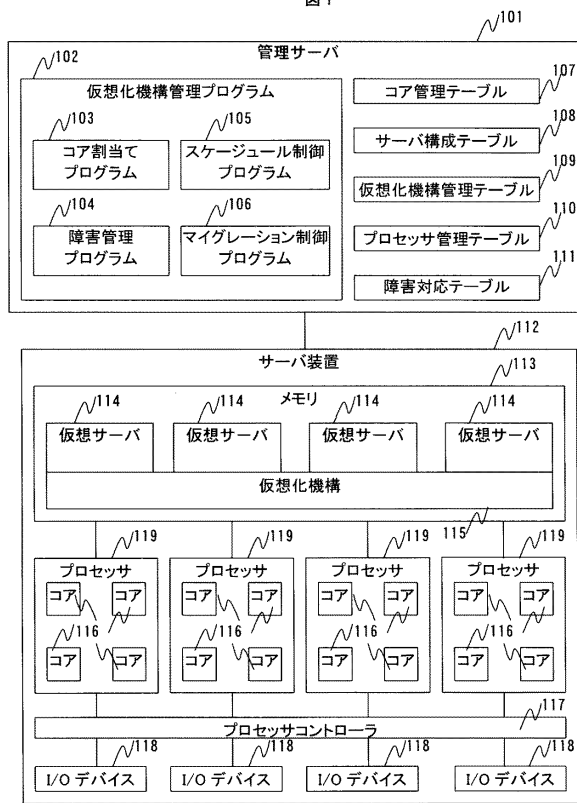
マルチコアプロセッサで稼働する計算機を用いる分野に利用される。

【符号の説明】

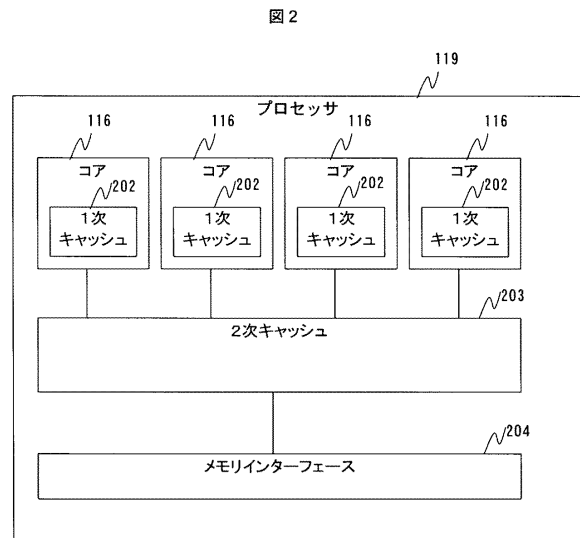
【0047】

- | | | |
|-----|-----------------|----|
| 101 | 管理サーバ | 20 |
| 102 | 仮想化機構管理プログラム | |
| 103 | コア割り当てプログラム | |
| 104 | 障害管理プログラム | |
| 105 | スケジュール制御プログラム | |
| 106 | マイグレーション制御プログラム | |
| 107 | コア管理テーブル | |
| 108 | サーバ構成テーブル | |
| 109 | 仮想化機構管理テーブル | |
| 110 | プロセッサ管理テーブル | |
| 111 | 障害対応テーブル | 30 |
| 112 | サーバ装置 | |
| 115 | 仮想化機構 | |
| 114 | 仮想サーバ | |
| 119 | プロセッサ | |
| 116 | コア | |

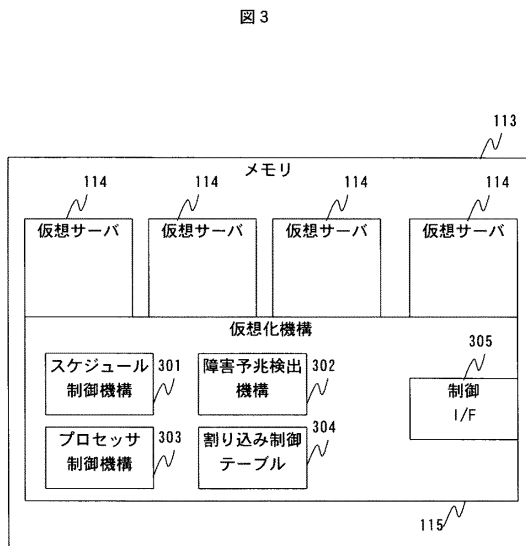
【図 1】



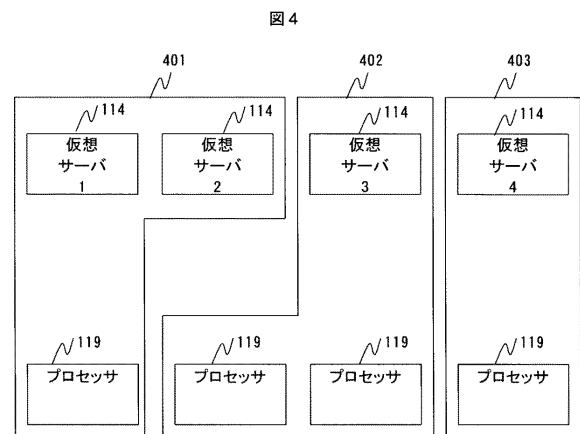
【図 2】



【図 3】



【図 4】



【図 5】

図 5

サーバ構成テーブル¹⁰⁸

システム ID	プロセッサ構成	プロセッサタイプ	メモリ容量	I/Oデバイス	仮想化機構 ID
SMP1	プロセッサ 1	4 コア	16GB	ネットワーク 1 ネットワーク 2	仮想化機構 1
	プロセッサ 2	4 コア		ディスクアダプタ 1 ディスクアダプタ 2	
SMP2	プロセッサ 3	8 コア	32GB	ネットワーク 1 ネットワーク 2	仮想化機構 2
	プロセッサ 4	8 コア		ネットワーク 3 ディスクアダプタ 1	
	プロセッサ 5	8 コア		ディスクアダプタ 2	
	プロセッサ 6	8 コア			

【図 6】

図 6

仮想化機構管理テーブル¹⁰⁹

仮想化機構 ID	仮想サーバ ID	割り当て プロセッサ ID	割り当て コア ID	割り当て メモリ	割り当て I/Oデバイス
仮想化機構 1	仮想サーバ 1	プロセッサ 1	コア 1	4GB	ネットワーク 1
		プロセッサ 2	コア 1		ディスクアダプタ 1
	仮想サーバ 2	プロセッサ 1	コア 2	8GB	ネットワーク 2
		プロセッサ 2	コア 2		ディスクアダプタ 2
仮想化機構 2	リソース プール	プロセッサ 1	コア 3	4GB	-
		プロセッサ 1	コア 4		
		プロセッサ 2	コア 3		
		プロセッサ 2	コア 4		
	仮想サーバ 3	プロセッサ 3	コア 1	16GB	ネットワーク 1 ネットワーク 2 ディスクアダプタ 1 ディスクアダプタ 2
		プロセッサ 4	コア 1		
		プロセッサ 5	コア 1		
		プロセッサ 6	コア 1		
	リソース プール	プロセッサ 3	コア 2~8	16GB	ネットワーク 3
		プロセッサ 4	コア 2~8		
		プロセッサ 5	コア 2~8		
		プロセッサ 6	コア 2~8		

【図 7】

図 7

プロセッサ管理テーブル¹¹⁰

プロセッサ ID	コア ID	状態
プロセッサ 1	コア 1	正常
	コア 2	
	コア 3	
	コア 4	
プロセッサ 2	コア 1	障害予兆検出
	コア 2	コア 3
	コア 3	障害
	コア 4	コア 4
プロセッサ 3	コア 1	障害予兆検出
	コア 2	2 次キャッシュ
	コア 3	
	コア 4	
	コア 1	
	コア 2	
	コア 3	
	コア 4	
プロセッサ 4	コア 1	正常
	コア 2	
	コア 3	
	コア 4	
	コア 1	
	コア 2	
	コア 3	
	コア 4	

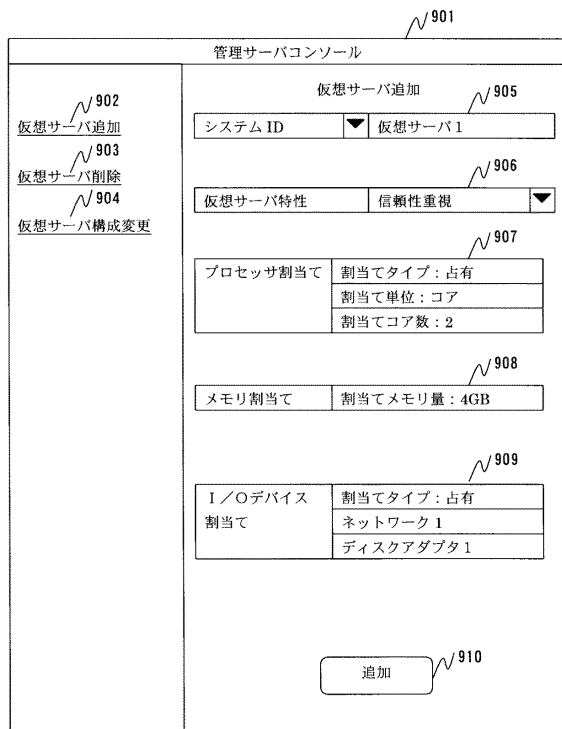
【図 8】

図 8

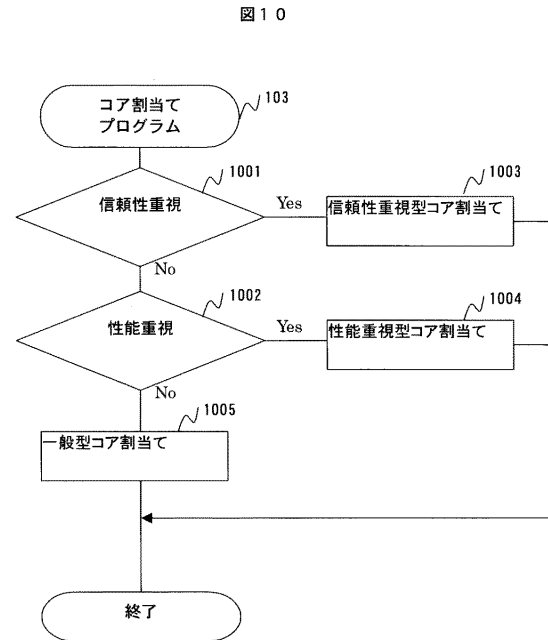
障害対応テーブル¹¹¹

プロセッサ種類	プロセッサ部位	障害予兆検出時対応	障害時対応
プロセッサ タイプ 1	コア	コア交代 仮想化機構スケジュール禁止	コア交代
	1 次キャッシュ	コア交代 仮想化機構スケジュール禁止	コア交代
	2 次キャッシュ	プロセッサ交代 仮想化機構スケジュール禁止	プロセッサ交代
	メモリバス	プロセッサ交代 仮想化機構スケジュール禁止	プロセッサ交代
		仮想化機構スケジュール禁止	
プロセッサ タイプ 2	コア	コア交代 仮想化機構スケジュール禁止	コア交代
	1 次キャッシュ	コア交代 仮想化機構スケジュール禁止	コア交代
	2 次キャッシュ	コア交代 仮想化機構スケジュール禁止	コア交代
	メモリバス	プロセッサ交代 仮想化機構スケジュール禁止	プロセッサ交代
		仮想化機構スケジュール禁止	

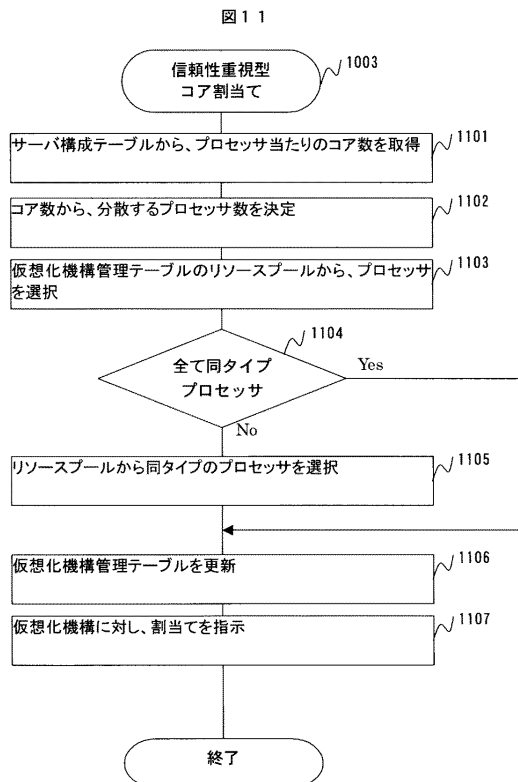
【図 9】



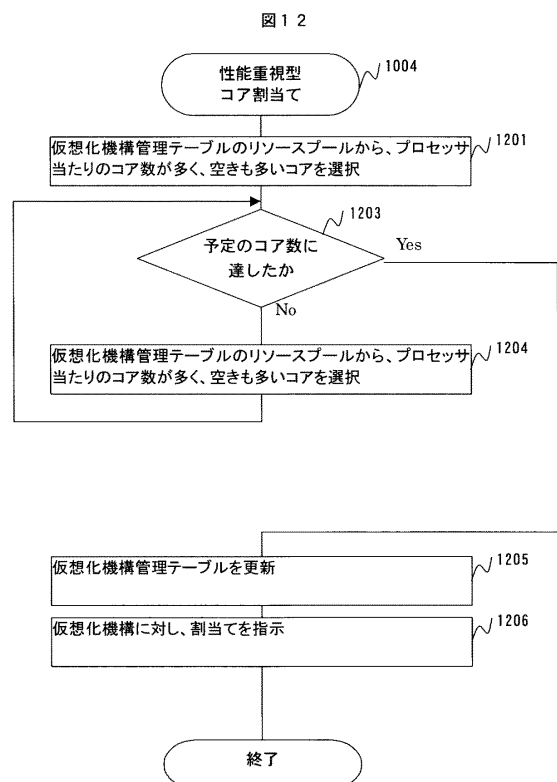
【図 10】



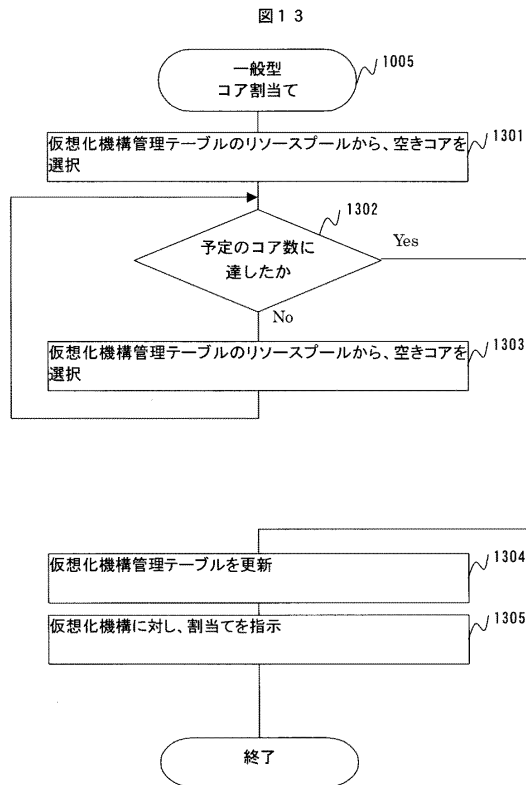
【図 11】



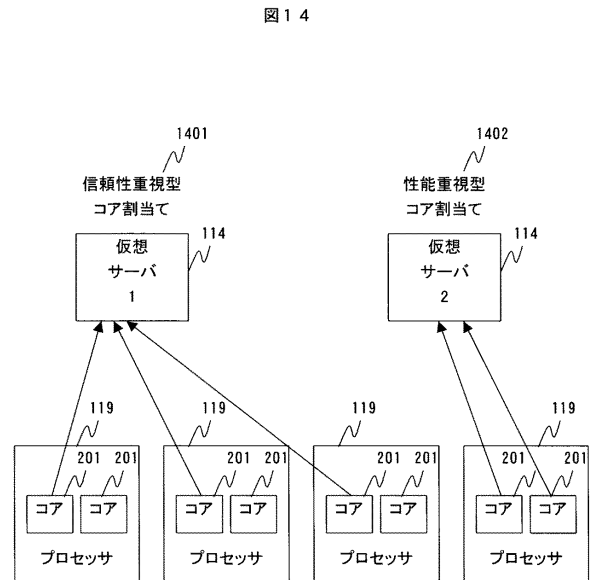
【図 12】



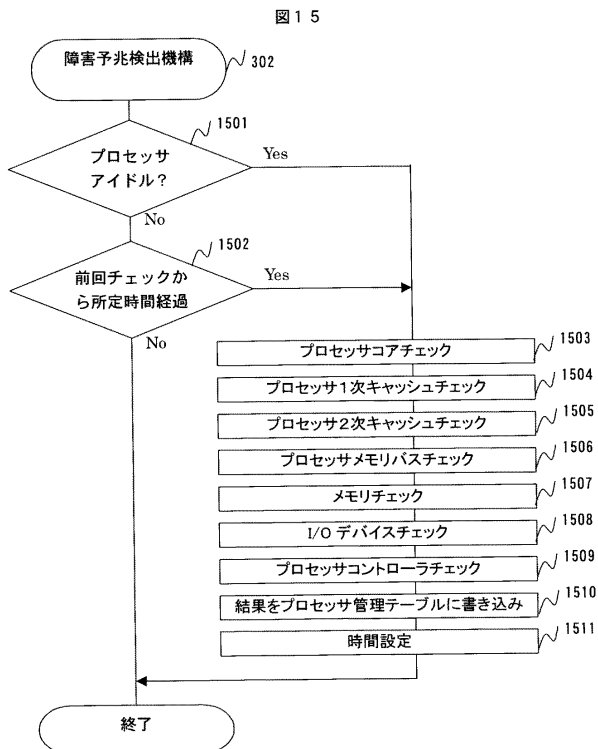
【図 13】



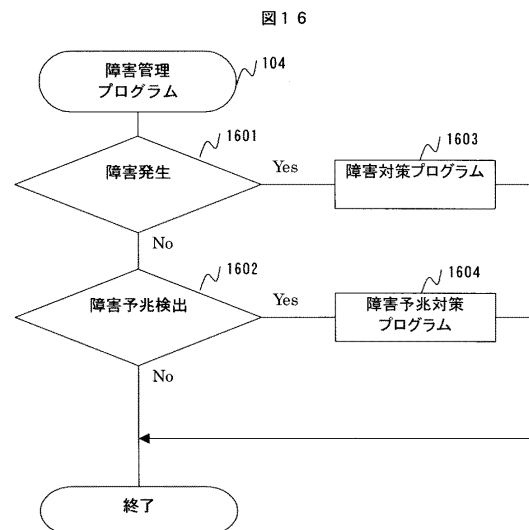
【図 14】



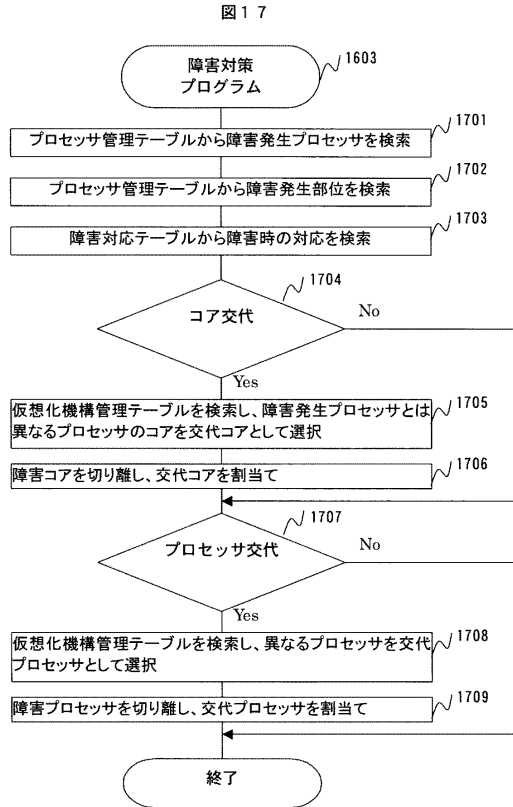
【図 15】



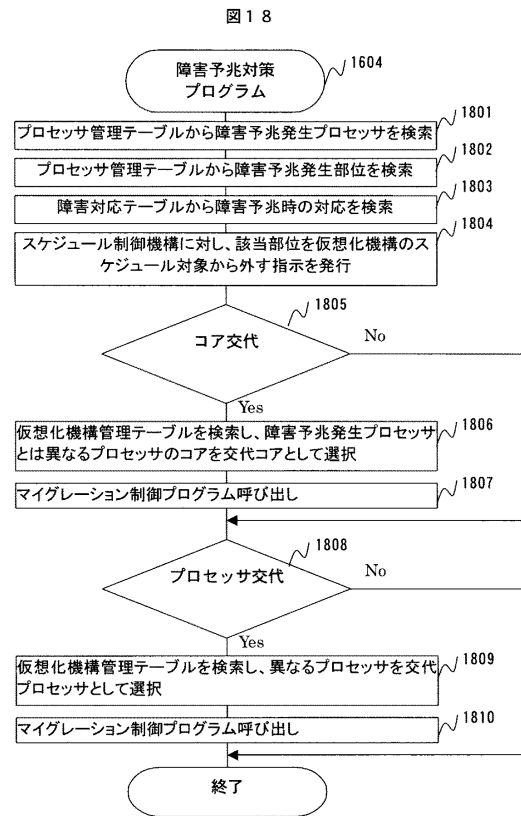
【図 16】



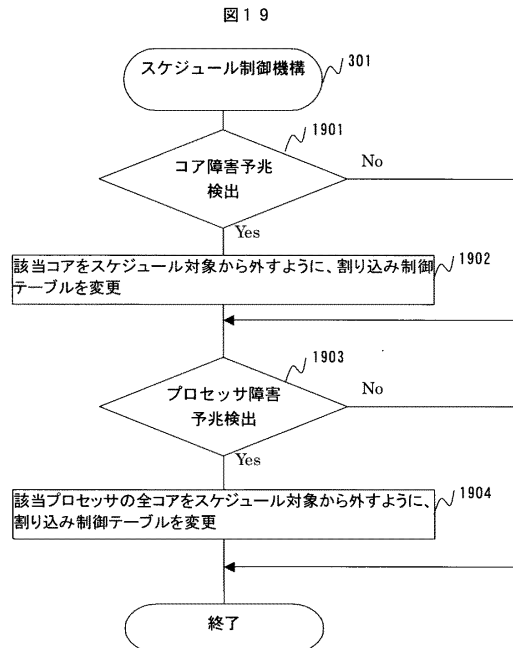
【図 17】



【図 18】



【図 19】



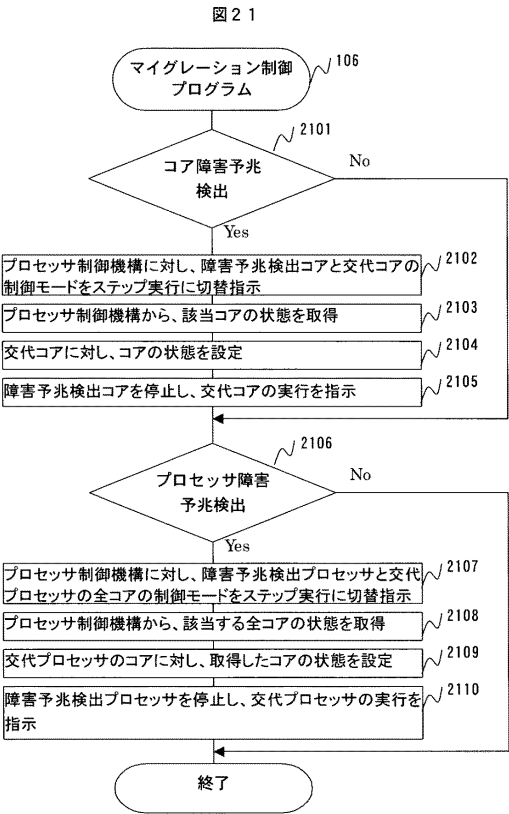
【図 20】

図 20

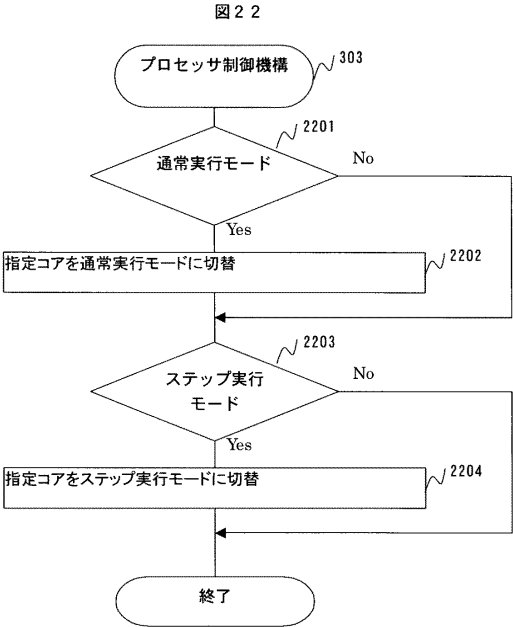
割り込み制御テーブル

プロセッサ ID	コア ID	割り込みレベル			
		レベル 1	レベル 2	レベル 3	レベル 4
プロセッサ 1	コア 1	不可	不可	不可	不可
	コア 2	可	可	可	可
	コア 3	可	可	可	可
	コア 4	可	可	可	可
プロセッサ 2	コア 1	不可	不可	不可	不可
	コア 2	不可	不可	不可	不可
	コア 3	不可	不可	不可	不可
	コア 4	不可	不可	不可	不可
プロセッサ 3	コア 1	可	可	可	不可
	コア 2	可	可	可	不可
	コア 3	可	可	可	不可
	コア 4	可	可	可	不可
	コア 1	可	可	可	不可
	コア 2	可	可	可	不可
	コア 3	可	可	可	不可
	コア 4	可	可	可	不可
プロセッサ 4	コア 1	可	可	不可	不可
	コア 2	可	可	可	不可
	コア 3	可	可	不可	不可
	コア 4	可	可	可	不可
	コア 1	可	可	不可	不可
	コア 2	可	可	可	可
	コア 3	可	可	可	不可
	コア 4	可	可	不可	不可

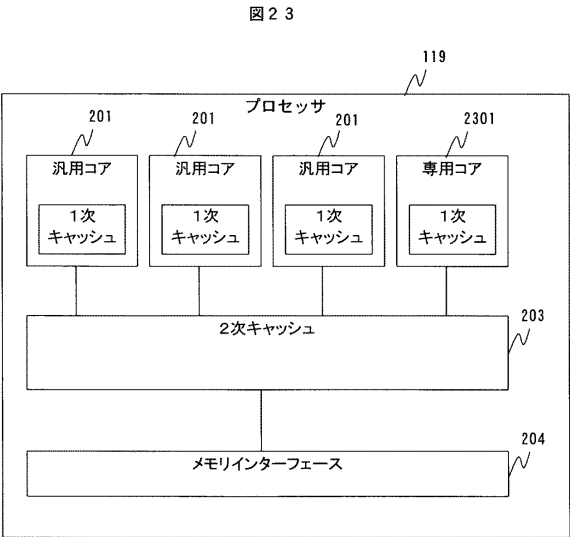
【図 2 1】



【図 2 2】



【図 2 3】



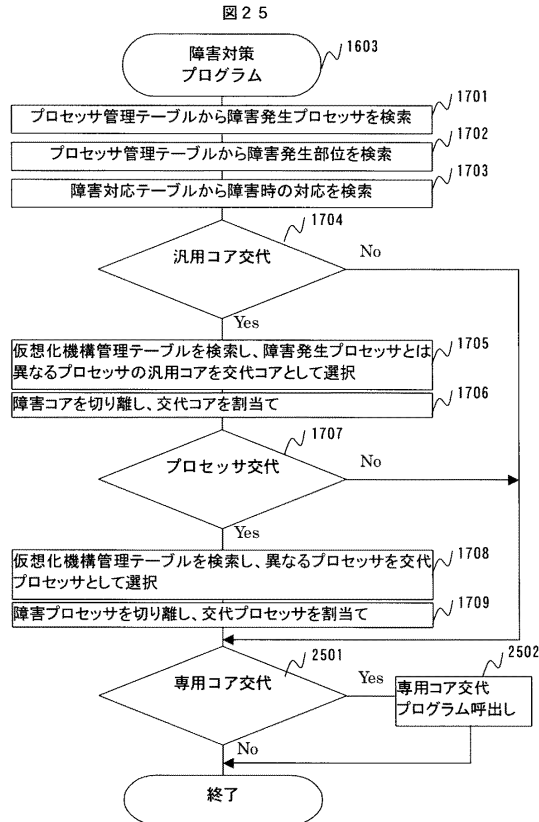
【図 2 4】

図 2 4

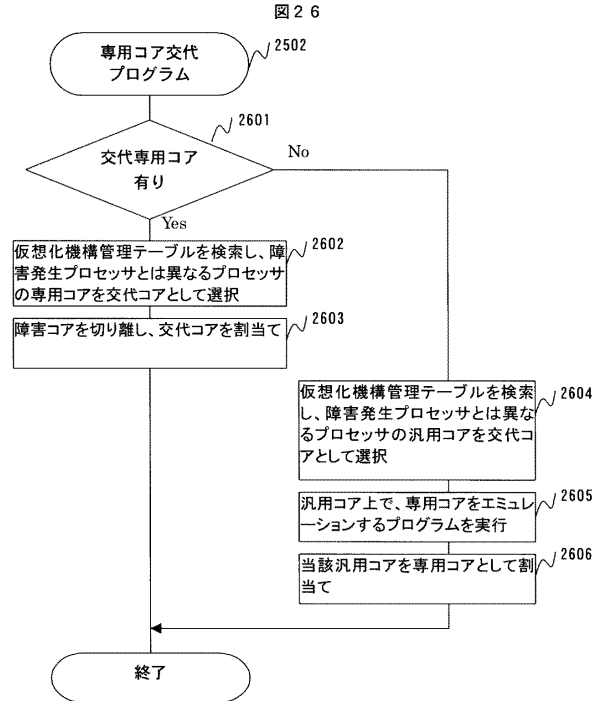
障害対応テーブル 111

801 プロセッサ種別	802 プロセッサ部位	2401 障害予兆検出時対応	障害時対応
プロセッサ タイプ 1	汎用コア	コア交代 仮想化機構スケジュール禁止	コア交代
	専用コア	プロセッサ交代	コア交代
	1 次キャッシュ	コア交代 仮想化機構スケジュール禁止	コア交代
	2 次キャッシュ	プロセッサ交代 仮想化機構スケジュール禁止	プロセッサ交代
	メモリバス	プロセッサ交代 仮想化機構スケジュール禁止	プロセッサ交代
プロセッサ タイプ 2	汎用コア	コア交代 仮想化機構スケジュール禁止	コア交代
	専用コア	プロセッサ交代	コア交代
	1 次キャッシュ	コア交代 仮想化機構スケジュール禁止	コア交代
	2 次キャッシュ	コア交代 仮想化機構スケジュール禁止	コア交代
	メモリバス	プロセッサ交代 仮想化機構スケジュール禁止	プロセッサ交代

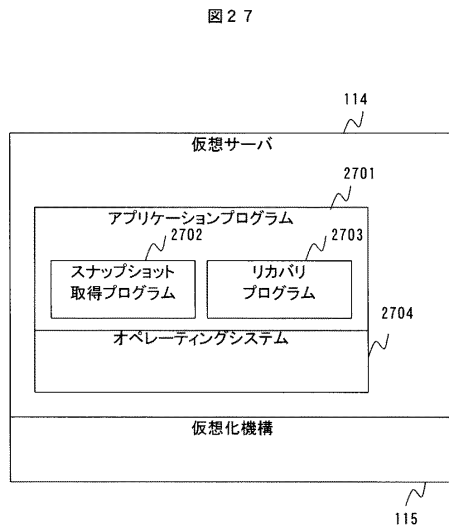
【図 25】



【図 26】



【図 27】



【図 28】

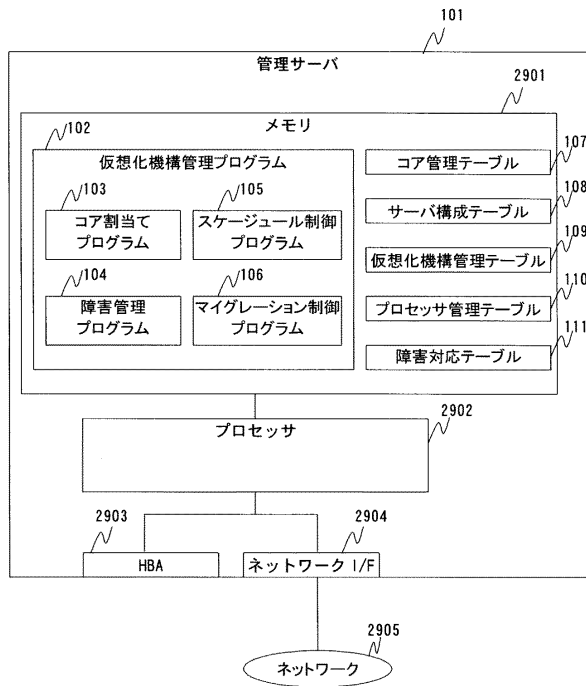
図 28

サーバ構成テーブル 108

501 システム ID	502 プロセッサ構成	503 プロセッサタイプ	504 メモリ容量	505 I/Oデバイス	506 仮想化機構 ID	2801 連携
SMP1	プロセッサ 1	4 コア	16GB	ネットワーク 1	仮想化機構 1	仮想化機構 2
	プロセッサ 2	4 コア		ネットワーク 2 ディスクアダプタ 1 ディスクアダプタ 2		
SMP2	プロセッサ 3	8 コア	32GB	ネットワーク 1	仮想化機構 2	仮想化機構 1
	プロセッサ 4	8 コア		ネットワーク 2		
	プロセッサ 5	8 コア		ネットワーク 3		
	プロセッサ 6	8 コア		ディスクアダプタ 1 ディスクアダプタ 2		

【図 29】

図 29



フロントページの続き

- (56)参考文献 特開2004-030363 (JP, A)
特開2006-011913 (JP, A)
特開2003-330740 (JP, A)
特開2003-173272 (JP, A)
特開2004-326749 (JP, A)
特表2005-502119 (JP, A)
特開2004-133753 (JP, A)
国際公開第2006/100747 (WO, A1)
特開2004-318885 (JP, A)
特開2006-178933 (JP, A)
特開2001-022599 (JP, A)
特開2001-297010 (JP, A)

- (58)調査した分野(Int.Cl., DB名)
G06F 11/20