(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6398658号
(P6398658)
(45)発行日　平成30年10月3日(2018.10.3)　(24)登録日　平成30年9月14日(2018.9.14)

(51)Int.Cl.　F I
G06F 11/20 (2006.01)　G06F 11/20 617

請求項の数 19 (全 19 頁)

(21)出願番号　特願2014-240195(P2014-240195)
(22)出願日　平成26年11月27日(2014.11.27)
(65)公開番号　特開2015-210812(P2015-210812A)
(43)公開日　平成27年11月24日(2015.11.24)
審査請求日　平成29年8月4日(2017.8.4)
(31)優先権主張番号　14165974.8
(32)優先日　平成26年4月25日(2014.4.25)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100192636
弁理士　加藤　隆夫
(72)発明者　サザン・ジェームズ　アラステア
イギリス国，アールジー２　Ｏディージェイ，レディング，ティペット　ライズ　１６番

最終頁に続く

(54)【発明の名称】アプリケーション・データを回復する方法

(57)【特許請求の範囲】
【請求項１】
インターコネクトによって接続された複数のノードを有するコンピュータ・システムにおいて障害を起こしたノードのメモリからアプリケーション・データを回復し、該アプリケーション・データを置換ノードに書き込む方法であって、
前記コンピュータ・システムのノードがアプリケーションを実行し、該アプリケーションはアプリケーション・データを生成し、該アプリケーションの最も最近の状態をノード・メモリに記憶し；
前記ノードが障害を起こし；
前記障害を起こしたノードのノード・メモリはその後、フェイルオーバー・メモリ・コントローラを使って制御され；
前記フェイルオーバー・メモリ・コントローラは前記アプリケーション・データを前記障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリに前記インターコネクトを通じてコピーし、
前記アプリケーションが走っている間であって前記ノード障害の前に、前記アプリケーションは、前記障害を起こしたノードおよび／または前記置換ノードのノード・メモリにおいて前記アプリケーションによって利用可能または利用不能であるノード・メモリの一部を登録する、
方法。
【請求項２】

前記アプリケーションが、前記利用可能または利用不能であるノード・メモリの一部を登録することを、前記障害を起こしたノードが内部的に前記アプリケーションにメモリを割り当てた後に行なう、
請求項１記載の方法。
【請求項３】
前記アプリケーションが前記一部を前記フェイルオーバー・メモリ・コントローラに登録する、請求項２記載の方法。
【請求項４】
前記障害を起こしたノードのノード・メモリが前記障害後に補助電力を供給され、補助電源は前記フェイルオーバー・メモリ・コントローラによって制御される、請求項１または２記載の方法。
【請求項５】
前記補助電源が、前記フェイルオーバー・メモリ・コントローラと一緒に設けられたバッテリーの形である、請求項１ないし４のうちいずれか一項記載の方法。
【請求項６】
前記補助電源が、前記障害の前およびあとに前記フェイルオーバー・メモリ・コントローラに給電する、請求項１ないし５のうちいずれか一項記載の方法。
【請求項７】
前記補助電源が、前記フェイルオーバー・メモリ・コントローラのインターコネクト接続に給電する、請求項１ないし６のうちいずれか一項記載の方法。
【請求項８】
前記補助電力は、前記障害を起こしたノードのプロセッサまたはノード・メモリ・コントローラのような別のコンポーネントを介してではなく、前記ノード・メモリに直接供給される、請求項１ないし７のうちいずれか一項記載の方法。
【請求項９】
前記フェイルオーバー・メモリ・コントローラは、前記障害を起こしたノードのプロセッサまたはノード・メモリ・コントローラのような別のコンポーネントを介してではなく、前記インターコネクトに直接結合される、請求項１ないし８のうちいずれか一項記載の方法。
【請求項１０】
前記コンピュータ・システムの管理プロセスがノードをモニタリングし、前記ノード障害を検出し、前記置換ノードを同定し、前記フェイルオーバー・メモリ・コントローラに前記アプリケーション・データを前記障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリにコピーするよう命令する、請求項１ないし９のうちいずれか一項記載の方法。
【請求項１１】
前記管理プロセスがまた、前記置換ノード上の前記アプリケーションを再開する、請求項１０記載の方法。
【請求項１２】
補助電源および／またはフェイルオーバー・メモリ・コントローラが単一のノードのためにまたは一群のノードのために設けられる、請求項１ないし１１のうちいずれか一項記載の方法。
【請求項１３】
それぞれが独自のノード・メモリおよびノード・メモリ・コントローラを有する、インターコネクトによって接続された複数のノードを有するコンピュータ・システム上でアプリケーションを走らせているときに、障害を起こしたノードからの回復において使うためのフェイルオーバー・メモリ・コントローラであって、
当該フェイルオーバー・メモリ・コントローラは前記障害を起こしたノードの前記メモリ・コントローラおよび／またはメモリに接続するよう動作可能であり；
当該フェイルオーバー・メモリ・コントローラは、前記障害を起こしたノードのノード

・メモリに記憶されているアプリケーション・データの、置換ノードのノード・メモリへの、前記インターコネクトを通じた転送を制御するよう構成されており、
前記アプリケーションが走っている間であって前記ノード障害の前に、前記アプリケーションは、前記障害を起こしたノードおよび／または前記置換ノードのノード・メモリにおいて前記アプリケーションによって利用可能または利用不能であるノード・メモリの一部を登録する、
フェイルオーバー・メモリ・コントローラ。
【請求項１４】
前記フェイルオーバー・メモリ・コントローラは、前記コンピュータ・システム内のノードから自律的である、請求項１３記載のフェイルオーバー・メモリ・コントローラ。
【請求項１５】
それぞれが独自のノード・メモリおよびノード・メモリ・コントローラを有する複数のノードと；
それらのノード上でアプリケーションを実行するときに障害を起こしたノードからの回復において使うためのフェイルオーバー・メモリ・コントローラと；
それらのノードおよび前記フェイルオーバー・メモリ・コントローラを接続するインターコネクトとを有するコンピュータ・システムであって、
前記フェイルオーバー・メモリ・コントローラは障害を起こしたノードの前記メモリ・コントローラおよび／またはメモリに接続するよう動作可能であり；
前記フェイルオーバー・メモリ・コントローラは、前記障害を起こしたノードのノード・メモリに記憶されているアプリケーション・データの、置換ノードのノード・メモリへの、前記インターコネクトを通じた転送を制御するよう構成されており、
前記アプリケーションが走っている間であって前記ノード障害の前に、前記アプリケーションは、前記障害を起こしたノードおよび／または前記置換ノードのノード・メモリにおいて前記アプリケーションによって利用可能または利用不能であるノード・メモリの一部を登録する、
コンピュータ・システム。
【請求項１６】
前記フェイルオーバー・メモリ・コントローラが各ノードのために設けられる、
請求項１５記載のコンピュータ・システム。
【請求項１７】
前記フェイルオーバー・メモリ・コントローラは、ノードの電力およびインターコネクト接続とは別個の電力接続およびインターコネクト接続を有する、
請求項１５または１６記載のコンピュータ・システム。
【請求項１８】
フェイルオーバー・メモリ・コントローラおよびそれぞれノード・メモリを含む複数のノードを有するコンピュータ・システム上で走るデーモンであって、前記複数のノードおよびフェイルオーバー・メモリ・コントローラはみなインターコネクトによって接続されており、
当該デーモンは前記ノード上でのアプリケーションの実行をモニタリングし、
当該デーモンはノード障害を検出し、置換ノードを同定し、前記フェイルオーバー・メモリ・コントローラに、前記障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリに前記インターコネクトを通じてアプリケーション・データをコピーするよう命令し、
前記アプリケーションが走っている間であって前記ノード障害の前に、前記アプリケーションは、前記障害を起こしたノードおよび／または前記置換ノードのノード・メモリにおいて前記アプリケーションによって利用可能または利用不能であるノード・メモリの一部を登録する、
デーモン。
【請求項１９】

前記アプリケーションが前記複数のノードによって分散式に実行される、請求項１ないし１２のうちいずれか一項記載の方法。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、それだけではないが特に並列コンピューティング環境および高性能コンピューティング（HPC: high performance computing）用途における、障害のあった計算ノードからのデータの回復に関する。本発明は、障害耐性のある分散式コンピューティングの分野で特に用途を見出し、エクサスケール・コンピュータに力点を置く。
【背景技術】
【０００２】
HPCシステムにおいては通例、計算集約的なおよび他の大規模なアプリケーションが実行される。そのようなHPCシステムはしばしば、実行可能形式の処理スレッドまたはプロセスのようなイベントの独立した諸シーケンスが並列に自律的に実行できる複数の処理ユニットまたは「コア」がある分散環境を提供する。
【０００３】
多くの異なるハードウェア構成およびプログラミング・モデルがHPCに適用可能である。現在HPCに対する一般的なアプローチは、それぞれ一つまたは複数のマルチコア・プロセッサ（または「チップ」）を有する複数のノードが高速ネットワークによって相互接続されるクラスター・システムである。各ノードは独自のメモリ領域をもつと想定され、該メモリ領域はそのノード内のすべてのコアにとってアクセス可能である。クラスター・システムは、一般的機能を実行するための既存のコード・ライブラリを利用して、ソース・コードを書く人間のプログラマーによってプログラムされることができる。次いでソース・コードはコンパイルされて、より低レベルの実行可能コード、たとえば特定の命令セットをもつプロセッサ型によって実行されることができるISA（Instruction Set Architecture［命令セット・アーキテクチャ］）レベルのコードに、あるいは特定のプロセッサ専用のアセンブリ言語にされる。しばしば、アセンブリ・コードをアセンブルまたは（仮想マシンの場合）インタープリットして実行可能な機械コードにする最終段がある。アプリケーションの実行可能形式（時に単に「実行可能物」と称される）は、オペレーティング・システム（OS）の監督のもとで実行され、ハードウェアを制御するためにOSおよびライブラリを使用する。使用されるソフトウェアの種々のレイヤーはまとめてソフトウェア・スタックと称されてもよい。
【０００４】
本稿で使うところの「ソフトウェア・スタック」という用語は、アプリケーションを実行するために必要とされるソフトウェアすべてを含み、基礎レベルのソフトウェア（オペレーティング・システムまたはOS）；たとえばノード間のインターコネクト、ディスクまたは他のメモリなどのようなハードウェア・コンポーネントとインターフェースをもつライブラリ（やはり一種のシステム・ソフトウェア）およびアプリケーション自身を含む。現在実行中のアプリケーションは、システム・ソフトウェアの上の、ソフトウェア・スタックの最上レイヤーと見ることができる。
【０００５】
複数のコアをもつコンピュータ・システムのアプリケーションは、プログラマーが複数コアの並列処理能力を活用できるようにするためのライブラリを増補した通常のコンピュータ言語（C/C++またはフォートランのような）で書かれてもよい。この点に関し、コアで実行される「プロセス」という言い方をするのが通例である。（マルチスレッド化された）プロセスは、マルチコアCPU内のいくつかのコアにまたがって実行されてもよく、各ノードは一つまたは複数のCPUを含んでいてもよい。そのような一つのライブラリは、メッセージ渡しインターフェース（MPI: Message Passing Interface）であり、これは分散メモリ・モデル（各プロセスがその独自のメモリ領域をもつものと想定される）を使

い、プロセス間の通信を容易にする。MPIは、プロセスのグループが定義され、区別されることを許容し、いわゆる「バリア同期（barrier synchronisation）」のためのルーチンを含む。これは、複数のプロセスまたは処理要素が協働できるようにするために重要な特徴である。

【０００６】

あるいはまた、共有メモリ並列プログラミングでは、すべてのプロセスまたはコアが同じメモリまたはメモリ領域にアクセスできる。共有メモリ・モデルでは、プロセス間のデータの通信を明示的に指定する必要はない（あるプロセスによってなされる変更は他のすべてのプロセスにとって透明なので）。しかしながら、同時に一つのプロセスのみがデータを修正することを保証するために、共有メモリへのアクセスを制御するライブラリを使うことが必要であることがある。

【０００７】

エクサスケール・コンピュータ（すなわち、1エクサフロップ（10^{18}浮動小数点演算毎秒）の持続的性能を発揮できるHPCシステム）は2020年までには配備されることが期待されている。この時間枠でエクサスケール・システムを開発するいくつかの国家的プロジェクトが発表されている。ペタスケール（現在の最新技術、約10^{15}フロップス）からエクサスケールへの移行は、ハードウェア技術における根本的な変更を必要とするものと予期されている。もはやプロセッサ・クロック周波数におけるさらなる上昇はなく、よってパフォーマンス改善は並列性または並行性の増大（可能性としては約10億個のコアまで）から帰結することになる。エクサスケール・システムの電力使用を受け容れ可能な窓内に保つという要求は、低電力（かつ低コスト）コンポーネントが使用される可能性が高いことを意味し、その結果、各コンポーネントについての平均故障間隔が短くなる。このように、エクサスケール・システムは、今日の最新技術のシステムよりもずっと多くのコンポーネントを含むことになる。そして、各コンポーネントは、今日の対応物より頻繁に故障する可能性が高くなる。エクサスケール・システムについての平均コンポーネント故障間隔は（現行のシステムの場合の日単位ではなく）分単位で測られる可能性が高い。

【０００８】

したがって、エクサスケール・ソフトウェアは特に、これらの障害に対する向上した耐性を必要とすることになり、コンポーネント障害を乗り越えて実行を継続することができる必要があるであろう。HPCアプリケーションは一般に、作業が利用可能な計算コアのすべてを横断して分散されることを保証するよう、慎重に負荷バランスが取られるので、故障したノードに割り当てられていた作業を実行するために置換ノードがアプリケーションにとって利用可能にされることが重要となりうる（この作業をすでにロードされている残りのノードの一つまたは複数に割り当てることは、負荷バランスを崩壊させ、著しいパフォーマンス劣化につながる可能性が高い）。

【０００９】

図１は、障害を起こしたノードを置換ノードで置き換えるプロセスを示している。この図は、システム中の六つのノード（ノード０～５）を示しており、ノード５が障害を起こし、もはやアプリケーションの実行に寄与できなくなっている。もちろん、現実には、ずっと多くのノードがコンピュータ・システムを構成している。置換ノード（ノード６）がシステムに利用可能にされ、障害を起こしたノードを置き換えるために挿入される。ひとたび置換ノードがアプリケーションに割り当てられたら、実行を継続するために必要とされるデータ（たとえば、アプリケーションによって計算された変数の値）で初期化することも必要である。

【発明の概要】

【発明が解決しようとする課題】

【００１０】

置換ノードを初期化する必要性は新しいものではなく、既知の初期化技法は次のようなものを含む。

【００１１】

・ノードをチェックポイント・ファイルから再開する。この方法は、データが正しい値に初期化されることを保証する。しかしながら、チェックポイントを生成するのは時間がかかる（大量のデータを別のノード上のメモリまたはディスク上のファイルにコピーすることに関わるため）。よって、データは一般に定期的にチェックポイント化され、チェックポイント間の間隔は比較的大きい。よって、チェックポイントからデータを復元するとき、最後のチェックポイント以降のすべての計算が繰り返される必要がある（少なくとも障害を起こしたノード上で、可能性としてはグローバルに）。このように、チェックポイント化には、時間的に二重のオーバーヘッドがある。まずはチェックポイントを生成する時、そしてチェックポイントから復元するために読み、再計算する時に再びである。

【００１２】

・障害を起こしたノード上の値を、生き残っているノード上の等価なデータの値から補間する。この方法は常に可能なわけではない（たとえば、各ノードがモデリング・アルゴリズムにおいて空間の別個の領域を受け持つ場合には、単にその境界上の値からこの領域の全体にわたる解を補間することが可能である可能性は低い）。障害を起こしたノード上のデータを他のノード上のデータから補間することが可能であるとしても、これを行なう結果として正確さが失われる。

【００１３】

これら従来技術の技法はいずれも欠点をもち、よって置換ノードを初期化する代替的な方法を提供することが望ましい。

【課題を解決するための手段】

【００１４】

本発明のある側面の実施形態によれば、インターコネクトによって接続された複数のノードを有するコンピュータ・システムにおいて障害を起こしたノードのメモリからアプリケーション・データを回復し、該アプリケーション・データを置換ノードに書き込む方法であって、前記コンピュータ・システムのノードがアプリケーションを実行し、該アプリケーションはアプリケーション・データを生成し、該アプリケーションの最も最近の状態をノード・メモリに記憶し；障害を起こしたノードのノード・メモリはその後、フェイルオーバー・メモリ・コントローラを使って制御され；前記フェイルオーバー・メモリ・コントローラは前記アプリケーション・データを障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリに前記インターコネクトを通じてコピーする、方法が提供される。

【００１５】

発明者らは、必要とされているのが、ノード外に繰り返しデータをコピーする必要なしに（そのようなコピーは時間がかかり、たいていの場合不必要なので）、障害を起こしたノード上の実際のデータ（正確さが失われる補間されたものではなく）を、好ましくは（ある時間前のものではなく）障害直前の状態から得る方法であることを認識するに至った。

【００１６】

発明実施形態は、現在の最新技術の限界を克服するために、障害を起こしたノードのメモリから直接データを回復する方法を提案する。

【００１７】

このように、発明実施形態によれば、複数のノードがあり、あるノードが障害を起こす場合、そのノードは置換ノードで置き換えられる。次いで、フェイルオーバー・メモリ・コントローラ（事実上、ノードが障害を起こしたときに使われるスタンバイ・メモリ・コントローラ）が障害を起こしたノードのノード・メモリの制御を受け持ち、アプリケーション・データを障害を起こしたノードのノード・メモリから置換ノードのノード・メモリにコピーする。したがって、この方法は、アプリケーションの最も最近の状態であるアプリケーション・データを復元でき、障害を起こしたノードからの、アプリケーションによって使用されているノード・メモリの内容全体を含め、それを置換ノードにコピーすることができる。置換ノードは、アプリケーションの実行において前に使われていなかったノ

ードであってもよい。置換ノードは、アプリケーションを実行するノードの障害があった場合のための予備に取っておかれたノードであってもよい。置換ノードは、初期化されたあとは、他のノードと同じ仕方で扱われてもよい。

【００１８】

複数のノードを有するシステムにおけるアルゴリズムの並列実行において、アプリケーション・データは、たとえば、そのノード上で計算された当該アルゴリズムの部分の最新バージョンを含むことができる。

【００１９】

ノード・メモリの全部がアプリケーション・データのために利用可能でないこともある。ノード・メモリは、たとえばオペレーティティング・システムのプロセスのような他のプロセスによっても使用されてもよい。ノード・メモリのある種の部分は、他の用途のために予約されていて、よってアプリケーション・データのためには利用不能であることもある。そのような状況では、アプリケーションはノード・メモリの一部を、好ましくはフェイルオーバー・メモリ・コントローラに登録してもよい。この登録は、現在実行中のノードのものであってもよく、よって障害を起こしたノード・メモリの正しいセクションのみをコピーすることを許容してもよい。代替的または追加的に、この登録は、置換ノードのものであってもよく、データが置換ノードの利用可能なセクションにコピーされることを保証してもよい。登録は、アプリケーションが実行中の任意の時点に行なわれることができるが、ノード障害の前に行なわれる。好ましい実施形態では、登録は、アプリケーションがノード内でメモリを割り当てられたのちできるだけ早く行なわれる。

【００２０】

登録されるノード・メモリの部分は、アプリケーションによる使用のために利用可能であるメモリの部分であることも、あるいは利用可能でないメモリの部分であることもできる（これらは等価であり、よってどちらのオプションでも障害を起こしたノードの正しいセクションおよび／または利用可能な置換セクションの判別を許容することになる）。ノード・メモリの前記部分は、障害を起こしたノード、置換ノードまたは障害を起こしたノードと置換ノードの両方のノード・メモリの一部分であることができる。それらのノードの一方または両方においてノード・メモリの全部がアプリケーション・データに利用可能ではない状況では、この実施形態は、障害を起こしたノードのアプリケーション・データのみが置換ノードにコピーされることおよび／またはそのデータが置換ノードのノード・メモリの利用可能な部分にコピーされることを保証できる。登録は好ましくは、アプリケーションにおいて使用される全ノードについてのパターンとしてではなく、ノード毎であり、そのため、各ノードは内部的にどのようにアプリケーションにメモリが割り当てられるかを決定できる。

【００２１】

好ましい発明実施形態では、障害を起こしたノードは完全に機能しなくなる可能性が高い。これは、ノード自身における、またはノードをシステムの残りの部分に接続する通信ファブリックにおける電気的障害のため、あるいは他の何らかの欠陥のためでありうる。メモリへの電力接続が利用不能でないことがある（たとえばインターコネクトまたはメモリ・コントローラ障害ではメモリへの電源供給は無傷であるが、内容がアクセスできなくなることがある）が、多くのノード障害シナリオにおいては電力が失われる可能性が高い。

【００２２】

いくつかの発明実施形態によれば、ノード障害はこのように、もはやノードへの電力供給がないこと、特にノード・メモリへの電力供給がもはやないことを意味しうる。そのような状況においては、システムの残りの部分から独立な補助電源を提供することが適切であることがある。補助電力は、システムの任意の一部または全部に電力を供給できるが、好ましくは障害を起こしたノードのノード・メモリに給電する。好ましくは、補助電力は、ノード障害があるときに自動的にスイッチ・オンされる。補助電力は、電力障害によってメモリが消し去られないよう、メモリへの電力を連続的に維持するよう作用できる。補

助電源は、いかなる好適な手段によって制御されてもよいが、好ましくは、フェイルオーバー・メモリ・コントローラによって制御される。

【００２３】

発明実施形態の補助電源は、いかなる種類の好適な電源であってもよいが、好ましくはバッテリー（一つまたは複数の電池）である。補助電源はシステムのどこに設けられてもよいが、好ましくは（ノードに付随していても別個に設けられてもよい）フェイルオーバー・メモリ・コントローラと一緒である。複数のフェイルオーバー・メモリ・コントローラがある場合、発明実施形態は、それぞれが独自のフェイルオーバー・メモリ・コントローラにリンクされた同数の補助電源を設けていてもよいし、あるいはそれぞれがいくつかのフェイルオーバー・メモリ・コントローラにリンクされた、より少数の補助電源を設けていてもよい。

【００２４】

補助電源は、任意の時点でフェイルオーバー・メモリ・コントローラに電力を供給できてもよい。これは、ノードの障害の前でもあとでもよく、好ましくは両方である。補助電源がフェイルオーバー・メモリ・コントローラの電源となるとき、フェイルオーバー・メモリ・コントローラのインターコネクト接続および電力を必要としうる他の任意のそのようなインターコネクト接続の電源となることも好ましい。

【００２５】

発明実施形態においては、補助電力は、ノードのノード・メモリに、直接的にまたはプロセッサまたはノード・メモリ・コントローラのような当該ノードの別のコンポーネントを介して供給されてもよい。好ましくは、補助電力は、障害を起こしたノードのノード・メモリに直接供給される。

【００２６】

フェイルオーバー・メモリ・コントローラの、ノードのインターコネクトへの接続は、直接的であっても、または障害を起こしたノードのプロセッサまたはノード・メモリ・コントローラのような別のコンポーネントを介してであってもよい。好ましくは、フェイルオーバー・メモリ・コントローラはインターコネクトに直接結合される。

【００２７】

本コンピュータ・システムは、ノードをモニタリングする（たとえばオペレーティング・システム・レベルでの）（中央）管理プロセスを有していてもよい。そのような状況では、管理プロセスがノード障害を検出できて、置換ノードを同定して、フェイルオーバー・メモリ・コントローラにアプリケーション・データを障害を起こしたノードのノード・メモリから置換ノードのノード・メモリにコピーするよう命令することが望ましい。管理プロセスがアプリケーションを再開することも好ましい。

【００２８】

発明実施形態において、本システムは、一つの補助電源または複数の補助電源を有しうる。システムにおけるノードの数および補助電源の数に依存して、各補助電源は単一のノードのためにまたは一群のノードのために設けられてもよい。

【００２９】

同様に、本システムは、一つのフェイルオーバー・メモリ・コントローラまたは複数のフェイルオーバー・メモリ・コントローラを有していてもよい。システム中のノードの数およびフェイルオーバー・メモリ・コントローラの数に依存して、各フェイルオーバー・メモリ・コントローラは単一のノードのためにまたは一群のノードのために設けられてもよい。

【００３０】

本発明の別の側面の実施形態によれば、それぞれが独自のノード・メモリおよびノード・メモリ・コントローラを有する、インターコネクトによって接続された複数のノードを有するコンピュータ・システム上で実行されるときに、障害を起こしたノードからの回復において使うためのフェイルオーバー・メモリ・コントローラであって、当該フェイルオーバー・メモリ・コントローラは障害を起こしたノードの前記メモリ・コントローラおよ

び／またはメモリに接続するよう動作可能であり；当該フェイルオーバー・メモリ・コントローラは、障害を起こしたノードのノード・メモリに記憶されているアプリケーション・データの、置換ノードのノード・メモリへの、前記インターコネクトを通じた転送を制御するよう構成されている、フェイルオーバー・メモリ・コントローラが提供される。

【００３１】

この側面は、前記コンピュータ・システムの一部として提供されてもよいフェイルオーバー・メモリ・コントローラに関係する。上記で概説したように、可能性としてはノード毎に一つで、複数のフェイルオーバー・メモリ・コントローラがあってもよい。発明実施形態では、その唯一の（またはそれぞれの）フェイルオーバー・メモリ・コントローラは、障害を起こしたノードを含む前記コンピュータ・システム内のノードの一部または全部、好ましくは該ノードの全部に関して自律的であってもよい。しかしながら、フェイルオーバー・メモリ・コントローラは、必要とされるときにノードの諸部分を制御する能力を有することができる。

【００３２】

本発明の別の側面の実施形態によれば、それぞれが独自のノード・メモリおよびノード・メモリ・コントローラを有する複数のノードと；それらのノード上でアプリケーションを実行するときに障害を起こしたノードからの回復において使うためのフェイルオーバー・メモリ・コントローラと；それらのノードおよび前記フェイルオーバー・メモリ・コントローラを接続するインターコネクトとを有するコンピュータ・システムであって、前記フェイルオーバー・メモリ・コントローラは障害を起こしたノードの前記メモリ・コントローラおよび／またはメモリに接続するよう動作可能であり；前記フェイルオーバー・メモリ・コントローラは、障害を起こしたノードのノード・メモリに記憶されているアプリケーション・データの、置換ノードのノード・メモリへの、前記インターコネクトを通じた転送を制御するよう構成されている、コンピュータ・システムが提供される。

【００３３】

前記フェイルオーバー・メモリ・コントローラは、好ましくは、障害を起こしたノードの前記メモリに、また障害を起こしたノードの前記メモリ・コントローラに直接接続される。たとえば前記ノード・メモリ・コントローラが前記ノード・メモリを整合した（consistent）状態のままにし（ノード・メモリ・コントローラが障害を受けたことも考えられるので、これは可能であればである）、次いで前記フェイルオーバー・コントローラに制御を渡すことを保証するためである。

【００３４】

前記コンピュータ・システムはHPCシステムまたはインターコネクトによって接続された分散したメモリおよびノードをもつ他の任意のコンピュータ・システムであってもよい。

【００３５】

コンピュータ・システム実施形態によれば、フェイルオーバー・メモリ・コントローラは、ノードの電力およびインターコネクト接続とは別個であってもなくてもよい、電力接続およびインターコネクト接続を有していてもよい。好ましくは、フェイルオーバー・メモリ・コントローラは、ノード電力およびインターコネクト接続とは別個の電力接続および／またはインターコネクト接続を有する。これは、フェイルオーバー・メモリ・コントローラのより多くの自律性を許容する。好ましい発明実施形態では、フェイルオーバー・メモリ・コントローラへの電源（これは前記補助電源と同じであってもよいが必ずしもそうでなくてもよい）は、すべてのノードの動作状態と独立であり、よって、その後電力を供給するために、いずれか特定のノードがいまだ動作しているかどうかには依拠しない。たいていの場合、フェイルオーバー・メモリ・コントローラは、コンピュータ・システムの残りの部分と、最終的には同じ電源を使ってもよい。発明実施形態は、システム全体が電力を失う状況において使われるよう特に設計されているのではない。

【００３６】

本発明の別の側面の実施形態によれば、フェイルオーバー・メモリ・コントローラおよ

びそれぞれノード・メモリを含む複数のノードを有するコンピュータ・システム上で走るデーモン（ユーザー制御のもとででではなくバックグラウンド・プロセスとして実行されるコンピュータ・プログラム）であって、前記複数のノードおよびフェイルオーバー・メモリ・コントローラはみなインターコネクトによって接続されており、当該デーモンは前記ノード上でのアプリケーションの実行をモニタリングし、当該デーモンはノード障害を検出し、置換ノードを同定し、フェイルオーバー・メモリ・コントローラに、障害を起こしたノードのノード・メモリから置換ノードのノード・メモリに前記インターコネクトを通じてアプリケーション・データをコピーするよう命令する、デーモンが提供される。

【００３７】

事実上、このデーモンは、先述した管理プロセスを提供できる。よって、本デーモンは、オペレーティング・システムの一部として自動的に実行されうる。

【００３８】

さらに一般的なプログラム側面によれば、分散コンピュータ・システムのようなコンピューティング装置にロードされたときに該コンピューティング装置を、上記の方法定義のいずれかまたはその任意の組み合わせに基づく方法段階を実行するよう構成するプログラムが提供される。

【００３９】

本発明の種々の側面の任意のものの特徴およびサブ特徴は自由に組み合わされてもよい。たとえば、フェイルオーバー・メモリ・コントローラおよび／またはコンピュータ・システムの好ましい実施形態は、方法の一つまたは複数の好ましい特徴に対応する機能を組み込むよう構成されてもよい。

【００４０】

本発明は、コンピュータ・ハードウェア、ファームウェア、ソフトウェアまたはそれらの組み合わせにおいて実装されることができる。実施形態は、コンピュータ・プログラムまたはコンピュータ・プログラム・プロダクト、すなわち情報担体において、たとえば非一時的な機械可読記憶デバイスにおいてまたは伝搬する信号において有体に具現されたコンピュータ・プログラムとして実装されることができる。

【００４１】

コンピュータ・プログラムは、コンピュータ・プログラム部分または二つ以上のコンピュータ・プログラムの形であることができ、コンパイル型またはインタープリタ型言語を含む任意の形のプログラミング言語で書くことができ、ライブラリ、スタンドアローン・プログラムとしてまたはモジュール、コンポーネント、サブルーチンまたはデータ処理環境における使用に好適な他のユニットとしてを含め、任意の形で展開できる。

【００４２】

本発明の方法段階は、入力データに対して作用して出力を生成することによって本発明の機能を実行するようコンピュータ・プログラムを実行するプログラム可能なプロセッサによって実行されることができる。

【００４３】

本発明は個別的な実施形態を使って記述されている。他の実施形態が付属の請求項の範囲内である。たとえば、本発明の段階は、異なる順序で実行され、それでいて望ましい結果を達成することができる。

【００４４】

好ましい実施形態に基づく装置は、ある種の機能を実行するよう構成される、動作可能であるまたは配置されるものとして記述される。この構成または配置は、ハードウェアまたはミドルウェアまたは他の任意の好適なシステムの使用によってであることができる。好ましい実施形態では、構成または配置はソフトウェアによる。

【００４５】

本発明についてこれから、図面に示される個別的な限定しない実施形態を参照しつつ説明する。

【図面の簡単な説明】

【００４６】
【図１】障害を起こしたノードの置換を示す概略図である。
【図２】本発明の一般的な実施形態を描くフローチャートである。
【図３】発明実施形態に基づく復元方法の概観である。
【図４】ａは従来技術のノード障害回復のフローチャートであり、ｂは発明実施形態に基づく障害を起こしたノードの回復のフローチャートである。
【図５】発明実施形態に基づく、置換ノードへのデータの移動を示す概略図である。
【図６】発明実施形態に基づく、ノード中のコンポーネントおよび該ノードから出るリンクの概略的なレイアウトを示す図である。
【発明を実施するための形態】
【００４７】
図２は、一般的な発明実施形態を表わすフローチャートである。ステップS10において、計算ノードがアプリケーションを実行する。ステップS20において、アプリケーション・データが生成され、アプリケーションの最も最近の状態がノード・メモリに記憶される。ステップS30においてノードが障害を起こす場合、フェイルオーバー・メモリ・コントローラがステップS40において、障害を起こしたノードのノード・メモリを制御する。最後に、フェイルオーバー・メモリ・コントローラは、ステップS50において、障害を起こしたノードのメモリから置換ノードのメモリにデータをコピーする。
【００４８】
ファイルオーバー・メモリ・コントローラは故障後に使われるだけなので、アプリケーション・データは、データ保存のような冗長なタスクを実行することなく、障害を起こしたノードから回復される。こうして、本方法は、事後反応的なものである。さらに、単一のフェイルオーバー・メモリ・コントローラが、必要であれば複数のノード上のメモリと対話することができる。発明実施形態は、アプリケーション・データ全体をあるノードから別のノードにコピーすることによって、並列に実行しているアプリケーションにおける予期されない障害に対応することができる。
【００４９】
発明実施形態において提案される方法は、コンピュータ・システムにフェイルオーバー・メモリ・コントローラを追加することに関わる。この部分は、一つまたは複数のノード内または各ノード内のメモリ・コントローラの機能と重複することができる。これは、フェイルオーバー・メモリ・コントローラに電力を供給し、障害を起こしたノードのメモリへの電力を維持し、それにより該メモリ内のデータが回復され、次いで置換ノードに転送されることを許容する補助／バックアップ電源に取り付けられてもよい。ノードの障害管理プロセスが提供されてもよい。たとえば、一つのノード上での障害に続いて、管理プロセスが障害を検出し、置換ノードを同定し、フェイルオーバー・メモリ・コントローラに、障害を起こしたノードのメモリに接続してその内容を置換ノードのメモリに直接コピーするよう指示する。これは、（標準的なチェックポイント化に比べ）障害後にアプリケーションを改めて初期化するために必要とされる時間を短縮し、繰り返さなければならない計算の量を最小限にする（プロセッサ・レジスタ内のデータは回復されないので、少量の計算はいまだ繰り返される必要がある）。
【００５０】
置換ノードが割り当てられることのできる二つの主要な方法がある。第一に、アプリケーションは、実際に必要とするより多くのノード（可能性としては、余剰な10%の「スペア」ノード）が割り当てられた状態で立ち上げられることができる。その際、アプリケーションを走らせているノードの一つにおいて障害が検出される（アプリケーション自身によって、あるいは何らかのモニタリング・ソフトウェア・フレームワーク、たとえば前記管理プロセスを通じて）場合、スペア・ノードがリザーブされており、待っていてくれる。
【００５１】
あるいはまた（可能性としては好ましいことに）、システム・ジョブ・スケジューラが

、任意の走っているアプリケーションに割り当てられることのできるスペア・ノードのプールを保持することができる。その場合、ノード障害の検出に続いて、アプリケーション（またはモニタリング・フレームワーク）はジョブ・スケジューラに接触し、スペア・ノードの一つへのアクセスを要求する。このシナリオでは、ジョブ・スケジューラは、新しいジョブが実行を始める際にスペア・ノードの十分大きなプールが利用可能なままであることを保証することを受け持つ。

【００５２】

図３は、だめになったノードからの回復のための方法の具体的な実施形態を示している。図３は、コンピュータ・システム１０内の多くのノード２０のうちの二つを示している。障害を起こした（またはだめになった）ノードは、左側にノード２０Ａとして示されており、新しい／置換ノードは右側にノード２０Ｂとして示されている。各ノードはメモリ６０およびメモリ・コントローラ５０を含む。他のノード部分は当業者に既知であり、図示していない。バッテリー４０およびフェイルオーバー・メモリ・コントローラ３０が両ノードとは別個に、可能性としては同じ物理的支持体の上に設けられている。バッテリーは障害を起こしたノード２０Ａのメモリおよびフェイルオーバー・メモリ・コントローラ３０にリンクされて示されている。バッテリー４０が置換ノード２０Ｂのメモリ６０にリンクされる必要はない。そのノードはいまだ電源に接続されているからである。フェイルオーバー・メモリ・コントローラ３０は、ネットワーク／インターコネクトを通じて置換ノード２０Ｂのメモリ６０にもリンクされている。

【００５３】

よって、フェイルオーバー・メモリ・コントローラおよびバッテリーは、コンピュータ・システムにおいて追加的な部分である。ノード障害の場合、バッテリーが、障害を起こしたノード上のメモリに電力を供給し、フェイルオーバー・メモリ・コントローラ（やはりバッテリーによって電力を受ける）がそのメモリの内容を回復し、障害を起こしたノードの代わりとなることが意図されている新しいノードに転送する。マネージャ７０がこのプロセス（障害を起こしたノードを同定し、フェイルオーバー・メモリ・コントローラを障害を起こしたノードに向け、メモリがどこにコピーされるべきかを指定する）を制御する。

【００５４】

図４のａおよびｂのフローチャートは、回復のない従来技術の障害を起こしたノードと、メモリの内容を新しいホスト・ノードに転送する発明実施形態のフェイルオーバー・メモリ・コントローラを使うノードの回復とを示している。

【００５５】

回復が提供されない場合、図４のａのように、アプリケーションはS110において開始され、ステップS120においてノード障害があり、ステップS130において、障害を起こしたノード上のメモリの内容はアプリケーションにとって失われる。

【００５６】

図４のｂでは、ファイルオーバー・メモリ・コントローラがデータを回復するために使われる。ステップS210において開始したのちすぐ、アプリケーションは、ステップS220において、ノード・メモリのどの部分を使っているか（あるいは等価なことだが、どの部分が他の使用のために予約されていて、アプリケーションのために利用可能でないか）を登録することが必要である。これは、CPU上の全メモリのいくつかの部分は他のプロセス、たとえばオペレーティング・システム（OS）よって使用されるからである。等価な諸プロセスが置換ノード上で走っている可能性が高く、それらのプロセスが使っているメモリを上書きすることは望ましくないであろう（それは、新たなノード上で問題を引き起こす可能性が高く、可能性としては新たなノードまで故障させうる）。このように、アプリケーションによって直接使用されているメモリのみが、障害後に転送されるべきであり、よってこれが早い段階において登録されることが必要である。

【００５７】

この登録に続いて、アプリケーションは、ノード障害S230があるまで走り続ける。その

間、マネージャ・デーモンが各ノードの健全性をモニタリングする（これを行なうためのデーモンはすでに存在しており、さらなる説明はここでは割愛する）。ステップS240における障害の検出に続いて、マネージャはステップS250において（障害を起こしたノードの代わりとするために）新しいホスト・ノードをアプリケーションに割り当て、そのノード上で当該アプリケーションをスタートアップするプロセスを開始し（S270）、フェイルオーバー・メモリ・コントローラに通知する（S260）。通知は、他の動作と並列であることができる。その間、アプリケーションは、残りの諸ノード上で継続されることができる（ただし、実行は最終的には、障害を起こしたノードからのデータを待って同期点においてホールドされる可能性が高い）。復元ステップS280では、障害を起こしたノードのメモリに対して電力が補助電源によって維持され、フェイルオーバー・メモリ・コントローラは前に登録されたメモリのセクションからのデータをコピーし、次いで、置換ノード上のメモリに直接転送する（新しいノード上のメモリ・コントローラの通知を含む）。ひとたびメモリが置換ノードに成功裏にコピーされ終わったら、管理デーモンはこのノード上で実行を再開することができる。

【００５８】

フェイルオーバー・コントローラは一つまたは複数のノードを受け持つことができる。上記のプロセスは、フェイルオーバー・メモリ・コントローラが、受け持ちの各ノード上のメモリおよびメモリ・コントローラに直接接続されることおよびネットワークへのアクセスをもつことを要求する。バッテリーは、ノード上のメモリに接続される必要があり（だが必ずしもメモリ・コントローラに接続される必要はない）、フェイルオーバー・メモリ・コントローラのネットワーク接続に電力を与えてもよい。

【００５９】

図５は、障害を起こしたノード２０Ａ上のメモリ６０から置換ノード２０Ｂ上のメモリ６０へのデータの移動を示している。アプリケーション・データのみが回復される（置換ノードは他のデータ、たとえばOSの、自分自身のインスタンスを有しているので）。この構成は、アプリケーションが、スタートアップ時に該アプリケーションによって使用されるメモリをフェイルオーバー・メモリ・コントローラに登録しておくことを必要とする。ただし、他の方法論があるまたは該アプリケーションにとって利用可能なメモリの設定された部分がある場合にはこの限りではない。

【００６０】

フェイルオーバー・メモリ・コントローラは、各ノードに一対一の追加として（可能性としては同じ回路基板上だが、独自のネットワーク接続および電力供給を含め自律的に作用する）、あるいはそれぞれが一群のプロセッサの（可能性としてはシステム全体の）メモリの回復を受け持つ当該システム内の（一つまたは複数の）別個の諸コンポーネントとして実装されることができる。発明実施形態は、これらの可能性のどれでも同じようにうまく機能する。

【００６１】

図６は、システムのコンポーネント間の物理的な関係を描いている（従来技術に対する相違は太く示されている）。この図は、障害を起こしたノード、管理デーモンおよび新しいノードを示している。

【００６２】

障害を起こした（または古い）ノードはメモリ・コントローラ５０と、メモリ６０と、CPU　８０と、ハードディスクドライブ（HDD）９０とを、ネットワーク・コネクタ１００を介した外部接続とともに有する。このノードはフェイルオーバー・メモリ・コントローラ３０およびバッテリー４０をも含んでいる。フェイルオーバー・メモリ・コントローラ３０はメモリ・コントローラ５０に直接接続され、バッテリー４０はメモリ６０およびフェイルオーバー・メモリ・コントローラ３０に直接接続されている。ネットワーク接続１００は、CPU　８０およびフェイルオーバー・メモリ・コントローラ３０の管理デーモン７０への接続を許容する。

【００６３】

太線内のバッテリーおよびフェイルオーバー・メモリ・コントローラは古いノード内に位置しているように示されているが、そうでなくてもよい。フェイルオーバー・メモリ・コントローラおよびバッテリーは代替的に、システム内の他のところに収容されることもでき（ネットワークに接続されており、ノードのメモリにアクセスするためにCPUおよびメモリ・コントローラをバイパスできる限り）、一つのフェイルオーバー・メモリ・コントローラが二つ以上のノード上のメモリを受け持っていてもよい。

【００６４】

発明実施形態は、次の利点の一部または全部をもちうる。

・アプリケーションの最も最近の可能な状態（標準的なチェックポイントが使われる場合の過去の何らかの距離のところからのバージョンではなく）が回復されるので、ノードの障害後に繰り返される必要のある計算の量が非常に著しく軽減される。

・解における正確さが失われない。これに対し、障害を起こしたノード上で解を生き残った解から補間すると、誤差が増す結果となりうる。

【００６５】

まとめると、好ましい発明実施形態によれば、システム内の処理ユニットから自律的に機能し、それらの処理ユニットのメモリにアクセスできるフェイルオーバー・メモリ・コントローラが最も重要な区別する技術的特徴である。故障したノード上のメモリに電力を供給するためのバッテリーまたは他の補助電源の使用も区別する技術的特徴である。

【００６６】

以上の実施例を含む実施形態に関し、さらに以下の付記を開示する。

（付記１）

インターコネクトによって接続された複数のノードを有するコンピュータ・システムにおいて障害を起こしたノードのメモリからアプリケーション・データを回復し、該アプリケーション・データを置換ノードに書き込む方法であって、

前記コンピュータ・システムのノードがアプリケーションを実行し、該アプリケーションはアプリケーション・データを生成し、該アプリケーションの最も最近の状態をノード・メモリに記憶し；

前記ノードが障害を起こし；

前記障害を起こしたノードのノード・メモリはその後、フェイルオーバー・メモリ・コントローラを使って制御され；

前記フェイルオーバー・メモリ・コントローラは前記アプリケーション・データを前記障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリに前記インターコネクトを通じてコピーする、

方法。

（付記２）

前記アプリケーションが走っており、前記ノード障害の前に、前記アプリケーションはノード・メモリの一部を、前記障害を起こしたノードおよび／または前記置換ノードのノード・メモリにおいて前記アプリケーションによって利用可能または利用不能であると登録する、

付記１記載の方法。

（付記３）

前記アプリケーションが前記一部を前記フェイルオーバー・メモリ・コントローラに登録する、付記２記載の方法。

（付記４）

前記障害を起こしたノードのノード・メモリが前記障害後に補助電力を供給され、補助電源は前記フェイルオーバー・メモリ・コントローラによって制御される、付記１または２記載の方法。

（付記５）

前記補助電源が、前記フェイルオーバー・メモリ・コントローラと一緒に設けられたバッテリーの形である、付記１ないし４のうちいずれか一項記載の方法。

（付記６）
前記補助電源が、前記障害の前およびあとに前記フェイルオーバー・メモリ・コントローラに給電する、付記１ないし５のうちいずれか一項記載の方法。
（付記７）
前記補助電源が、前記フェイルオーバー・メモリ・コントローラのインターコネクト接続に給電する、付記１ないし６のうちいずれか一項記載の方法。
（付記８）
前記補助電力は、前記障害を起こしたノードのプロセッサまたはノード・メモリ・コントローラのような別のコンポーネントを介してではなく、前記ノード・メモリに直接供給される、付記１ないし７のうちいずれか一項記載の方法。
（付記９）
前記フェイルオーバー・メモリ・コントローラは、前記障害を起こしたノードのプロセッサまたはノード・メモリ・コントローラのような別のコンポーネントを介してではなく、前記インターコネクトに直接結合される、付記１ないし８のうちいずれか一項記載の方法。
（付記１０）
前記コンピュータ・システムの管理プロセスがノードをモニタリングし、前記ノード障害を検出し、前記置換ノードを同定し、前記フェイルオーバー・メモリ・コントローラに前記アプリケーション・データを前記障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリにコピーするよう命令する、付記１ないし９のうちいずれか一項記載の方法。
（付記１１）
前記管理プロセスがまた、前記置換ノード上の前記アプリケーションを再開する、付記１０記載の方法。
（付記１２）
補助電源および／またはフェイルオーバー・メモリ・コントローラが単一のノードのためにまたは一群のノードのために設けられる、付記１ないし１１のうちいずれか一項記載の方法。
（付記１３）
それぞれが独自のノード・メモリおよびノード・メモリ・コントローラを有する、インターコネクトによって接続された複数のノードを有するコンピュータ・システム上でアプリケーションを走らせているときに、障害を起こしたノードからの回復において使うためのフェイルオーバー・メモリ・コントローラであって、
当該フェイルオーバー・メモリ・コントローラは前記障害を起こしたノードの前記メモリ・コントローラおよび／またはメモリに接続するよう動作可能であり；
当該フェイルオーバー・メモリ・コントローラは、前記障害を起こしたノードのノード・メモリに記憶されているアプリケーション・データの、置換ノードのノード・メモリへの、前記インターコネクトを通じた転送を制御するよう構成されている、
フェイルオーバー・メモリ・コントローラ。
（付記１４）
前記フェイルオーバー・メモリ・コントローラは、前記コンピュータ・システム内のノードから自律的である、付記１３記載のフェイルオーバー・メモリ・コントローラ。
（付記１５）
それぞれが独自のノード・メモリおよびノード・メモリ・コントローラを有する複数のノードと；
それらのノード上でアプリケーションを実行するときに障害を起こしたノードからの回復において使うためのフェイルオーバー・メモリ・コントローラと；
それらのノードおよび前記フェイルオーバー・メモリ・コントローラを接続するインターコネクトとを有するコンピュータ・システムであって、
前記フェイルオーバー・メモリ・コントローラは障害を起こしたノードの前記メモリ・

コントローラおよび／またはメモリに接続するよう動作可能であり；
前記フェイルオーバー・メモリ・コントローラは、前記障害を起こしたノードのノード・メモリに記憶されているアプリケーション・データの、置換ノードのノード・メモリへの、前記インターコネクトを通じた転送を制御するよう構成されている、
コンピュータ・システム。
（付記１６）
前記フェイルオーバー・メモリ・コントローラが各ノードのために設けられる、
付記１５記載のコンピュータ・システム。
（付記１７）
前記フェイルオーバー・メモリ・コントローラは、ノードの電力およびインターコネクト接続とは別個の電力接続およびインターコネクト接続を有する、
付記１５または１６記載のコンピュータ・システム。
（付記１８）
フェイルオーバー・メモリ・コントローラおよびそれぞれノード・メモリを含む複数のノードを有するコンピュータ・システム上で走るデーモンであって、前記複数のノードおよびフェイルオーバー・メモリ・コントローラはみなインターコネクトによって接続されており、
当該デーモンは前記ノード上でのアプリケーションの実行をモニタリングし、
当該デーモンはノード障害を検出し、置換ノードを同定し、前記フェイルオーバー・メモリ・コントローラに、前記障害を起こしたノードのノード・メモリから前記置換ノードのノード・メモリに前記インターコネクトを通じてアプリケーション・データをコピーするよう命令する、
デーモン。

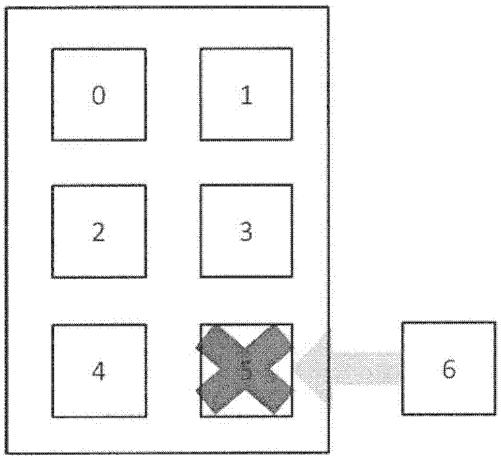
【符号の説明】
【００６７】
S10　ノードがアプリケーションを実行
S20　アプリケーション・データを生成し、アプリケーションの最新の状態をノード・メモリに記憶
S30　ノード障害？
S40　フェイルオーバー・メモリ・コントローラが、障害を起こしたノードのノード・メモリを制御
S50　フェイルオーバー・メモリ・コントローラが、障害を起こしたノードのメモリから置換ノードのノード・メモリにデータをコピー
S110　アプリケーション開始
S120　ノード障害
S130　障害を起こしたノード上のメモリの内容はアプリケーションにとって失われる
S210　アプリケーション開始
S220　アプリケーションがメモリをフェイルオーバー・メモリ・コントローラに登録
S230　ノード障害
S240　ノード障害がデーモンによって検出される
S250　新たなホスト・ノードを開始
S260　フェイルオーバー・メモリ・コントローラを介してメモリを回復
S270　新しいホスト上でアプリケーションを開始
S280　復旧

２０Ａ　ノード
２０Ｂ　新しいノード
３０　フェイルオーバー・メモリ・コントローラ
４０　バッテリー

５０　メモリ・コントローラ
６０　メモリ
７０　マネージャ（管理デーモン）
８０　CPU
９０　HDD
１００　ネットワーク

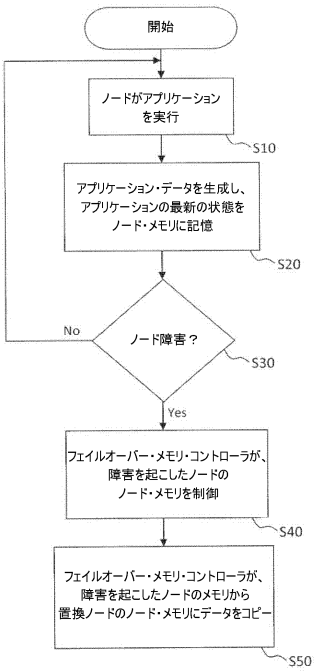
【図１】

障害を起こしたノードの置換を示す概略図



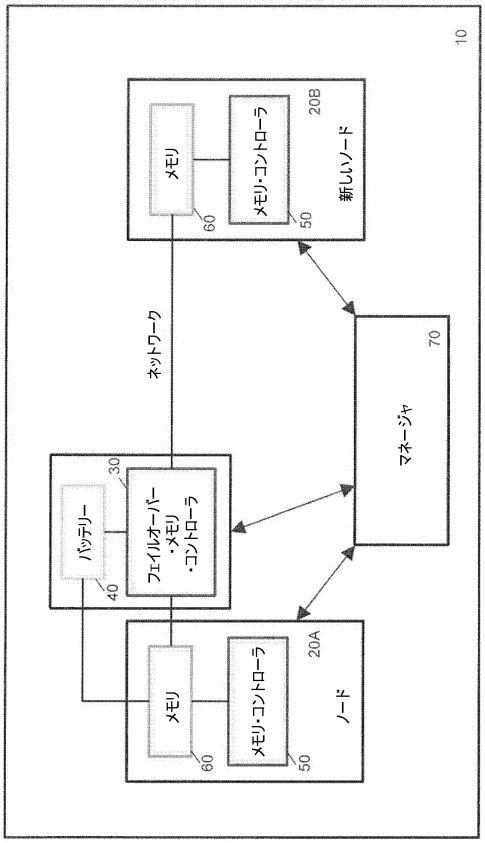
【図２】

本発明の一般的な実施形態を描くフローチャート



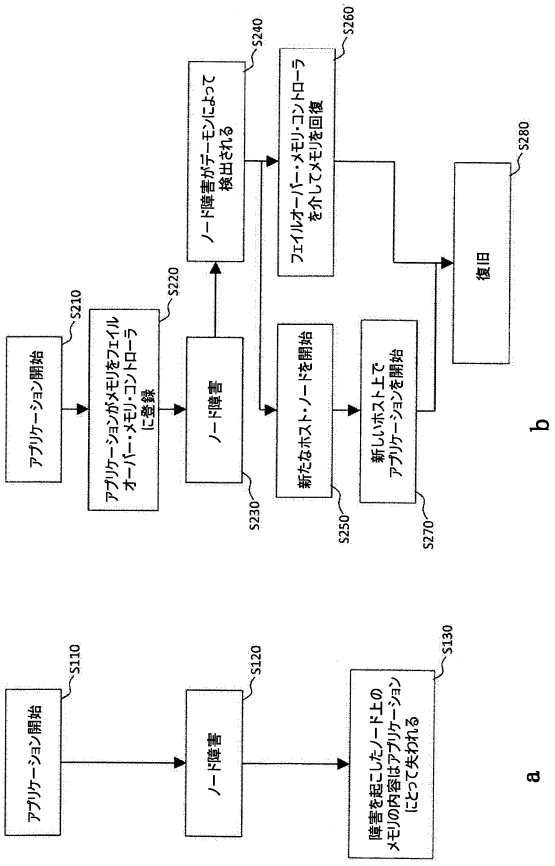
【図３】

発明実施形態に基づく復元方法の概観図



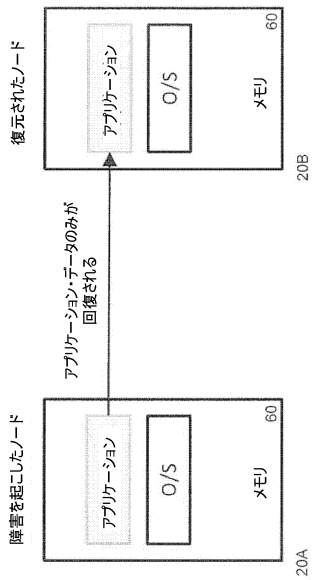
【図４】

aは従来技術のノード障害回復のフローチャート、
bは発明実施形態に基づく障害を起こしたノードの回復のフローチャート



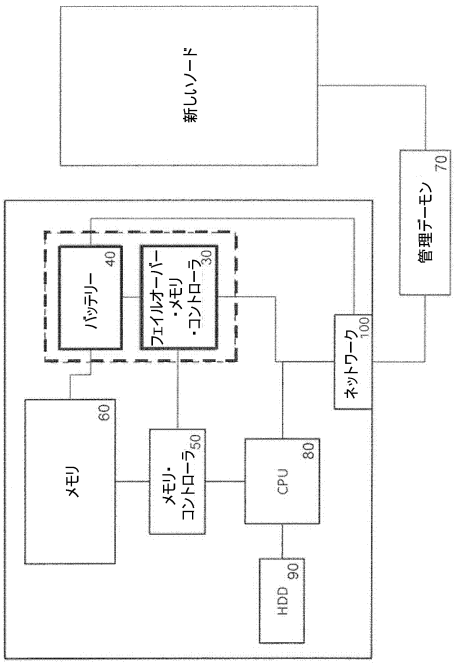
【図５】

発明実施形態に基づく、置換ノードへのデータの移動を示す概略図



【図６】

発明実施形態に基づく、ノード中のコンポーネント
および該ノードから出るリンクの概略的なレイアウトを示す図



フロントページの続き

(72)発明者　ウィルソン・ニコラス
イギリス国，ユービー７　７ユージー，ミドルセックス，ウエスト　ドレイトン，ローワン　ロード　１８２番

審査官　滝谷　亮一

(56)参考文献　特開２００６－３０９２９２（ＪＰ，Ａ）
特開昭６２－００５７５９（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１１／２０