(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2005-500622
(P2005-500622A)
(43)公表日　平成17年1月6日(2005.1.6)

(51)Int.Cl.[7]	F I			テーマコード（参考）
G06F 9/46	G06F	9/46	350	5B083
G06F 13/00	G06F	9/46	340F	5B098
	G06F	13/00	301M	

審査請求 未請求　予備審査請求 有　（全 57 頁）

(21)出願番号	特願2003-521964(P2003-521964)
(86)(22)出願日	平成14年8月9日(2002.8.9)
(85)翻訳文提出日	平成16年2月13日(2004.2.13)
(86)国際出願番号	PCT/US2002/025530
(87)国際公開番号	WO2003/017126
(87)国際公開日	平成15年2月27日(2003.2.27)
(31)優先権主張番号	09/932,456
(32)優先日	平成13年8月16日(2001.8.16)
(33)優先権主張国	米国(US)

(71)出願人　504056174
ニューイシス・インコーポレーテッド
ＮＥＷＩＳＹＳ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国　テキサス州７８７５９，オースティン，ジョリービル・ロード，１０８１４，ビルディング　４，スイート　３００
(74)代理人　110000028
特許業務法人明成国際特許事務所
(72)発明者　オーラー・リチャード・アール.
アメリカ合衆国　ニューヨーク州１０５８９　サマーズ，ボニー・ドライブ，８

最終頁に続く

(54)【発明の名称】データ転送ルーティングメカニズムを用いるコンピュータシステムパーティショニング

(57)【要約】
【課題】複数のプロセッサを含む複数のリソース、前記複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャ、および前記複数のリソースを少なくとも１つのパーティションにコンフィギャする少なくとも１つのパーティショニングプロセッサを有するコンピュータシステムを提供する。
【解決手段】それぞれのパーティションは、前記複数のリソースのサブセットを備える。前記少なくとも１つのパーティショニングプロセッサは、以前に特定されたパーティショニングスキーマにしたがって前記複数のプロセッサのうちの少なくとも１つと前記複数のプロセッサのうちの少なくとも１つの他のものとの間の少なくとも１つのリンクをイネーブルすることによって前記リソースをコンフィギャするよう動作可能である。そのようにイネーブルされたリンク（群）は、前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する。

【特許請求の範囲】
【請求項１】
複数のプロセッサを含む複数のリソース、
前記複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャ、および
前記複数のリソースを少なくとも１つのパーティションにコンフィギャする少なくとも１つのパーティショニングプロセッサであって、それぞれのパーティションは、前記複数のリソースのサブセットを備え、前記少なくとも１つのパーティショニングプロセッサは、以前に特定されたパーティショニングスキーマにしたがって前記プロセッサに関連付けられた複数のルーティングテーブルのうちの少なくとも１つに書き込むことによって前記リソースをコンフィギャするよう動作可能であり、それぞれのルーティングテーブルは関連付けられたプロセッサおよび前記複数のプロセッサのうちの他のプロセッサの間のリンクを表し、前記リンクは前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する、少なくとも１つのパーティショニングプロセッサ
を備えるコンピュータシステム。
【請求項２】
請求項１に記載のコンピュータシステムであって、前記複数のリソースは、メモリデバイス、メモリ範囲、Ｉ／Ｏバス、Ｉ／Ｏバスに接続されたＩ／Ｏデバイス、およびルーティング割り込みのための割り込みメカニズムのうちの少なくとも１つを含むコンピュータシステム。
【請求項３】
請求項１に記載のコンピュータシステムであって、前記複数のリソースは、Ｉ／Ｏスイッチを含み、関連付けられた前記ルーティングテーブルのうちの１つを持つ前記Ｉ／Ｏスイッチは、前記Ｉ／Ｏスイッチ、前記プロセッサのうちの少なくとも１つ、および少なくとも１つのＩ／Ｏリソースの間のリンクを表すコンピュータシステム。
【請求項４】
請求項３に記載のコンピュータシステムであって、前記少なくとも１つのＩ／Ｏリソースは少なくとも１つのイーサネットデバイスおよびＳＣＳＩデバイスを備えるコンピュータシステム。
【請求項５】
請求項１に記載のコンピュータシステムであって、それぞれのルーティングテーブルはエントリ群のテーブルを備え、前記エントリ群の選択されたもののうちのそれぞれは、前記リソースのうちの１つのアドレスを、前記プロセッサのうちの１つおよび前記プロセッサのうちの前記１つと接続するリンクに関連付けるコンピュータシステム。
【請求項６】
請求項１に記載のコンピュータシステムであって、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャはコヒーレントハイパートランスポート（ｃＨＴ）インフラストラクチャを備えるコンピュータシステム。
【請求項７】
請求項１に記載のコンピュータシステムであって、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャは、前記プロセッサをリングトポロジを用いて相互接続するコンピュータシステム。
【請求項８】
請求項１に記載のコンピュータシステムであって、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャは、前記プロセッサをメッシュトポロジを用いて相互接続するコンピュータシステム。
【請求項９】
請求項１に記載のコンピュータシステムであって、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャは、前記プロセッサのそれぞれを前記プロセッサの他のそれぞれに直接に接続するコンピュータシステム。

【請求項１０】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサは、前記分散化ポイント・ツー・ポイント伝送インフラストラクチャによって相互接続された前記複数のプロセッサのうちの少なくとも１つを備えるコンピュータシステム。
【請求項１１】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサは、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャによって相互接続された前記複数のプロセッサから分離されているコンピュータシステム。
【請求項１２】
請求項１１に記載のコンピュータシステムであって、前記コンピュータシステムの初期化を促進するブートメモリをさらに備え、前記ブートメモリは、前記複数のプロセッサのうちの少なくとも１つの、前記少なくとも１つのパーティショニングプロセッサとしての動作を促進する、その中に記憶されたコンピュータプログラムインストラクションを有するコンピュータシステム。
【請求項１３】
請求項１に記載のコンピュータシステムであって、前記以前に特定されたパーティショニングスキーマは、前記コンピュータシステムの動作中に起こるイベントに応答して生成されるコンピュータシステム。
【請求項１４】
請求項１３に記載のコンピュータシステムであって、前記イベントは、前記コンピュータシステムの初期化、前記リソースのうちの少なくとも１つの故障、前記リソースのうちの少なくとも１つと関連付けられた動作負荷の変化、時間の経過、特定のソフトウェアの使用、および利用可能な電源リソースの変化のうちの１つを含むコンピュータシステム。
【請求項１５】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサをユーザインタフェースに接続する少なくとも１つのパーティショニングプロセッサリンクをさらに備え、前記以前に特定されたパーティショニングスキーマは、前記コンピュータシステムのユーザによって前記ユーザインタフェースおよび前記少なくとも１つのパーティショニングプロセッサリンクを介して特定されるコンピュータシステム。
【請求項１６】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサは、前記コンピュータシステムの初期化のときに前記ルーティングテーブルを生成するよう動作可能であるコンピュータシステム。
【請求項１７】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサは、前記コンピュータシステムの動作中に前記ルーティングテーブル群のうちの前記少なくとも１つのルーティングテーブルを改変するよう動作可能であるコンピュータシステム。
【請求項１８】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティションは、複数のパーティション群を備えるコンピュータシステム。
【請求項１９】
請求項１８に記載のコンピュータシステムであって、前記複数のパーティションのうちの少なくとも１つは、前記複数のリソースの機能的サブセットを備えるコンピュータシステム。
【請求項２０】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティション

は、前記複数のリソースのうちの全ての動作中のものを含む単一のパーティションを備えるコンピュータシステム。
【請求項２１】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサは、１つのパーティショニングプロセッサを備えるコンピュータシステム。
【請求項２２】
請求項１に記載のコンピュータシステムであって、前記少なくとも１つのパーティショニングプロセッサは、１つより多いパーティショニングプロセッサを備えるコンピュータシステム。
【請求項２３】
複数のプロセッサおよび前記複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャを含む複数のリソースを有するコンピュータシステムにおいて用いられるコンピュータによって実現される方法であって、前記方法は、
前記複数のリソースを少なくとも１つのパーティションにコンフィギャすることを含みえ、それぞれのパーティションは、前記複数のリソースのサブセットを備え、前記リソースの前記コンフィギャすることは、以前に特定されたパーティショニングスキーマにしたがって前記プロセッサに関連付けられた複数のルーティングテーブルのうちの少なくとも１つに書き込むことによって実現され、それぞれのルーティングテーブルは関連付けられたプロセッサおよび前記複数のプロセッサのうちの他のプロセッサの間のリンクを表し、前記リンクは前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する方法。
【請求項２４】
請求項２３に記載の方法であって、前記複数のリソースは、Ｉ／Ｏスイッチを含み、関連付けられた前記ルーティングテーブルのうちの１つを持つ前記Ｉ／Ｏスイッチは、前記Ｉ／Ｏスイッチ、前記プロセッサのうちの少なくとも１つ、および少なくとも１つのＩ／Ｏリソースの間のリンクを表す方法。
【請求項２５】
請求項２４に記載の方法であって、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャは非コヒーレントハイパートランスポート（ｎｃＨＴ）インフラストラクチャを備える方法。
【請求項２６】
請求項２３に記載の方法であって、前記複数のリソースをコンフィギャすることは、前記分散化ポイント・ツー・ポイント伝送インフラストラクチャによって相互接続された前記複数のプロセッサのうちの少なくとも１つを備える少なくとも１つのパーティショニングプロセッサを用いて達成される方法。
【請求項２７】
請求項２３に記載の方法であって、前記複数のリソースをコンフィギャすることは、前記分散化されたポイント・ツー・ポイント伝送インフラストラクチャによって相互接続された前記複数のプロセッサから分離されている少なくとも１つのパーティショニングプロセッサを用いて達成される方法。
【請求項２８】
請求項２３に記載の方法であって、前記コンピュータシステムの動作中に起こるイベントに応答して前記以前に特定されたパーティショニングスキーマを生成することをさらに含む方法。
【請求項２９】
請求項２８に記載の方法であって、前記イベントは、前記コンピュータシステムの初期化、前記リソースのうちの少なくとも１つの故障、前記リソースのうちの少なくとも１つと関連付けられた動作負荷の変化、時間の経過、特定のソフトウェアの使用、および利用可能な電源リソースの変化のうちの１つを含む方法。

【請求項３０】
請求項２３に記載の方法であって、前記コンピュータシステムのユーザによって特定された前記以前に特定されたパーティショニングスキーマを受け取ることをさらに含む方法。
【請求項３１】
請求項２３に記載の方法であって、前記複数のルーティングテーブルのうちの前記少なくとも１つに書き込むことは、前記コンピュータシステムの初期化のときに前記複数のルーティングテーブルを生成することを含む方法。
【請求項３２】
請求項２３に記載の方法であって、前記複数のルーティングテーブルのうちの前記少なくとも１つに書き込むことは、前記コンピュータシステムの動作中に前記ルーティングテーブル群のうちの前記少なくとも１つのルーティングテーブルを改変することを含む方法。
【請求項３３】
請求項１に記載の方法であって、前記少なくとも１つのパーティションは、複数のパーティション群を備える方法。
【請求項３４】
請求項３３に記載の方法であって、前記複数のパーティションのうちの少なくとも１つは、前記複数のリソースの機能的サブセットを備える方法。
【請求項３５】
請求項２３に記載の方法であって、前記少なくとも１つのパーティションは、前記複数のリソースのうちの全ての動作中のものを含む単一のパーティションを備える方法。
【請求項３６】
複数のプロセッサを含む複数のリソース、
前記複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャ、および
前記複数のリソースを少なくとも１つのパーティションにコンフィギャする少なくとも１つのパーティショニングプロセッサであって、それぞれのパーティションは、前記複数のリソースのサブセットを備え、前記少なくとも１つのパーティショニングプロセッサは、以前に特定されたパーティショニングスキーマにしたがって前記複数のプロセッサのうちの少なくとも１つと前記複数のプロセッサのうちの少なくとも１つの他のものとの間の少なくとも１つのリンクをイネーブルすることによって前記リソースをコンフィギャするよう動作可能であり、前記少なくとも１つのリンクは前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する、少なくとも１つのパーティショニングプロセッサを備えるコンピュータシステム。
【請求項３７】
請求項３６に記載のコンピュータシステムであって、前記少なくとも１つのリンクをイネーブルすることは、前記以前に特定されたパーティショニングスキーマにしたがって前記プロセッサに関連付けられた複数のルーティングテーブルのうちの少なくとも１つに書き込むことを含むコンピュータシステム。
【請求項３８】
請求項３６に記載のコンピュータシステムであって、前記少なくとも１つのリンクをイネーブルすることは、前記以前に特定されたパーティショニングスキーマにしたがって前記少なくとも１つのリンクに関連付けられた少なくとも１つのスイッチを閉じることを含むコンピュータシステム。
【請求項３９】
複数のプロセッサおよび前記複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャを含む複数のリソースを有するコンピュータシステムにおいて用いられるコンピュータによって実現される方法であって、前記方法は、
前記複数のリソースを少なくとも１つのパーティションにコンフィギャすることを含みえ、それぞれのパーティションは、前記複数のリソースのサブセットを備え、前記リソースの前記コンフィギャすることは、以前に特定されたパーティショニングスキーマにしたが

って前記複数のプロセッサのうちの少なくとも１つと前記複数のプロセッサのうちの少なくとも１つの他のものとの間の少なくとも１つのリンクをイネーブルすることによって実現され、それぞれのルーティングテーブルは関連付けられたプロセッサおよび前記複数のプロセッサのうちの他のプロセッサの間のリンクを表し、前記リンクは前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する方法。

【請求項４０】

請求項３９に記載の方法であって、前記少なくとも１つのリンクをイネーブルすることは、前記以前に特定されたパーティショニングスキーマにしたがって前記プロセッサに関連付けられた複数のルーティングテーブルのうちの少なくとも１つに書き込むことを含む方法。

【請求項４１】

請求項３９に記載の方法であって、前記少なくとも１つのリンクをイネーブルすることは、前記以前に特定されたパーティショニングスキーマにしたがって前記少なくとも１つのリンクに関連付けられた少なくとも１つのスイッチを閉じることを含む方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、大きくはコンピュータシステムにおけるリソースのパーティショニングに関する。より具体的には本発明は、そのようなリソースを従来に可能であったよりもより正確にパーティショニングする技術を提供する。

【背景技術】

【０００２】

リソースを分散化コンピューティングシステムにおいて、または単一のコンピュータシステムの中央電子装置群（ＣＥＣ）内においてパーティショニングする基本的なアイディアは、新しいものではない。しかし従来のパーティショニング技術にはシステムリソースを比較的粗いレベルで、例えば物理的に別個のマシン、プロセッサのグループなどにアロケートすることだけが可能なものもあったが、他のものは、例えばシステムの初期化においてだけマルチプロセッサグループ中のプロセッサ群のサブセットを永久にあるパーティションに割り当てるなど、柔軟性のない状態で動作する。

【０００３】

従来のパーティショニング技術のいくつかの欠点は図１を参照して理解されるだろう。図１は、ハブ１０６および１０８を介して相互接続された２つの４プロセッサグループ（すなわちクワッド）１０２および１０４を持つ従来の８プロセッサシステム１００を示す。従来のクワッド中のプロセッサ１１０のそれぞれは、ブロードキャストバス１１２に接続され、このバスはプロセッサのいずれをも他のプロセッサからアイソレートしない、つまり任意のプロセッサによる特定のリソース（例えばＩ／Ｏ１１４またはメモリ１１６）へのアクセス要求はいずれも他のプロセッサの全てによって受け取られる。よってこのようなシステムおいて起こる唯一のポイント・ツー・ポイント通信は、２つのクワッド間を接続するハブ間に行われるので、このようなシステムのパーティショニングはハブ間でだけしか行われない、つまり１つのパーティションに１クワッド、他のもう１つのパーティションにもう１つのクワッドという形になる。

【０００４】

マルチプロセッサシステムの設計の比較的新しいアプローチは、プロセッサ間のブロードキャスト通信をポイント・ツー・ポイントデータ転送メカニズムに置き換える。このポイント・ツー・ポイントデータ転送おいては、プロセッサはネットワークノードと同様に分散化コンピューティングシステム内、例えばワイドエリアネットワーク内で通信を行う。すなわちプロセッサは複数の通信リンクを介して相互接続され、リクエストは、それぞれのプロセッサに関連付けられたルーティングテーブルにしたがってリンクにわたってプロセッサ間を転送される。この意図するところは、単位時間当たりにマルチプロセッサプラットフォーム内を伝送される情報量を増やすことにある。このアプローチの例は、カリフ

ォルニア州サニーベールのアドバンスト・マイクロ・デバイス社（ＡＭＤ）によって最近アナウンスされたハイパートランスポート（ＨＴ）アーキテクチャであり、例えば、カリフォルニア州、アナハイムにおける２００１年３月２６～２８日のWinHEC 2001において発表されたHyperTransport™ Technology: A High-Bandwidth, Low-Complexity Bus Architectureと題された論文に記載され、その全体が全ての目的のためにここで参照によって援用される。ＡＭＤアーキテクチャの動作は図２の簡略化されたブロック図を参照して簡単に説明される。

【０００５】

システム２００がブートアップするたびに、プロセッサ２０２ａ～２０２ｄのそれぞれに関連付けられたルーティングテーブルは初期化されなければならない。残りのプロセッサ群がまだ動作しておらず、ＨＴインフラストラクチャを用いてそれぞれのプロセッサ内のルーティングテーブルを構築する「ディスカバリ」アルゴリズムを実行するあいだに、プライマリプロセッサ２０２ａ、すなわちＢＩＯＳ２０４と通信するプロセッサはスタートアップする。このアルゴリズムは「貪欲な」アルゴリズムであり、利用可能、つまり「到達可能な」システムリソースの全てをキャプチャし、単一の分割されないシステムをキャプチャされたリソースから構築しようと試みる。プライマリプロセッサ２０２ａは、それに応答する全てのシステムリソース、すなわち全ての隣接装置を識別することによってこのプロセスを開始する。それからそれぞれのこのようなリソース（例えばメモリバンク２０６ａおよびプロセッサ２０２ｂおよび２０２ｃ）について、プライマリプロセッサ２０２ａは、そのリソース（例えばメモリバンク２０６ｂおよび２０６ｃ）に応答する（すなわち「所有されている」）全てのリソースを要求する。この情報を用いて、プライマリプロセッサはルーティングテーブルを構築する。残念ながらこのようなシステムによって生成されたルーティングテーブルは、リソース故障、負荷の不均衡、またはシステムリソースの再アロケーションが望ましい任意の状況変化にかかわらず、システム動作中は変化しない。

【発明の開示】

【発明が解決しようとする課題】

【０００６】

上記を鑑みて、コンピュータシステムリソースがより柔軟に正確にパーティショニングされる技術を提供することが望ましい。

【課題を解決するための手段】

【０００７】

本発明によれば、単一プラットフォーム、マルチプロセッサシステムにおけるプロセッサ間の分散化されたポイント・ツー・ポイント通信によって代表される革新が利用されることによって、コンピュータシステムリソースのパーティショニングにおいて従来可能であったよりもより高いレベルの柔軟性および正確さを提供する。すなわち分散化されたポイント・ツー・ポイント通信インフラストラクチャが採用されることで、以前に特定されたスキーマにしたがってルーティングテーブルを生成することを通してシステムリソースを正確にアロケートすることが可能となる。本発明のさまざまな実施形態によれば、このようなパーティショニングは、システムリソースの任意の所望のアロケーションを達成するために、静的に、例えばシステムのスタートアップ時に、あるいは動的に、例えばある種のイベントに応答してなされえる。

【０００８】

すなわち本発明は、複数のプロセッサを含む複数のリソース、前記複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャ、および前記複数のリソースを少なくとも１つのパーティションにコンフィギャする少なくとも１つのパーティショニングプロセッサを有するコンピュータシステムを提供する。それぞれのパーティションは、前記複数のリソースのサブセットを備える。前記少なくとも１つのパーティショニングプロセッサは、以前に特定されたパーティショニングスキーマにしたがって前記複数のプロセッサのうちの少なくとも１つと前記複数のプロセッサのうちの少な

くとも１つの他のものとの間の少なくとも１つのリンクをイネーブルすることによって前記リソースをコンフィギャするよう動作可能である。そのようにイネーブルされたリンク（群）は、前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する。本発明の具体的な実施形態によれば、前記リンクを前記イネーブルすることは、前記以前に特定されたパーティショニングスキーマにしたがって前記プロセッサに関連付けられた複数のルーティングテーブルのうちの少なくとも１つに書き込むことによって実現される。

【０００９】

このようなコンピュータシステムをパーティショニングする方法も記載される。具体的な実施形態によれば、前記複数のリソースは、少なくとも１つのパーティションにコンフィギャされる。それぞれのパーティションは、前記複数のリソースのサブセットを備える。前記リソースの前記コンフィギャすることは、以前に特定されたパーティショニングスキーマにしたがって前記複数のプロセッサのうちの少なくとも１つと前記複数のプロセッサのうちの少なくとも１つの他のものとの間の少なくとも１つのリンクをイネーブルすることによって実現される。このようにイネーブルされた少なくとも１つのリンクは、前記ポイント・ツー・ポイント伝送インフラストラクチャの部分に対応する。本発明の具体的な実施形態によれば、前記少なくとも１つのリンクを前記イネーブルすることは、前記以前に特定されたパーティショニングスキーマにしたがって前記プロセッサに関連付けられた複数のルーティングテーブルのうちの少なくとも１つに書き込むことによって実現される。

本発明の性質および優位性のさらなる理解は、明細書および図面の残りの部分を参照して達成されよう。

【発明を実施するための最良の形態】

【００１０】

本発明の具体的な実施形態が、例えばＡＭＤのハイパートランスポートマルチプロセッサアーキテクチャのような特定のポイント・ツー・ポイント通信インフラストラクチャを参照してこれから説明される。しかし本発明の範囲は、説明された特定の実施形態やＡＭＤアーキテクチャに限定されないことが理解されよう。すなわち、そのプロセッサおよび／またはさまざまな他のシステムリソース内でポイント・ツー・ポイント通信を行う任意のマルチプロセッサアーキテクチャが本発明の具体的な実施形態を実現するのに適する。

【００１１】

図３は、本発明の具体的な実施形態に基づいて設計されたサーバ３００の簡略化されたブロック図を示す。サーバ３００は、実質的に同一のプロセッサ３０２ａ～３０２ｄ、基本Ｉ／Ｏシステム（ＢＩＯＳ）３０４、メモリバンク３０６ａ～３０６ｄを備えるメモリサブシステム、プロセッサ３０２ａ～３０２ｄおよびＩ／Ｏスイッチ３１０を相互接続するためのポイント・ツー・ポイント通信リンク３０８ａ～３０８ｅ、およびリンク３１４ａ～３１４ｅによって図３で示されるＪＴＡＧインタフェースを介してプロセッサ３０２ａ～３０２ｄおよびＩ／Ｏスイッチ３１０と通信するサービスプロセッサ３１２を含む。Ｉ／Ｏスイッチ３１０はシステムの残りをＩ／Ｏアダプタ３１６および３２０に接続する。

【００１２】

具体的な実施形態によれば、本発明のサービスプロセッサは、上述の「貪欲な」アルゴリズムの使用と対照的に、以前に特定されたパーティショニングスキーマによってシステムリソースをパーティショニングするインテリジェンスを有する。これは、システムプロセッサに関連付けられたルーティングテーブルをサービスプロセッサによって直接に操作することを通して達成され、これはポイント・ツー・ポイント通信インフラストラクチャによって可能になる。ルーティングテーブルは、システムリソースを制御および分離するのに用いられ、システムリソース間の接続関係がこの中で定義される。

【００１３】

具体的な実施形態によれば、サービスプロセッサは、それ自身のオペレーティングシステムアプリケーションのセット（システムの残りと関連付けられたオペレーティングシステ

ム（群）とは別個である）およびそれ自身のＩ／Ｏを持つ自律プロセッサである。このプロセッサは、残りのプロセッサ、メモリ、およびＩ／Ｏが機能していないときでも動作する。このプロセッサは、全てのシステムリソースが所望のように動作していることを確実にする外部管理インテリジェンスとして動作する。

【００１４】

しかし以前に特定されたパーティショニングスキーマは、分離されたサービスプロセッサによってインプリメントされなければならないわけではないことに注意されたい。すなわち例えば、システムプロセッサのうちの１つがこの目的のために用いられてもよい。このような実施形態によれば、例えばシステムＢＩＯＳがシステムのプライマリプロセッサを用いてスキーマを有効にするよう変更されえる。

【００１５】

さらに本発明のさまざまな実施形態によれば、パーティションは、さまざまなシステムリソースの組み合わせを表しえる。すなわち例えば、「キャパシティオンデマンド」のシナリオにおいては、パーティションは単一のプロセッサによって表されえ、この場合そのパーティションからプロセッサを取り除くことによって残りの要素はＯＳを走らせることができなくなる（よってユーザはこのパーティションについてはチャージされない）。パーティションはまたプロセッサおよびいくつかの関連するメモリまたはＩ／Ｏによって表される。大きくは、コンピュータシステム中で利用可能なリソースの任意の機能サブセットがパーティションとして考えられえる。

【００１６】

より大きくは、図３に示される特定のアーキテクチャは単に例示的であり、本発明の実施形態は、異なる構成およびリソースの相互接続、および示されたシステムリソースのそれぞれについてのさまざまな代替物を有すると意図されることが理解されよう。しかし図示の目的のため、サーバ３００の具体的な詳細が想定される。例えば図３に示されるリソースのほとんどは単一の電子的アセンブリ上に常駐すると想定される。さらにメモリバンク３０６ａ～３０６ｄは、デュアルインラインメモリモジュール（ＤＩＭＭ）として物理的には提供されるダブルデータレート（ＤＤＲ）メモリを備えてもよい。Ｉ／Ｏアダプタ３１６は例えば、永久記憶装置へのアクセスを提供するウルトラダイレクトメモリアクセス（ＵＤＭＡ）コントローラまたはスモールコンピュータシステムインタフェース（ＳＣＳＩ）コントローラでありえる。Ｉ／Ｏアダプタ３２０は、例えばローカルエリアネットワーク（ＬＡＮ）またはインターネットのようなネットワークとの通信を提供するよう構成されるイーサネットカードでありえる。

【００１７】

具体的な実施形態によれば図３に示されるように、Ｉ／Ｏアダプタ３１６および３２０の両方は対称的なＩ／Ｏアクセスを提供する。すなわちそれぞれがＩ／Ｏの対等なセットへのアクセスを提供する。理解されるようにこのような構成は、複数のパーティションが同じタイプのＩ／Ｏにアクセスするパーティショニングスキームを促進する。しかし実施形態によってはＩ／Ｏなしのパーティションが作られることも想定される。例えば、１つ以上のプロセッサおよび関連付けられたメモリリソース、すなわちメモリ複合体（memory complex）を含むパーティションは、そのメモリ複合体をテストする目的で作られえる。

【００１８】

ある実施形態によれば、サービスプロセッサ３１２は集積されたチップセット機能を持つＰＰＣコントローラであるモトローラMPC855Tである。

【００１９】

サービスプロセッサ３１２は、サーバ３００のリソースをパーティショニングすることに主に責任を負う。図３に示される実施形態によれば、サービスプロセッサ３１２は、プロセッサ３０２ａ～３０２ｄおよびＩ／Ｏスイッチ３１０の利用をアロケートするが、例えばメモリバンクまたはさまざまなＩ／Ｏデバイスのような他のリソースを直接管理するようプログラムされてもよい。サービスプロセッサ３１２の構成は、通信リンク３２２を介してサーバプロセッサ３１２が接続される管理／サーバコンソール（不図示）を介して有

効にされえる。
【００２０】
図４は、ある実現例によるサービスプロセッサ３１２のための相互接続の高レベルブロック図である。本発明のパーティショニングエンジンは、図３～５に示されるようなサービスプロセッサ３１２とは非常に異なって見えうることに注意することが重要である。すなわちポイント・ツー・ポイント通信インフラストラクチャを用いてルーティングテーブルをインテリジェントに再構成できる任意のメカニズムは本発明の範囲内である。例えば他のありえるメカニズムとしては、パーティショニングを行うために１つ以上のプロセッサ３０２を用いることも含まれる。
【００２１】
しかしこの実施形態においては、サービスプロセッサ３１２は、ＤＲＡＭ記憶ブロック４０２およびフラッシュメモリ４０４に直接接続される。ＤＲＡＭ４０２は、フラッシュ４０４内に記憶されたプログラムのサービスプロセッサによる実行を促進する。サービスプロセッサ３１２はまたＰＣＩバス４０６を介してセンサインタフェース４０８、イーサネットカード４１０、およびＪＴＡＧインタフェース４１２に接続される。センサインタフェース４０８は例えば、温度、電源電圧、またはセキュリティロックの表示を行うモニタリング回路（不図示）からの入力を含みえる。センサインタフェース４０８はまた、例えばシステムのファンを起動する制御信号のようなさまざまな出力を含みえる。イーサネットカード４１０は、サービスプロセッサ３１２と、例えばネットワークアドミニストレータがサーバをモニタおよびコンフィギャするサービスコンソールとの間のインタフェースを提供する。
【００２２】
具体的な実施形態によれば、インタフェース４１２は、テストアクセスポートおよびバウンダリスキャンアーキテクチャの要件を記載するジョイントテストアクショングループ（ＪＴＡＧ）標準として知られるＩＥＥＥ標準１１４９．１を完全にサポートする。テストアクセスポート（ＴＡＰ）は、テストデータイン（ＴＤＩ）ピン、テストデータアウト（ＴＤＯ）ピン、テストクロック（ＴＣＫ）ピン、テストモードセレクト（ＴＭＳ）ピン、およびオプションとしてＴＡＰコントローラをテストロジックリセットの状態に設定するテストリセット（ＴＲＳＴ）ピンを含むいくつかのピンを備える。ＴＡＰコントローラは、集積回路上のバウンダリスキャンを制御する１６ステートの有限ステートマシンである。詳細が後述されるように、ＪＴＡＧインタフェース４１２は、サービスプロセッサ３１２およびプロセッサ３０２ａ～３０２ｄの間の通信を促進し、それによりコンピュータシステムのリソースの静的および動的パーティショニングの両方を可能にする。具体的な実施形態によれば、この通信は簡単なアウトバウンドマルチプレクサを用いて促進される。
【００２３】
再び図３を参照し本発明の具体的な実施形態によれば、プロセッサ３０２ａ～３０２ｄはＨＴインフラストラクチャとして知られここでもそのように呼ぶポイント・ツー・ポイントデータ転送メカニズムを採用するＡＭＤ　Ｋ８プロセッサを備えてもよい。すなわちプロセッサは複数のポイント・ツー・ポイント通信リンク（すなわち３０８ａ～３０８ｅ）を介して相互接続され、リクエストはそれぞれのプロセッサに関連付けられたルーティングテーブルにしたがってプロセッサ間でリンクを通して転送される。ＨＴインフラストラクチャは、例えば、コンピュータシステム中の周辺機器相互接続（ＰＣＩ）デバイス、アクセラレーテッドグラフィックポート（ＡＧＰ）デバイス、ダイナミックランダムアクセスメモリ（ＤＲＡＭ）デバイス、および他の専用高バンド幅バスのようなさまざまなコンピュータリソースの相互接続を可能にする。
【００２４】
ＨＴには２つの形態がある。ノンコヒーレントな１つの形態、つまりｎｃＨＴは、Ｉ／Ｏリンクのために開発され、図３に示される（３０８ｃおよび３０８ｄ）。ここで説明されるコヒーレントＨＴ、つまりｃＨＴメカニズムは、遠くのメモリ、すなわちリモートクラスタにおけるプロセッサに接続されたメモリ同様、ローカルなメモリ、すなわちそのクラ

スタ内のプロセッサに接続されたメモリにプロセッサアレイがアクセスできる非均一なメモリアクセスマトリクスのためにＡＭＤによって開発されてきた。

【００２５】

本発明のさまざまな実施形態によれば、プロセッサ３０２ａ～３０２ｄは実質的に同一である。図５は、そのようなプロセッサ３０２の簡略化されたブロック図であり、プロセッサ３０２は、複数のＨＴポート５０４ａ～５０４ｃを持つＨＴインタフェース５０２およびそれらに関連付けられたルーティングテーブル５０６ａ～５０６ｃを含む。それぞれのＨＴポートは、コンピュータシステムにおいて１６ビットのＨＴリンク、例えば図３のリンク３０８ａ～３０８ｅを介して、他のリソース、例えばプロセッサまたはＩ／Ｏデバイスとの通信を可能にする。

【００２６】

図５に示されるｃＨＴインフラストラクチャは、ポイント・ツー・ポイントの分散化されたルーティングメカニズムとして一般化されえ、このメカニズムは、任意のさまざまなトポロジ、例えばリング、メッシュなどによってシステムプロセッサを相互接続する複数のセグメント（例えばゴーズ・イン・ゴーズ・アウト（ＧＩＧＯ）バス）を備える。これらセグメントのそれぞれのエンドポイントのそれぞれは、ユニークなノードＩＤを持つ接続されたプロセッサ、およびそれが「所有する」複数の関連付けられたリソースに関連付けられ、例えばそれに接続されるメモリがどれだけであるか、それに接続されるＩ／Ｏは何であるかなどが関連付けられる。

【００２７】

分散化されたルーティングメカニズム中のノードのそれぞれに関連付けられたルーティングテーブル群は、コンピュータシステムリソース間の相互接続の現在の状態を全体として表す。ある与えられたノード（例えばプロセッサ）によって所有されたリソースのそれぞれ（例えば特定のメモリ範囲またはＩ／Ｏデバイス）は、そのノードに関連付けられたルーティングテーブル（群）においてアドレスとして表現される。リクエストがノードに到達するとき、リクエストされたアドレスは、適切なノードおよびリンクを特定するノードのルーティングテーブル中の２レベルエントリと比較され、このルーティングテーブルは例えば、もしこのアドレスが要るならノードｘに行け、もしノードｘに行きたいならリンクｙを用いろ、などが特定される。

【００２８】

ルーティングテーブルに関連付けられたものとしては、これらテーブルが初期化または変更されるときに用いられるコントロールがある。これらコントロールは、システムが正しく動作することを続けながらもルーティングテーブル中の値を、原子性を保証しながら（atomically）変更する手段を提供する。

【００２９】

図５に示されるように、プロセッサ３０２は、関連するルーティングテーブル中の情報にしたがってポイント・ツー・ポイント通信を３つの他のプロセッサと行うことができる。具体的な実施形態によれば、ルーティングテーブル５０６ａ～５０６ｃは、２レベルのテーブルを備え、第１レベルはシステムリソース（例えばメモリバンク）のユニークなアドレスを対応するＨＴノード（例えばプロセッサのうちの１つ）に関連付け、第２レベルはそれぞれのＨＴノードを、現在のノードから当該ノードに到達するために利用されるべきリンク（例えば３０８ａ～３０８ｅ）に関連付ける。

【００３０】

プロセッサ３０２はまた、ＪＴＡＧハンドシェークレジスタ群５０８のセットを持ち、これらはとりわけサービスプロセッサおよびプロセッサ３０２の間の通信を促進する。すなわちサービスプロセッサは、ルーティングテーブル５０６ａ～５０６ｃに結果として格納されるように、ルーティングテーブルエントリをハンドシェークレジスタ５０８に書き込む。図５に示されるプロセッサアーキテクチャは本発明の具体的な実施形態を説明する目的の例示的なものに過ぎないことを理解されたい。本発明の他の実施形態を実現するためには、例えばより少ないまたはより多い数のポートおよび／またはルーティングテーブル

が用いられてもよい。
【００３１】
具体的な実施形態によれば図３のサーバ３００は、ここで説明される技術を用いて単一の４プロセッサシステムとして、または２つ以上の機能的に別個のパーティション群として動作するようコンフィギャされえる。結果としてできあがるシステムコンフィギュレーションのアプリオリの知識なしで動作するＨＴインフラストラクチャのデザイナによって想定される「貪欲な（greedy）」アルゴリズムとは対照的に、サービスプロセッサ３１２は、先に具体的に述べられたパターニングスキーマによって全てのまたはいくつかのプロセッサ３０２ａ～３０２ｄ（およびＩ／Ｏスイッチ３１０）に関連付けられたルーティングテーブルを生成および／またはダイナミックに変更することによってサーバ３００のコンフィギュレーションを促進する。これは、ルーティングテーブルエントリを適切なプロセッサのＪＴＡＧハンドシェークレジスタ５０８（およびＩ／Ｏスイッチ３１０に関連付けられた同様のテーブル）に書き込むサービスプロセッサ３１２によって達成される。後で説明されるように、このシステムコンフィギュレーション／パーティショニングは、静的に、例えばサーバブートアップ時に、または動的に、例えばサーバ３００の動作中に行われえる。
【００３２】
本発明は、本発明にしたがって設計されたコンピュータシステム、例えばサーバが動的に再コンフィギャされる（例えば再パーティショニングされる）実施形態をも包含する。すなわち再コンフィギュレーションは、システムをリブートすることなく、またはサービスプロセッサの再イニシャライズ動作を行うことなく達成され、例えばシステムリソースの故障のようなあるランタイムイベントに応答して起こる。
【００３３】
大きくは、本発明によるシステムリソースの動的パーティショニングは、さまざまな種類のランタイムイベントに応答して起こりえることを理解されたい。動的パーティショニングを起動しえるイベントまたはトリガの具体的な例には、現在存在するパーティションに関連付けられたリソース（例えばプロセッサ）の故障、所定期間（例えば特定のシステムリソースのメンテナンス期間）の満了、負荷分布を変える特定のアプリケーションまたはアプリケーション群の組み合わせを走らせること、およびパーティションを再コンフィギャする旨のユーザの命令が含まれる。
【００３４】
このようなユーザの命令は、例えば、さまざまな条件についての負荷評価に関する実験、特定のアプリケーションのサポートのためのリソースの再アロケーション、一日の期間にわたっての既知の負荷変動を扱うためのリソースの再アロケーション、またはユーザまたはユーザのグループとの合意に基づくシステムリソースの再アロケーションでありえる。後者のシナリオにおいてシステムユーザたちは、彼等の使用のために彼等にアロケートされたリソースにしたがって、すなわちキャパシティオンデマンドで、しかし従来可能であったよりも粒状性のより細かいリソースレベルでチャージされえる。これは、複数のプロセッサを相互接続する分散化されたポイント・ツー・ポイント伝送インフラストラクチャ、例えばＨＴインフラストラクチャを採用することは、例えば単一のプロセッサ、メモリバンク、またはＩ／Ｏバスのアロケーションを可能にするからである。
【００３５】
本発明の技術による静的および動的の両方のシステムコンフィギュレーション／パーティショニングが達成される特定の実施形態がこれから説明される。ここで用いられるように、静的なパーティショニングという語は、そのコンピュータシステムに関連付けられた１つ以上のオペレーティングシステムをシャットダウンして、パーティショニングを行い、１つ以上のオペレーティングシステムを再起動することを指す。対照的に、動的パーティショニングは、パーティショニングを行うために１つ以上の現在走っているオペレーティングシステムについて処理をすることを指す。
【００３６】

例えば本発明のさまざまな実施形態によれば、サービスプロセッサは現在存在するパーティションからリソースを除去するためのリクエストを出すことができ、このリクエストに応答して、影響を受けるオペレーティングシステム（群）はシャットダウンすることなくリソース除去を扱えるかどうかを決定する。もし可能であれば、サービスプロセッサはここに記載されたように適切なルーティングテーブルを再コンフィギャすることが許される。もしそうでなければ、いかなるパーティショニングも静的になされなければならない。

【００３７】

もし一方で、利用可能なリソースを現在存在するパーティションに追加するリクエストであれば、サービスプロセッサは、影響を受けるオペレーティングシステム（群）に対してリソースを追加する要求を出す。影響されるオペレーティングシステム（群）は、それから動作を止めることなくリソースを追加することが可能でありえる。

【００３８】

今度は図６を参照する。電源がシステム（６０２）に与えられると、サービスプロセッサは、以前に特定されたコンフィギュレーションにしたがって、システムプロセッサに関連付けられたルーティングテーブルを生成する（６０４）ことによって、システムプロセッサのそれぞれを初期化する。すなわち静的パーティショニングが実行される。初期のシステムコンフィギュレーションは、例えばサービスプロセッサが接続されるフラッシュメモリに記憶されるか、またはシステムをオンラインにするシステムアドミニストレータによって特定されえる。システムコンフィギュレーションは、幅広いオペレーション上の目的を達成するために１つ以上の機能的に別個のパーティションに対応しえる。いったんルーティングテーブルが生成されると、システムオペレーションが開始し、ここでシステムリソースは１つ以上の機能的に別個のパーティションにアロケートされる（６０６）。

【００３９】

サービスプロセッサはそれから、システムの再パーティショニングを必要とするランタイムイベントの発生がないかをチェックするため進行中のサーバアクティビティを監視し続ける（６０８）。上述のように本発明は、このような幅広いランタイムイベントが動的パーティショニングをトリガするのに適すると想定する。しかし説明の目的のために、システムリソースの故障がこの例示的な実施形態においては用いられる。よって６０８においてサービスプロセッサは、このような故障に関するシステムエラーメッセージを探す。もしこのようなエラーメッセージが検出されると（６１０）、メッセージはエラーの重大性を決定するために分析される（６１２）。もしエラーが破局的であるなら（６１４）、すなわちシステムが信頼性をもって動作しえないなら、リモートファシリティに知らせが行き（６１６）、システムはシャットダウンされる（６１８）。

【００４０】

もし一方で、エラーが破局的でないなら（６１４）、故障した要素は分離され（６２０）、故障した要素（群）が結果として生じる機能パーティション群のいずれからも除去される新しいシステムコンフィギュレーションが導かれる（６２２）。実施形態によっては、この新しいシステムコンフィギュレーションの導出は、現在のシステムコンフィギュレーション、例えばサービスプロセッサに関連付けられたフラッシュメモリに記憶されたコンフィギュレーション、および故障情報を参照して行われる。サービスプロセッサはそれから、適切なシステムリソースを静止状態にし（６２４）、システムノード、例えばプロセッサに関連付けられたルーティングテーブルおよびコントロールに適切な変更を施す（６２６）ことによって新しいパーティショニングスキームを有効にする。

【００４１】

本発明の具体的な実施形態によれば、サービスプロセッサは、任意の故障したプロセッサを自動的に静止状態にさせ、それによってそれを利用可能なプロセッサプールから除去するようコンフィギャされる。対応するコマンドはサービスプロセッサからオペレーティングシステムへとアドバンストコンフィギュレーション・パワーインタフェース（ＡＣＰＩ）２．０プロトコルを用いて送られえる。故障したプロセッサにアサインされたスレッドはすでに完了しており（もし可能であれば）、いったんオペレーティングシステムが静止

状態になると、アクノリッジがサービスプロセッサに送られ、サービスプロセッサはこんどは故障したプロセッサをそのパーティションから取り除く。サービスプロセッサは、プロセッサ群のうちの他の１つを故障したプロセッサパーティションにアサインしてもよく、縮小された状態にパーティションを維持してもよく、あるいはパーティションごとなくしてもよい。

【００４２】

故障した要素（群）が分離され、再パーティショニングが定義された後、新しく形成されたパーティションを適切に初期化するためにシステムをリスタートすることが必要かどうかについて判断がなされる（６２８）。もし必要であれば、システムリスタートが開始される（６３０）。もし必要でなければ、システム全体のリブートを必要とすることなく、ある特定のシステム要素が初期化を必要とするかどうかについて判断がなされる（６３２）。もし必要であれば、電源オンリセット信号が特定された要素（群）に送られる（６３４）。そうでなければサーバは、それから通常動作を続けることが許される。あるいはシステムリソースの故障によってしばしばシステムのリスタートが必要となると仮定するなら、そのような故障に応答してシステムが再パーティショニングされた後にはシステムリスタートが自動的に起こるような実施形態も想定される。

【００４３】

前述のことを参照し本発明の技術は、サービスプロセッサではなくシステムプロセッサ群のうちの１つによってシステム初期化が導かれるようなシステムにおいて、中央処理装置（ＣＰＵ）の電源オン時の故障を扱うためにも採用されえることが理解されよう。具体的な実施形態によれば本発明のサービスプロセッサは、そのルーティングテーブルをコンフィギャするために「アウトオブバンド」でシステムプロセッサと通信を行うので、コンピュータシステムは、プライマリシステムプロセッサが調子が悪くなり、セカンダリシステムプロセッサ（群）との初期化チェーン、すなわち「インバンド」通信を完了できないパーティションの故障に対してよりロバストに作られえる。このような場合、故障したプロセッサを除外し、新しいプライマリシステムプロセッサを指定し、中断した初期化を完了させるために、サービスプロセッサはプライマリプロセッサが故障したことを検出し、それからセカンダリシステムプロセッサ（群）のルーティングテーブルおよびコントロールがサービスプロセッサによって直接に変更されえる。

【００４４】

前述のように本発明は、コンピュータシステムのリソースの動的な再パーティショニングが起動されえるさまざまなメカニズムを想定する。あるそのようなメカニズムである、システムアドミニストレータによって入力されたランタイムパーティショニングリクエストが図７のフローチャートを参照してここで説明される。図示される実施形態は、例えばシステム管理コンソールからのパーティショニングリクエストを探して、進行中のサーバアクティビティを監視することから始まる（７０２）。もしそのようなリクエストが受け取られると（７０４）、サービスプロセッサは、リクエストが有効かどうかを判断する（７０６）。本発明のさまざまな実施形態によれば、パーティショニングリクエストの有効性は、任意の判断基準にしたがって決定されえる。例えばリクエストによって示されるパーティションは、有効なパーティションのための最低リソースリストを照らしてチェックされえる。もしリクエストが有効でないと判断されると（７０６）、リクエストは無効であり拒絶されることを示すレスポンスがコンソールに送られる（７０８）。さまざまな実施形態によれば、レスポンスは、リクエストがなぜ無効かを示す、例えばリクエストがブートイメージを含まないパーティションを作ろうとしたことを示す付加的な情報を示してもよい。

【００４５】

もし一方で、パーティショニングリクエストが有効であると判断されるなら（７０６）、サービスプロセッサは、新しいパーティションを有効にするために、適切なシステムリソース（例えばプロセッサ、メモリ、Ｉ／Ｏ）を静止状態にし（７１０）、静止状態にされたシステムリソースに関連付けられたルーティングテーブルに適切な変更を行う（７１２

）ことによって要求されたパーティショニングスキームを実現する。
【００４６】
システムパーティショニングが再定義された後で、新しく形成されたパーティションの適切な初期化のためにシステム全体をリスタートすることが必要かどうかについての判断がなされる（７１４）。もしそうならシステムがリスタートされる（７１６）。もし必要でないなら、特定のシステム要素（群）が初期化を必要とするかどうかについてのさらなる判断がなされる（７１８）。もし必要なら、それから電源オンリセット信号がそのような要素（群）に送られる（７２０）。再パーティショニングが完了した後で、進行中のサーバアクティビティのモニタリングが再開される（７０２）。よって本発明は、ランタイム中にサーバ（または任意のコンピュータシステム）を再パーティショニングすることを可能にする。
【００４７】
より一般には上述のように、このような再パーティショニングは、幅広い種類のランタイムイベントによって、幅広い目的のために起こりえる。例えばシステムアドミニストレータからのリクエストに応答する代わりに、システムリソースの再パーティショニングは、走らされている特定の組み合わせのアプリケーション群に応答して起こりえる。すなわちシステムは、あるレベルのリソースをそれぞれ必要とするアプリケーション群の第１の組み合わせを走らせるように最初はパーティショニングされえる。システムリソースの異なるアロケーションを要求する第２の組み合わせのアプリケーション群が走らされるとき、リソース群の新しいアロケーションを有効にするためのシステムの自動再パーティショニングが起こりえる。
【００４８】
本発明は特に具体的な実施形態について示され説明されてきたが、当業者には開示された実施形態の形態および詳細への変更が本発明の範囲から逸脱することなくなされえることが理解されよう。例えば本発明は、説明された実施形態から任意の数のプロセッサを持つ実現例へと一般化されえる。
【００４９】
さらに本発明の技術は、ここに説明された静的および動的パーティショニングを実行するためには必ずしも別個のサービスプロセッサを必要としない。すなわちシステムプロセッサのうちの１つが、これらの機能をシステムスタートアップ時において、またはランタイムイベントに応答して実行するように構成されえる。マルチプロセッサが協働してパーティショニングを実行するように構成される実施形態も想定される。これらの種類の動作を命令するプログラムは例えばシステムＢＩＯＳメモリに記憶されえる。
【００５０】
さらにシステム初期化において、およびシステム動作中においての両方で本発明の技術が採用される特定の実施形態がここでは説明されるが、本発明は、本発明によって可能になる技術がこれらのうちの一方か他方かにおいてだけ採用される実施形態も包含することが理解されよう。すなわち本発明がシステムを初期化するのにだけ用いられる実施形態も想定される。逆に、動的パーティショニングだけが実行される実施形態も想定される。
【００５１】
ここで使用される「リソース」という語は、単一のプロセッサの物理的な実現例よりももっと多くを包含するよう想定される。すなわち本発明のさまざまな実施形態によれば、リソースは、コンピューティングシステムの主要な要素群のうちの任意のものとして考えられえる。そのような要素群にはいくつかを挙げれば、１つ以上のプロセッサ、メモリのバンク（典型的には物理的なメモリ、例えばＤＩＭＭのバンク）、Ｉ／Ｏバス（例えばＰＣＩまたはＩＳＡバス）、およびバス上の要素（例えばＳＣＳＩカード、通信カード、ファイバーチャネルカード）のような物理的要素群が含まれる。しかしシステムリソースは、プロセッサに関連付けられた論理部品のような論理要素（例えばＤＭＡエンジンまたはルーティング割り込みのための割り込みメカニズム）でありえることに注意されたい。
【００５２】

また本発明は、パーティショニングスキームを実現するためにルーティングテーブルが操作される実施形態に限定されるべきではないことに注意されたい。すなわち本発明は、システムプロセッサ間のリンク群をイネーブルしたり、ディセーブルしたりすることによってパーティショニングが実現されるコンピュータシステムおよびパーティショニング方法を包含すると考えられえ、分散化されたポイント・ツー・ポイント伝送インフラストラクチャの部分を表すこれらリンク群は相互接続される。例えば、これらリンク群をイネーブルしたりディセーブルしたりする代替の方法は、スイッチを開いたり、閉じたりすることでありえる。

【００５３】

最後に、本発明のさまざまな効果、局面、および目的がここにさまざまな実施形態を参照して説明されてきたが、本発明の範囲はこのような効果、局面、および目的を参照して限定されるべきではないことが理解されよう。むしろ本発明の範囲は添付の特許請求の範囲を参照して決定されるべきである。

【図面の簡単な説明】

【００５４】

【図１】従来のマルチプロセッサシステムのブロック図である。

【図２】ポイント・ツー・ポイントデータ転送インフラストラクチャを持つマルチプロセッサシステムのブロック図である。

【図３】本発明の具体的な実施形態によって設計され動作するマルチプロセッサシステムの一部のブロック図である。

【図４】図３のマルチプロセッサシステムのパーティショニングを制御する例示的なメカニズムのブロック図である。

【図５】本発明とともに用いるプロセッサのブロック図である。

【図６】本発明の具体的な実施形態を示すフローチャートである。

【図７】本発明の他の具体的な実施形態を示すフローチャートである。

【図１】

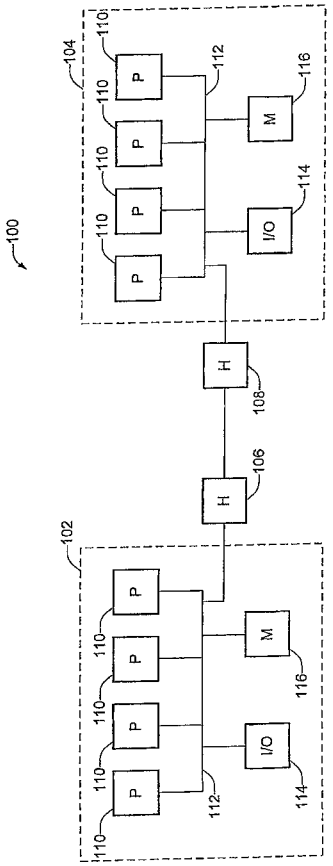


Fig. 1

【図２】

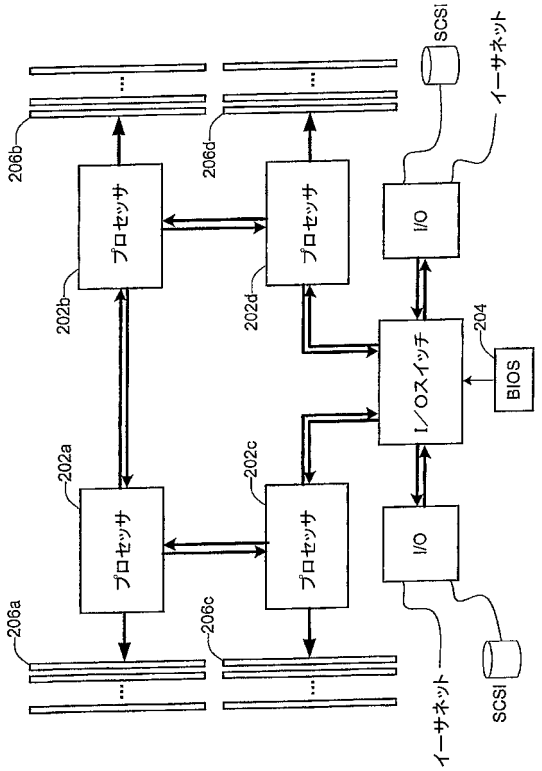


Fig. 2

【図３】

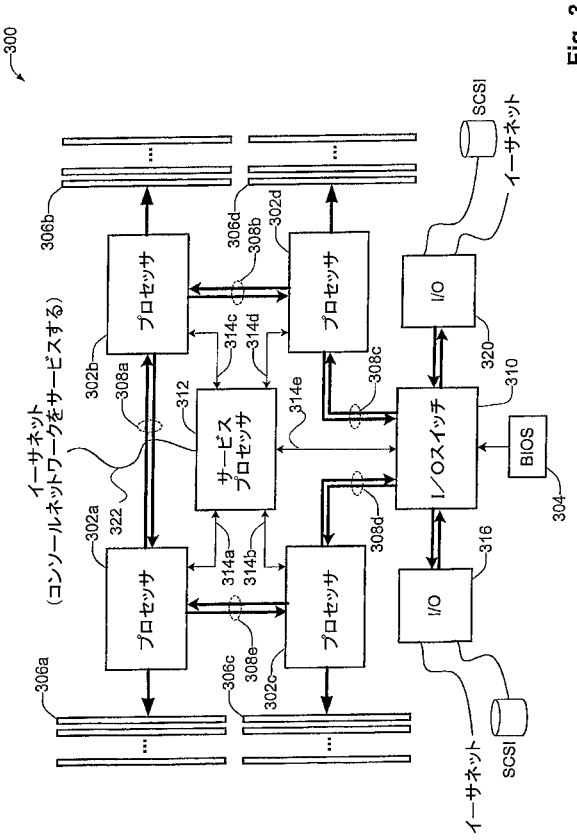


Fig. 3

【図４】

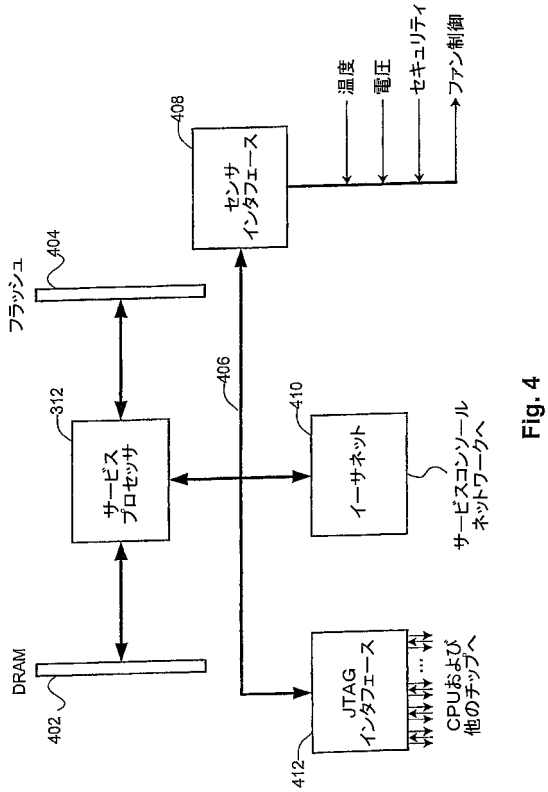


Fig. 4

【図５】

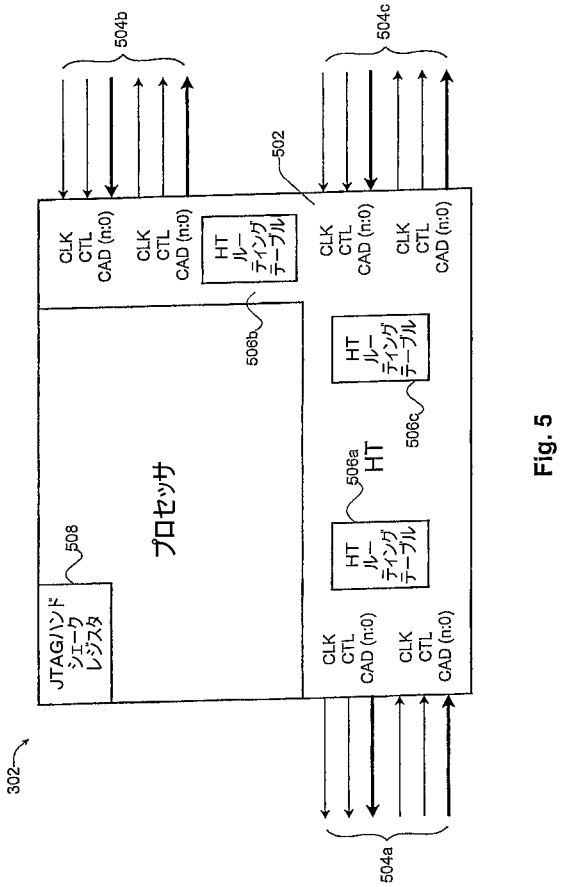


Fig. 5

【図６】

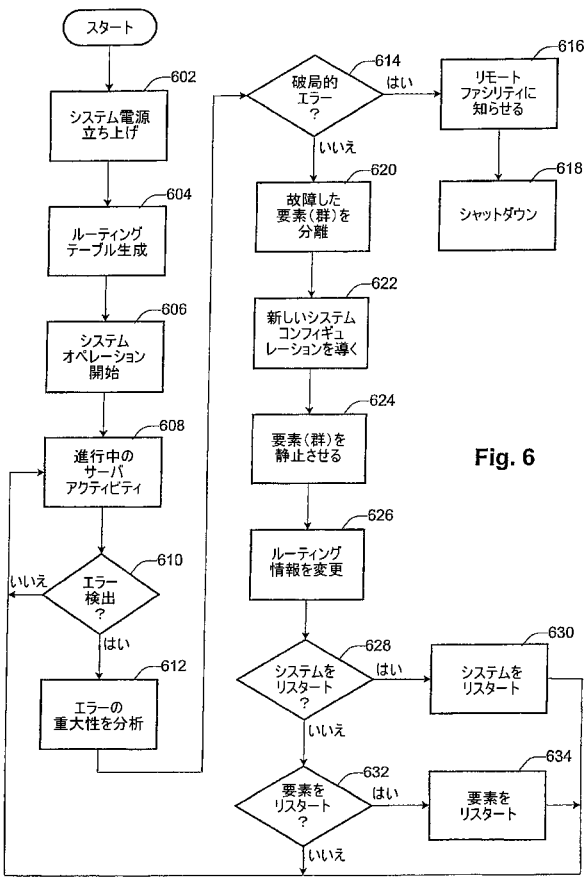


Fig. 6

【図７】

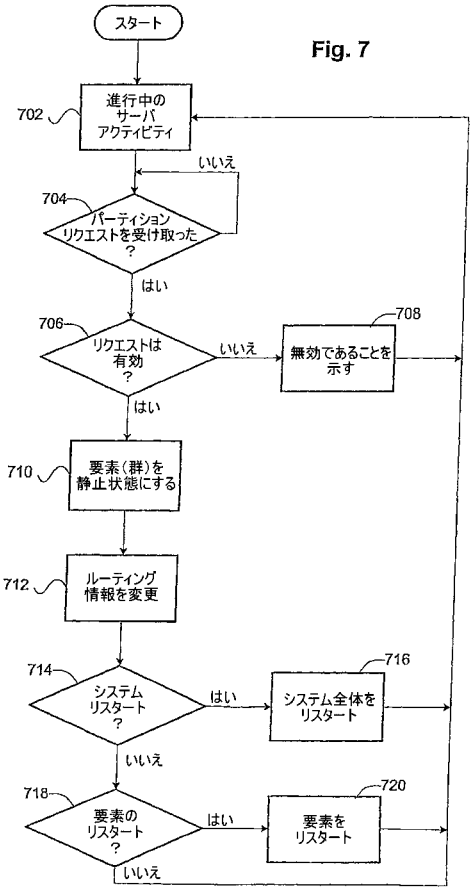

スタート
進行中のサーバアクティビティ
702
いいえ
704
パーティションリクエストを受け取った？
はい
706
リクエストは有効？
いいえ
708
無効であることを示す
はい
710
要素（群）を静止状態にする
712
ルーティング情報を変更
714
システムリスタート？
はい
716
システム全体をリスタート
いいえ
718
要素のリスタート？
はい
720
要素をリスタート
いいえ

Fig. 7

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 February 2003 (27.02.2003)

PCT

(10) International Publication Number
WO 03/017126 A1

(51) International Patent Classification⁷: G06F 15/173

(21) International Application Number: PCT/US02/25530

(22) International Filing Date: 9 August 2002 (09.08.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/932,456 16 August 2001 (16.08.2001) US

(71) Applicant: **NEWISYS, INC.** [US/US]; 11612 Bee Caves Road, Building 1, Suite 150, Austin, TX 78738 (US).

(72) Inventors: **OEHLER, Richard, R.**; 8 Bonny Drive, Somers, NY 10589 (US). **KULPA, William, G.**; 105 Vixen Court, Lakeway, TX 78734 (US).

(74) Agent: **VILLENEUVE, Joseph, M.**; Beyer Weaver & Thomas, LLP, P.O. Box 778, Berkeley, CA 94704-0778 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

WO 03/017126 A1

(54) Title: COMPUTER SYSTEM PARTITIONING USING DATA TRANSFER ROUTING MECHANISM

(57) Abstract: A computer system is described having a plurality of resources which includes a plurality of processors, a distributed point-to-point transmission infrastructure for interconnecting the plurality of processors, and a partitioning processor for configuring the plurality of resources into at least one partition. Each partition comprises a subset of the plurality of resources. The partitioning processor is operable to configure the resources by enabling at least one link between at least one of the plurality of processors and at least one other one of the plurality of processors according to a previously specified partitioning schema. The link(s) so enabled corresponds to a portion of the point-to-point transmission infrastructure.

COMPUTER SYSTEM PARTITIONING USING DATA TRANSFER ROUTING MECHANISM

BACKGROUND OF THE INVENTION

The present invention relates generally to the partitioning of resources in computer systems. More specifically, the present invention provides techniques for more precisely partitioning such resources than previously possible.

The basic idea of partitioning resources in distributed computing systems or even within the central electronic complex (CEC) of a single computer system is not new. However, some prior partitioning techniques have only been able to allocate system resources on a relatively coarse level, e.g., physically separate machines, groups of processors, etc., while others operate somewhat inflexibly, e.g., permanently assigning subsets of processors in a multi-processor group to a partition only upon system initialization.

Some of the drawbacks of previous partitioning techniques may be understood with reference to Fig. 1 which depicts a conventional eight-processor system 100 with two four-processor groups (i.e., quads) 102 and 104 interconnected via hubs 106 and 108. Each of the processors 110 within a conventional quad is connected to a broadcast bus 112 which does not allow isolation of any one of the processors from any of the others, i.e., each request for access to a particular resource (e.g., I/O 114 or memory 116) by any of the processors is received by all of the other processors. Thus, because the only point-to-point communication that occurs in such a system is between the hubs connecting the two quads, the partitioning of such a system can only occur as between the hubs, i.e., one quad in one partition, the other quad in another.

A relatively new approach to the design of multi-processor systems replaces broadcast communication among processors with a point-to-point data transfer mechanism in which the processors communicate similarly to network nodes in a distributed computing system, e.g., a wide area network. That is, the processors are interconnected via a plurality of communication links and requests are transferred among the processors over the links according to routing tables associated with each processor. The intent is to increase the amount of information transmitted within a multi-processor platform per unit time. An example of this approach is the HyperTransport (HT) architecture recently announced by Advanced Micro Devices Inc. (AMD) of Sunnyvale, California, and described, for example, in a paper entitled *HyperTransport™ Technology: A High-Bandwidth, Low-Complexity Bus Architecture* delivered at WinHEC 2001, March 26-28, 2001, at Anaheim, California, the entirety of which is incorporated herein by reference for all purposes. Operation of the AMD architecture will now be briefly described with reference to the simplified block diagram of Fig. 2.

Each time system 200 boots up, the routing tables associated with each of processors 202a-202d must be initialized. The primary processor 202a, i.e., the one communicating with BIOS 204, starts up while the remaining processors are not running and executes a "discovery" algorithm which builds up the routing tables in each of the processors using the HT infrastructure. This algorithm is a "greedy" algorithm which attempts to capture all of the available or "reachable" system resources and build a single, undivided system from the captured resources. Primary processor 202a begins this process by identifying all system resources which are responsive to it, i.e., all of the adjacencies. Then for each such resource (e.g., memory bank 206a and processors 202b and 202c), primary processor 202a requests

all of the resources responsive to (i.e., "owned by") that resource (e.g., memory banks 206b and 206c). Using this information, the primary processor builds the routing tables. Unfortunately, the routing tables generated by such a system remain unchanged during system operation regardless of resource failures, load imbalances, or any other changes in conditions under which a reallocation of system resources would be desirable.

In view of the foregoing, it is desirable to provide techniques by which computer system resources may be more flexibly and precisely partitioned.

SUMMARY OF THE INVENTION

According to the present invention, the innovation represented by distributed point-to-point communication among processors in a single-platform, multi-processor system is leveraged to provide a greater level of flexibility and precision in the partitioning of computer system resources than ever before possible. That is, a distributed point-to-point communication infrastructure is employed to allow precise allocation of system resources through generation of routing tables according to a previously specified schema. According to various embodiments of the present invention, such partitioning may be done statically, e.g., at system start up, or dynamically, e.g., in response to some event, to achieve any desired allocation of system resources.

That is, the present invention provides a computer system having a plurality of resources which includes a plurality of processors, a distributed point-to-point transmission infrastructure for interconnecting the plurality of processors, and a partitioning processor for configuring the plurality of resources into one or more partitions. Each partition comprises a subset of the plurality of resources. The partitioning processor is operable to configure the resources by enabling at least one link between at least one of the plurality of processors and at least one other one of the plurality of processors according to a previously specified partitioning schema. The link(s) so enabled corresponds to a portion of the point-to-point transmission infrastructure. According to a specific embodiment of the invention, the enabling of the link(s) is effected by writing to at least one of a plurality of routing tables associated with the processors according to the previously specified partitioning schema.

Methods for partitioning such a computer system are also described. According to a specific embodiment, the plurality of resources are configured into one or more partitions. Each such partition comprises a subset of the plurality of resources. The configuring of the resources is effected by enabling at least one link between at least one of the plurality of processors and at least one other of the plurality of processors according to a previously specified partitioning schema. The at least one link so enabled corresponds to a portion of the point-to-point transmission infrastructure. According to a specific embodiment of the invention, the enabling of the at least one link is effected by writing to at least one of a plurality of routing tables associated with the processors according to the previously specified partitioning schema.

A further understanding of the nature and advantages of the present invention may be realized by reference to the remaining portions of the specification and the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a conventional multi-processor system;

Fig. 2 is a block diagram of a multi-processor system having a point-to-point data transfer infrastructure;

Fig. 3 is a block diagram of a portion of a multi-processor system which is designed and operates according to a specific embodiment of the present invention;

Fig. 4 is a block diagram of an exemplary mechanism for controlling partitioning of the multi-processor system of Fig. 3;

Fig. 5 is a block diagram of a processor for use with the present invention;

Fig. 6 is a flowchart illustrating a specific embodiment of the present invention; and

Fig. 7 is a flowchart illustrating another specific embodiment of the present invention.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

A specific embodiment of the present invention will now be described with reference to a specific point-to-point communication infrastructure such as, for example, AMD's HyperTransport multi-processor architecture. It should be understood, however, that the scope of the invention is not limited to the particular embodiment described or the AMD architecture. That is, any multi-processor architecture having point-to-point communication among its processors and/or various other system resources is suitable for implementing specific embodiments of the present invention.

Fig. 3 shows a simplified block diagram of a server 300 designed in accordance with a specific embodiment of the present invention. Server 300 includes substantially identical processors 302a-302d, a Basic I/O system (BIOS) 304, a memory subsystem comprising memory banks 306a-306d, point-to-point communication links 308a-308e for interconnecting processors 302a-302d and I/O switch 310, and a service processor 312 which communicates with processors 302a-302d and I/O switch 310 via a JTAG interface represented in Fig. 3 by links 314a-314e. I/O switch 310 connects the rest of the system to I/O adapters 316 and 320.

According to specific embodiments, the service processor of the present invention has the intelligence to partition system resources according to a previously specified partitioning schema as opposed to the use of the "greedy" algorithm described above. This is achieved through direct manipulation of the routing tables associated with the system processors by the service processor which is made possible by the point-to-point communication infrastructure. The routing tables are used to control and isolate various system resources, the connections between which are defined therein.

According to a specific embodiment, the service processor is an autonomous processor with its own set of operating system applications (which is separate from the operating system(s) associated with the rest of the system) and its own I/O. It can run when the rest of the processors, memory, and I/O are not functioning. It operates as an external supervising intelligence which makes sure that all of the system resources are operating as desired.

It should be noted, however, that the previously specified partitioning schema need not be implemented by a separate service processor. That is, for example, one of the system processors could be employed for this purpose. According to such an embodiment, for example, the system BIOS could be altered to effect the schema using the system's primary processor.

In addition, according to various embodiments of the present invention, a partition may represent a variety of system resource combinations. That is, for example, in a "capacity on demand" scenario a partition could be represented by a single processor, removal of the processor from its partition rendering the remaining components unable to run an OS (and therefore the user would not be charged for this partition). A partition could also be represented by a processor and some associated memory or I/O. In general, any functional subset of the resources available in a computer system can be thought of as a partition.

More generally, it should be understood that the specific architecture shown in Fig. 3 is merely exemplary and that embodiments of the present invention are contemplated having different configurations and resource interconnections, and a variety of alternatives for each of the system resources shown. However, for purpose of illustration, specific details of server 300 will be assumed. For example, most of the resources shown in Fig. 3 are assumed to reside on a single electronic assembly.

In addition, memory banks 306a-306d may comprise double data rate (DDR) memory which is physically provided as dual in-line memory modules (DIMMs). I/O adapter 316 may be, for example, an ultra direct memory access (UDMA) controller or a small computer system interface (SCSI) controller which provides access to a permanent storage device. I/O adapter 320 may be an Ethernet card adapted to provide communications with a network such as, for example, a local area network (LAN) or the Internet.

According to a specific embodiment and as shown in Fig. 3, both of I/O adapters 316 and 320 provide symmetric I/O access. That is, each provides access to equivalent sets of I/O. As will be understood, such a configuration would facilitate a partitioning scheme in which multiple partitions have access to the same types of I/O. It should also be understood, however, that embodiments are envisioned in which partitions without I/O are created. For example, a partition including one or more processors and associated memory resources, i.e., a memory complex, could be created for the purpose of testing the memory complex.

According to one embodiment, service processor 312 is a Motorola MPC855T microprocessor which is a PPC controller with integrated chipset functions.

Service processor 312 is primarily responsible for partitioning the resources of server 300. According to the embodiment shown in Fig. 3, service processor 312 allocates usage of processor 302a-302d and I/O switch 310 although it could be programmed to directly manage other resources such as, for example, memory banks or various I/O devices. Configuration of service processor 312 may be effected via a management/server console (not shown) to which service processor 312 is connected via communication link 322.

Fig. 4 is a high level block diagram of the interconnections for service processor 312 according to one implementation. It is important to note that the partitioning engine of the present invention could look very different from service processor 312 as depicted in Figs. 3-5. That is, any mechanism which can intelligently reconfigure the routing tables using a point-to-point communication infrastructure is within the scope of the invention. For example, other possible mechanisms include using one or more of processors 302 to effect the partitioning.

In this embodiment, however, service processor 312 has direct connections to a DRAM storage block 402 and flash memory 404. DRAM 402 facilitates execution by the service processor of a program stored in flash 404. Service processor 312 is also connected via PCI bus 406 to a sensor interface 408, an Ethernet card 410, and a JTAG interface 412. Sensor interface 408 may include, for example, inputs from monitoring circuits (not shown) which provide indications of temperature, supply voltage, or security locks. Sensor interface 408 may also have various outputs such as, for example, a control signal for activating the system's fan. Ethernet card 410 provides an interface between service processor 312 and, for example, a service console by which the network administrator can monitor and configure the server.

According to a specific embodiment, interface 412 fully supports IEEE Standard 1149.1, commonly known as the Joint Test Action Group (JTAG) standard which describes the requirements for a test access port and boundary scan architecture. The test access port (TAP) comprises several pins including a test data in (TDI) pin, a test data out (TDO) pin, a test clock (TCK) pin, a test mode select (TMS) pin, and, optionally, a test reset (TRST) pin for driving the TAP controller to the test-logic-reset state. The TAP controller is a 16-state finite state machine that controls the boundary scan logic on the integrated circuit. As will be explained

further below, JTAG interface 412 facilitates communication between service processor 312 and processors 302a-302d, thereby enabling both static and dynamic partitioning of the computer system's resources. According to a specific embodiment, this communication is facilitated using a simple outbound multiplexer.

Referring once again to Fig. 3 and according to a specific embodiment of the invention, processors 302a-302d may comprise AMD K8 processors which employ a point-to-point data transfer mechanism known as and referred to herein as the HT infrastructure. That is, the processors are interconnected via a plurality of point-to-point communication links (i.e., 308a-308e) and requests are transferred among the processors over the links according to routing tables associated with each processor. The HT infrastructure allows for the interconnection of a variety of computer resources such as, for example, peripheral component interconnect (PCI) devices, accelerated graphic port (AGP) devices, dynamic random access memory (DRAM) devices, and other dedicated, high-bandwidth buses in the computer system.

There are two forms of HT. The first form, non-coherent, or ncHT, was developed for I/O links and is illustrated in Fig. 3 (308c and 308d). The coherent HT, or cHT, mechanism described herein has been developed by AMD for non-uniform memory access matrices in which arrays of processors can access local memory, i.e., memory connected to the processors in their cluster, as well as distant memory, i.e., memory connected to processors in remote clusters.

According to various embodiments of the invention, processors 302a-302d are substantially identical. Fig. 5 is a simplified block diagram of such a processor 302 which includes an HT interface 502 having a plurality of HT ports 504a-504c and routing tables 506a-506c associated therewith. Each HT port allows communication

with other resources, e.g., processors or I/O devices, in the computer system via 16-bit HT links, e.g., links 308a-308e of Fig. 3.

The cHT infrastructure shown in Fig. 5 can be generalized as a point-to-point, distributed routing mechanism which comprises a plurality of segments interconnecting the systems processors (e.g., a goes-in-goes-out (GIGO) bus) according to any of a variety of topologies, e.g., ring, mesh, etc. Each of the endpoints of each of the segments is associated with a connected processor which has a unique node ID and a plurality of associated resources which it "owns," e.g., how much memory it's connected to, what I/O it's connected to, etc.

The routing tables associated with each of the nodes in the distributed routing mechanism collectively represent the current state of interconnection among the computer system resources. Each of the resources (e.g., a specific memory range or I/O device) owned by any given node (e.g., processor) is represented in the routing table(s) associated with the node as an address. When a request arrives at a node, the requested address is compared to a two level entry in the node's routing table identifying the appropriate node and link, i.e., if you want that address, go to node x; and if you want to go to node x use link y.

Associated with the routing tables are controls that are used when these tables are initialized or modified. These controls provide a means to atomically change values in the routing tables while the system continues to operate correctly.

As shown in Fig. 5, processor 302 can conduct point-to-point communication with three other processors according to the information in the associated routing tables. According to a specific embodiment, routing tables 506a-506c comprise two-level tables, a first level associating the unique addresses of system resources (e.g., a memory bank) with a corresponding HT node (e.g., one of the processors), and a

second level associating each HT node with the link (e.g., 308a-308e) to be used to reach the node from the current node.

Processor 302 also has a set of JTAG handshake registers 508 which, among other things, facilitate communication between the service processor and processor 302. That is, the service processor writes routing table entries to handshake registers 508 for eventual storage in routing tables 506a-506c. It should be understood that the processor architecture depicted in Fig. 5 is merely exemplary for the purpose of describing a specific embodiment of the present invention. For example, a fewer or greater number of ports and/or routing tables may be used to implement other embodiments of the invention.

According to a specific embodiment, server 300 of Fig. 3 may be configured using the techniques described herein to operate as a single four-processor system, or as two or more functionally separate partitions. In contrast to the "greedy" algorithm contemplated by the designers of the HT infrastructure which operates without a priori knowledge of the eventual system configuration, service processor 312 facilitates the configuration of server 300 by generating and/or dynamically altering the routing tables associated with all or some of processors 302a-302d (and I/O switch 310) according to a previously specified partitioning schema. This is accomplished by service processor 312 writing routing table entries to the JTAG handshake registers of the appropriate processors (and similar tables associated with I/O switch 310) via interface links 314a-314e. As will be described, this system configuring/partitioning may be done either statically, e.g., at server boot up, or dynamically, e.g., during operation of server 300.

The present invention encompasses embodiments in which a computer system, e.g., a server, designed according to the invention is dynamically reconfigured (e.g.,

repartitioned). That is, the reconfiguration is accomplished without rebooting the system or re-initializing operation of the service processor, and occurs in response to some run-time event such as, for example, failure of a system resource.

In general, it should be understood that dynamic partitioning of system resources according to the invention may occur in response to a wide variety of run-time events. Specific examples of events or stimuli which can precipitate a dynamic partitioning include failure of a resource (e.g., a processor) associated with a currently existing partition, expiration of a predefined time interval (e.g., a maintenance period for a particular system resource), running of a particular application or combination of applications which alters the load distribution, and user instructions to reconfigure partitions.

Such user instructions may be, for example, for experimentation relating to load evaluation for various conditions, reallocation of resources for support of a particular application, reallocation of resources to handle known load variations over the course of a day, or reallocation of system resources in accordance with agreements with users or groups of users. Under the latter scenario, system users could be charged according to the resources that are allocated to them for their use, i.e., capacity on demand, but on a more granular resource level than ever before possible. This is because exploitation of a distributed point-to-point transmission infrastructure for interconnecting a plurality of processors, e.g., the HT infrastructure, allows allocation of, for example, a single processor, memory bank, or I/O bus.

A particular embodiment will now be described in which both static and dynamic system configuration/partitioning according to the techniques of the present invention are accomplished. As used herein, the term static partitioning refers to shutting down one or more operating systems associated with the computer system,

performing the partitioning, and restarting one or more operating systems. By contrast, dynamic partitioning refers to working with one or more currently running operating systems to effect the partitioning.

For example, according to various embodiments of the invention, the service processor could make a request to remove resources from a currently existing partition, in response to which the affected operating system(s) would determine whether it could handle removal without shutting down. If so, the service processor is allowed to reconfigure the appropriate routing tables as described herein. If not, then any partitioning must be done statically.

If, on the other hand, a request to add available resources to a currently existing partition, the service processor makes a call to the affected operating system(s) to add the resources. The affected operating system(s) might then be able to add the resources without stopping operation.

Referring now to Fig. 6, when power is applied to the system (602), the service processor initializes each of the system processors by generating their associated routing tables according to a previously specified configuration (604), i.e., a static partitioning is performed. The initial system configuration may be stored, for example, in flash memory to which the service processor is connected, or specified by the system administrator bringing the system on line. The system configuration may correspond to one or more functionally separate partitions to effect a wide variety of operational goals. Once the routing tables have been generated, system operation begins in which the system resources are allocated to the one or more functionally separate partitions (606).

The service processor then continues to monitor ongoing server activity for the occurrence of a run-time event which requires repartitioning of the system (608). As

described above, the present invention contemplates a wide variety of such run-time events as being suitable for stimulating dynamic partitioning. However, for the purposes of illustration, the failure of a system resource will be used in this exemplary embodiment. Thus, in 608 the service processor looks for system error messages relating to such failures. If such an error message is detected (610), the message is analyzed to determine the severity of the error (612). If the error is catastrophic (614), i.e., the system will be unable to operate reliably, then a remote facility is notified (616), and the system is shut down (618).

If, on the other hand, the error is not catastrophic (614), the failed component(s) are isolated (620) and a new system configuration is derived (622) in which the failed component(s) is (are) excluded from any of the resulting functional partitions. According to some embodiments, the derivation of the new system configuration is done with reference to the current system configuration, e.g., the configuration stored in flash memory associated with the service processor, and the failure information. The service processor then effects the new partitioning scheme by quiescing the appropriate system resources (624) and making the appropriate modifications to the routing tables and controls associated with the system nodes, e.g., the processors (626).

According to a specific embodiment of the present invention, the service processor is configured to automatically quiesce any failed processor and thereby remove it from the available processor pool. The corresponding command may be sent from the service processor to the operating system using the Advanced Configuration and Power Interface (ACPI) 2.0 protocol. Threads already assigned to the failed processor are completed (if possible) and, once the operating system has quiesced, an acknowledgement is sent to the service processor which then removes

the failed processor from its partition. The service processor may assign another one of the processors to the failed processor's partition, leave the partition in its reduced state, or eliminate the partition altogether.

After the failed component(s) has (have) been isolated and any repartitioning defined, a determination is made as to whether it is necessary to restart the system in order to properly initialize the newly formed partitions (628). If so, a system restart is initiated (630). If not, a further determination is made as to whether any specific system components require initialization without necessitating a system-wide reboot (632). If so, a power-on reset signal is sent to the identified component(s) (634). Otherwise, the server is then allowed to continue normal operation. Alternatively, given that failures of system resources often result in the necessity for a system restart, embodiments are contemplated in which a system restart would automatically occur after a system is repartitioned in response to such a failure.

It will be understood with reference to the foregoing that the techniques of the present invention may be employed to deal with power-on failures of a central processing unit (CPU) in a system in which system initialization is directed by one of the system processors rather than the service processor. Because, according to particular embodiments, the service processor of the present invention communicates "out-of-band" with the system processors to configure their routing tables, a computer system can be made more robust in the face of the failure of a partition when the primary system processor goes bad and cannot complete the initialization chain with the secondary system processor(s), i.e., "in-band" communication. In such an instance, when the service processor detects that the primary processor has failed (either from an active failure signal or lack of any signal) then the routing tables and controls of the secondary system processor(s) can be modified directly by the service

processor to exclude the failed processor, designate a new primary system processor, and complete the stalled initialization.

As mentioned above, the present invention contemplates a variety of mechanisms by which dynamic repartitioning of the resources of a computer system may be initiated. One such mechanism, a run-time partitioning request entered by a system administrator, will now be described with reference to the flowchart of Fig. 7. The embodiment shown begins with the monitoring of ongoing server activity for partitioning requests from, for example, a system management console (702). If such a request is received (704), the service processor determines whether the request is valid (706). According to various specific embodiments of the present invention, the validity of a partitioning request may be determined according to any of a variety of criteria. For example, the partitions indicated by the request could be checked against a minimum resource list for a valid partition. If the request is determined not to be valid (706), a response is sent to the console indicating that the request is invalid and denied (708). According to various embodiments, the response could indicate additional information as to why the request is invalid, e.g., the request tried to create a partition which contains no boot image.

If, on the other hand, the partitioning request is determined to be valid (706), the service processor effects the requested partitioning scheme by quiescing the appropriate system resources (e.g., processors, memory, I/O) to effect the new partitions (710), and making the appropriate modifications to the routing tables associated with the quiesced system resources (712).

After the system partitioning has been redefined, a determination is made as to whether it is necessary to restart the entire system for the proper initialization of the newly formed partitions (714). If so, the system is restarted (716). If not, a further

determination is made as to whether any specific system component(s) requires initialization (718). If so, then a power-on reset signal is sent to such component(s) (720). After repartitioning is complete, the monitoring of ongoing server activity resumes (702). Thus, the present invention makes it possible to repartition a server (or any computing system) during run-time.

More generally and as discussed above, such repartitioning can result from a wide variety of run-time events and for a wide variety of purposes. For example, instead of responding to a request from the system administrator, a repartitioning of system resources could occur in response to a particular combination of applications being run. That is, the system may be initially partitioned to run a first combination of applications each of which requires a certain level of resources. When a second combination of applications are to be run requiring a different allocation of system resources, an automatic repartitioning of the system to effect the new allocation of resources may occur.

While the invention has been particularly shown and described with reference to specific embodiments thereof, it will be understood by those skilled in the art that changes in the form and details of the disclosed embodiments may be made without departing from the spirit or scope of the invention. For example, the present invention may be generalized from the embodiments described to implementations having any number of processors.

Moreover, the techniques of the present invention do not necessarily require a separate service processor to effect the static and dynamic partitioning described herein. That is, one of the system's processor could be configured to perform these functions either at system start up or in response to a run-time event. Embodiments are also contemplated in which multiple processors are configured to cooperatively

perform the partitioning. The program for directing these types of operation could be stored in, for example, the system BIOS memory.

In addition, although a specific embodiment is described herein in which the techniques of the present invention are employed both at system initialization and during system operation, it will be understood that the present invention encompasses embodiments in which the techniques enabled by the invention are employed only at one or the other. That is, embodiments in which the invention is only used to initialize a system are contemplated. Conversely, embodiments in which only dynamic partitioning is effected are also contemplated.

It should also be understood that the term "resource" as used herein is contemplated to include far more than the physical implementation of a single microprocessor. That is, according to various embodiments of the invention, a resource can be thought of as any of the major elements of a computing system. Such elements may include physical elements such as one or more processors, banks of memory (typically physical memory, e.g., banks of DIMMs), I/O buses (e.g., PCI or ISA buses), and elements on a bus (e.g., SCSI card, a communication card, Fibre Channel card) to name a few. However, it should also be noted that system resources can be logical elements such as the logical components associated with a processor (e.g., a DMA engine or an interrupt mechanism for routing interrupts).

It is also important to note that the present invention should not be limited to embodiments in which routing tables are manipulated to effect a partitioning scheme. That is, the present invention can be thought of as encompassing computer systems and methods of partitioning in which the partitioning is effected by the enabling and disabling of links between the systems processors, the links representing portions of the distributed point-to-point transmission infrastructure by the processors are

interconnected. For example, an alternative way to enable and disable these links would be by opening and closing switches.

Finally, although various advantages, aspects, and objects of the present invention have been discussed herein with reference to various embodiments, it will be understood that the scope of the invention should not be limited by reference to such advantages, aspects, and objects. Rather, the scope of the invention should be determined with reference to the appended claims.

WHAT IS CLAIMED IS:

1. A computer system, comprising:

a plurality of resources including a plurality of processors;

a distributed point-to-point transmission infrastructure for interconnecting the plurality of processors; and

at least one partitioning processor for configuring the plurality of resources into at least one partition, each partition comprising a subset of the plurality of resources, the at least one partitioning processor being operable to configure the resources by writing to at least one of a plurality of routing tables associated with the processors according to a previously specified partitioning schema, each routing table representing links between an associated processor and other ones of the plurality of processors, the links corresponding to portions of the point-to-point transmission infrastructure.

2. The computer system of claim 1 wherein the plurality of resources further includes at least one of a memory device, a memory range, an I/O bus, I/O devices coupled to an I/O bus, and an interrupt mechanism for routing interrupts.

3. The computer system of claim 1 wherein the plurality of resources includes an I/O switch, the I/O switch having one of the routing tables associated therewith representing links between the I/O switch, at least one of the processors, and at least one I/O resource.

4. The computer system of claim 3 wherein the at least one I/O resource comprises at least one of an Ethernet device and a SCSI device.

5. The computer system of claim 1 wherein each routing table comprises a table of entries, each of selected ones of the entries associating an address of one of the resources with one of the processors and a link for connecting with the one of the processors.

6. The computer system of claim 1 wherein the distributed point-to-point transmission infrastructure comprises a coherent HyperTransport (cHT) infrastructure.

7. The computer system of claim 1 wherein the distributed point-to-point transmission infrastructure interconnects the processors using a ring topology.

8. The computer system of claim 1 wherein the distributed point-to-point transmission infrastructure interconnects the processors using a mesh topology.

9. The computer system of claim 1 wherein the distributed point-to-point transmission infrastructure directly connects each of the processors with every other one of the processors.

10. The computer system of claim 1 wherein the at least one partitioning processor comprises at least one of the plurality of processors interconnected by the distributed point-to-point transmission infrastructure.

11. The computer system of claim 1 wherein the at least one partitioning processor is separate from the plurality of processors interconnected by the distributed point-to-point transmission infrastructure.

12. The computer system of claim 11 further comprising a boot memory for facilitating initialization of the computer system, the boot memory having computer program instructions stored therein for facilitating operation of at least one of the plurality of processors as the at least one partitioning processor.

13. The computer system of claim 1 wherein the previously specified partitioning schema is generated in response to an event occurring during operation of the computer system.

14. The computer system of claim 13 wherein the event comprises one of initialization of the computer system, a failure of at least one of the resources, a change in operating load associated with at least one of the resources, passage of a period of time, use of particular software, and a change in available power resources.

15. The computer system of claim 1 further comprising at least one partitioning processor link for connecting the at least one partitioning processor with a user interface, and wherein the previously specified partitioning schema is specified by a user of the computer system via the user interface and the at least one partitioning processor link.

16. The computer system of claim 1 wherein the at least one partitioning processor is operable to generate the routing tables upon initialization of the computer system.

17. The computer system of claim 1 wherein the at least one partitioning processor is operable to alter the at least one of the routing tables during operation of the computer system.

18. The computer system of claim 1 wherein the at least one partition comprises a plurality of partitions.

19. The computer system of claim 18 wherein at least one of the plurality of partitions comprising a functional subset of the plurality of resources.

20. The computer system of claim 1 wherein the at least one partition comprises a single partition including all operational ones of the plurality of resources.

21. The computer system of claim 1 wherein the at least one partitioning processor comprises one partitioning processor.

22. The computer system of claim 1 wherein the at least one partitioning processor comprises more than one partitioning processor.

23. A computer implemented method for use in a computer system having a plurality of resources including a plurality of processors and a distributed point-to-point transmission infrastructure for interconnecting the plurality of processors, the method comprising configuring the plurality of resources into at least one partition, each partition comprising a subset of the plurality of resources, the configuring of the resources being effected by writing to at least one of a plurality of routing tables associated with the processors according to a previously specified partitioning schema, each routing table representing links between an associated processor and other ones of the plurality of processors, the links corresponding to portions of the point-to-point transmission infrastructure.

24. The method of claim 23 wherein the plurality of resources includes an I/O switch, the I/O switch having one of the routing tables associated therewith representing links between the I/O switch, at least one of the processors, and at least one I/O resource.

25. The method of claim 24 wherein the distributed point-to-point transmission infrastructure comprises a non-coherent HyperTransport (ncHT) infrastructure.

26. The method of claim 23 wherein configuring the plurality of resources is achieved using at least one partitioning processor which comprises at least one of the plurality of processors interconnected by the distributed point-to-point transmission infrastructure.

27. The method of claim 23 wherein configuring the plurality of resources is achieved using at least one partitioning processor which is separate from the plurality of processors interconnected by the distributed point-to-point transmission infrastructure.

28. The method of claim 23 further comprising generating the previously specified partitioning schema in response to an event occurring during operation of the computer system.

29. The method of claim 28 wherein the event comprises one of initialization of the computer system, a failure of at least one of the resources, a change in operating load associated with at least one of the resources, passage of a period of time, use of particular software, and a change in available power resources.

30. The method of claim 23 further comprising receiving the previously specified partitioning schema as specified by a user of the computer system.

31. The method of claim 23 wherein writing to the at least one of the plurality of routing tables comprises generating the plurality of routing tables upon initialization of the computer system.

32. The method of claim 23 wherein writing to the at least one of the plurality of routing tables comprises altering the at least one of the routing tables during operation of the computer system.

33. The computer system of claim 23 wherein the at least one partition comprises a plurality of partitions.

34. The computer system of claim 33 wherein at least one of the plurality of partitions comprising a functional subset of the plurality of resources.

35. The computer system of claim 23 wherein the at least one partition comprises a single partition including all operational ones of the plurality of resources.

36. A computer system, comprising:

a plurality of resources including a plurality of processors;

a distributed point-to-point transmission infrastructure for interconnecting the plurality of processors; and

at least one partitioning processor for configuring the plurality of resources into at least one partition, each partition comprising a subset of the plurality of resources, the at least one partitioning processor being operable to configure the resources by enabling at least one link between at least one of the plurality of processors and at least one other one of the plurality of processors according to a previously specified partitioning schema, the at least one link corresponding to a portion of the point-to-point transmission infrastructure.

37. The computer system of claim 36 wherein enabling the at least one link comprises writing to at least one of a plurality of routing tables associated with the processors according to the previously specified partitioning schema.

38. The computer system of claim 36 wherein enabling the at least one link comprises closing at least one switch associated with the at least one link according to the previously specified partitioning schema.

39. A computer implemented method for use in a computer system having a plurality of resources including a plurality of processors and a distributed point-to-point transmission infrastructure for interconnecting the plurality of processors, the method comprising configuring the plurality of resources into at least one partition, each partition comprising a subset of the plurality of resources, the configuring of the resources being effected by enabling at least one link between at least one of the plurality of processors and at least one other one of the plurality of processors according to a previously specified partitioning schema, the at least one link corresponding to a portion of the point-to-point transmission infrastructure.

40. The method of claim 39 wherein enabling the at least one link comprises writing to at least one of a plurality of routing tables associated with the processors according to the previously specified partitioning schema.

41. The method of claim 39 wherein enabling the at least one link comprises closing at least one switch associated with the at least one link according to the previously specified partitioning schema.

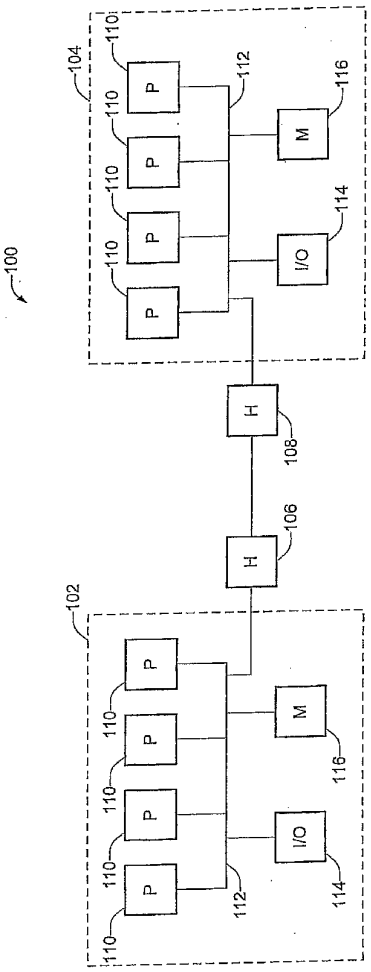


Fig. 1

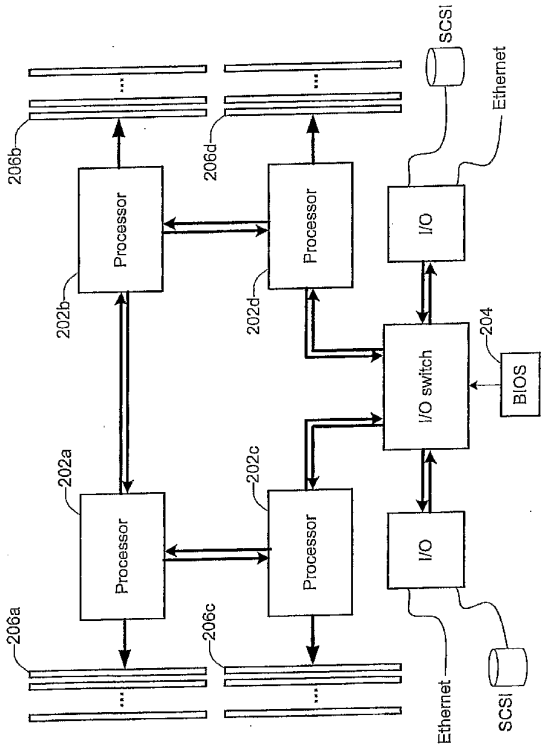

SCSI
Ethernet
206b
206d
Processor
Processor
I/O
202b
202d
204
I/O switch
BIOS
202a
202c
Processor
Processor
I/O
206a
206c
Ethernet
SCSI

Fig. 2

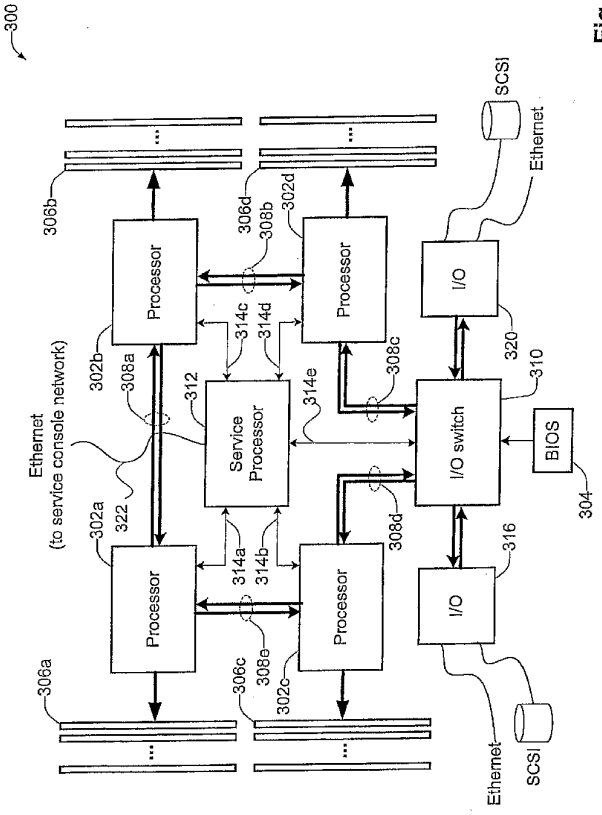

300
306b
306d
308b
302d
SCSI
Ethernet
Processor
Processor
I/O
314c
314d
320
302b
308a
312
308c
314e
310
Ethernet
(to service console network)
Service Processor
I/O switch
BIOS
304
302a
322
308d
316
314a
314b
Processor
Processor
I/O
306a
306c
308e
302c
Ethernet
SCSI

Fig. 3

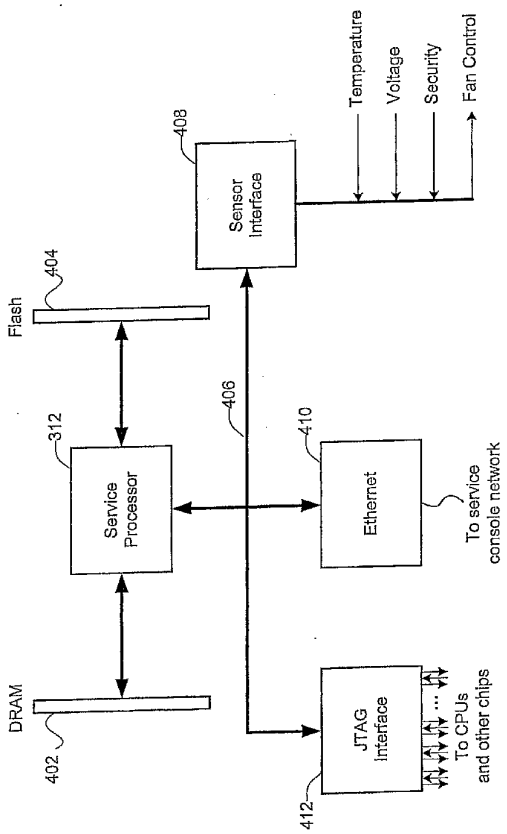

Temperature
Voltage
Security
Fan Control
408
Sensor Interface
404
Flash
406
312
Service Processor
410
Ethernet
To service console network
DRAM
402
JTAG Interface
To CPUs and other chips
412

Fig. 4

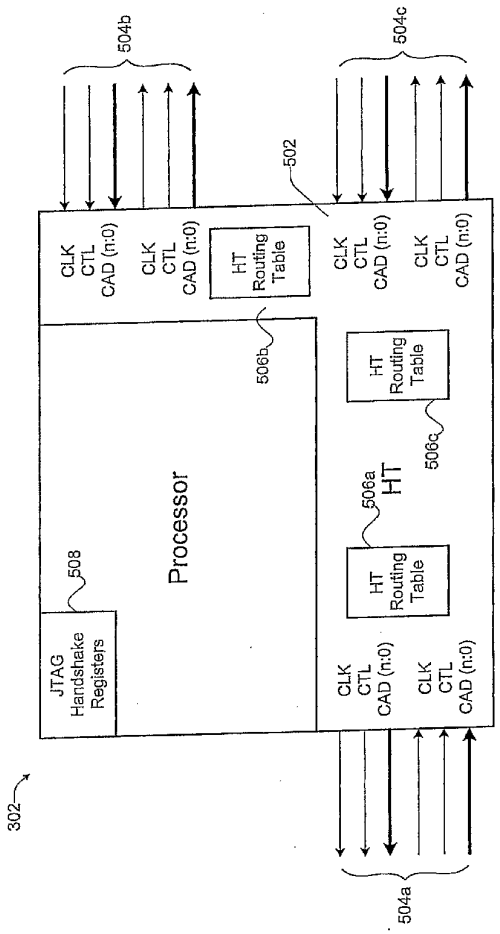

504b
504c
502
CLK CTL CAD (n:0)
CLK CTL CAD (n:0)
HT Routing Table
CLK CTL CAD (n:0)
CLK CTL CAD (n:0)
506b
HT Routing Table
506c
506a HT
HT Routing Table
508
JTAG Handshake Registers
Processor
CLK CTL CAD (n:0)
CLK CTL CAD (n:0)
302
504a

Fig. 5

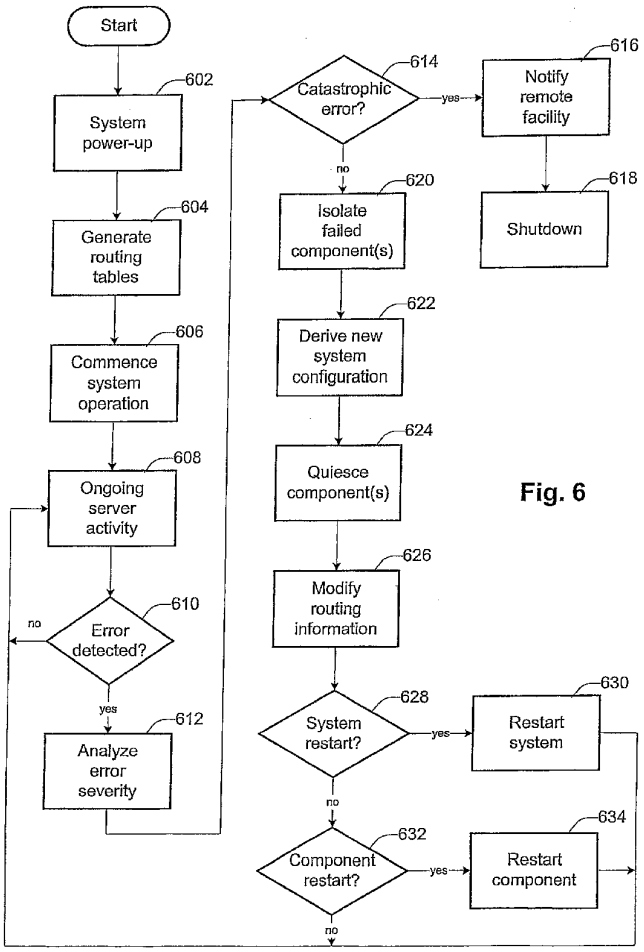
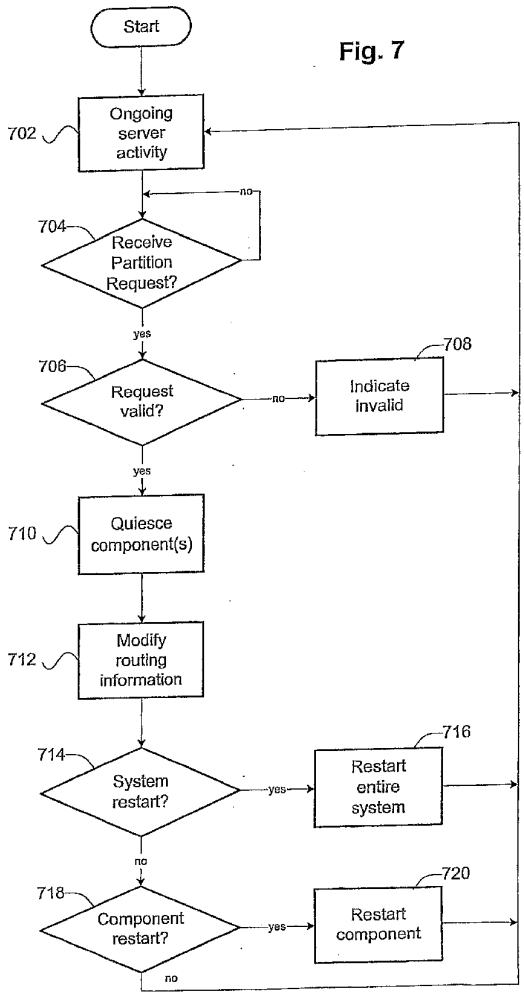

Start
602
System power-up
604
Generate routing tables
606
Commence system operation
608
Ongoing server activity
610
Error detected?
no
yes
612
Analyze error severity
614
Catastrophic error?
yes
616
Notify remote facility
618
Shutdown
no
620
Isolate failed component(s)
622
Derive new system configuration
624
Quiesce component(s)
626
Modify routing information
628
System restart?
yes
630
Restart system
no
632
Component restart?
yes
634
Restart component
no

Fig. 6

Fig. 7


Start
Ongoing server activity
702
no
704
Receive Partition Request?
yes
706
Request valid?
no
708
Indicate invalid
yes
710
Quiesce component(s)
712
Modify routing information
714
System restart?
yes
716
Restart entire system
no
718
Component restart?
yes
720
Restart component
no

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/25530

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 15/173
US CL : 709/ 238, 242, 243
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 709/ 238, 242, 243

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,970,232 A (PASSINT et al) 19 October 1999 (19.10.1999), abstract, column 3, lines 15-67, column 4, lines 1-10.	1-41
Y	US 6,188,759 B1 (LORENZEN et al) 13 February 2001 (13.02.2001), abstract, column 1, lines 18-67.	1-41
Y,P	US 2001/0037435 A1 (VAN DOREN) 01 November 2001 (01.11.2001), [0011]-[0016].	1-41

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search
17 October 2002 (17.10.2002)

Date of mailing of the international search report
03 JAN 2003

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Meng-Ai T An
James R. Matthews
Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VC,VN,YU,ZA,ZM,ZW

（特許庁注：以下のものは登録商標）
イーサネット

(72)発明者 クルパ・ウィリアム・ジー.
アメリカ合衆国 テキサス州７８７３４ レイクウェイ，ヴィクセン・コート，１０５

Ｆターム(参考) 5B083 AA04 BB03 CD01 CD07 CD09 GG08
5B098 GD03 GD07 GD15 HH01