

(19) 日本国特許庁(JP)

(12) 公 開 特 許 公 報(A)

(11) 特許出願公開番号
特開2004-355233
(P2004-355233A)

(43) 公開日 平成16年12月16日(2004. 12. 16)

(51) Int.Cl.⁷
G06F 11/18
G06F 15/17

F I
G O 6 F 11/18 3 1 O C
G O 6 F 15/17 6 7 8 A

テーマコード (参考)
5 B O 3 4
5 B O 4 5

審査請求 未請求 請求項の数 30 O L (全 41 頁)

(21) 出願番号	特願2003-150932 (P2003-150932)	(71) 出願人	000004237
(22) 出願日	平成15年5月28日 (2003. 5. 28)		日本電気株式会社
			東京都港区芝五丁目7番1号
(特許庁注：以下のものは登録商標)		(74) 代理人	100088959
L i n u x			弁理士 境 廣巳
		(72) 発明者	藤山 健一郎
			東京都港区芝五丁目7番1号 日本電気株
			式会社内
		(72) 発明者	中村 暢達
			東京都港区芝五丁目7番1号 日本電気株
			式会社内
		F ターム (参考)	5B034 AA04
			5B045 GG11 JJ02 JJ03 JJ07 JJ08
			JJ13 JJ23 JJ26

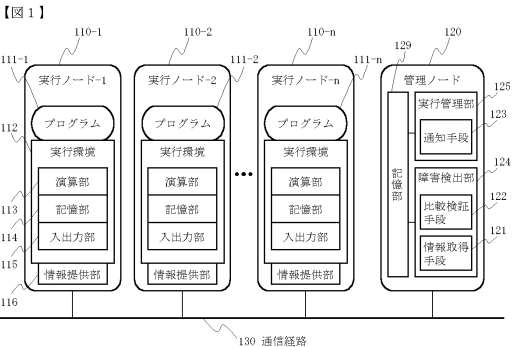
(54) 【発明の名称】耐障害システム、プログラム並列実行方法、耐障害システムの障害検出装置およびプログラム

(57) 【要約】

【課題】複数のプログラムを複数の実行ノードで並列実行する耐障害システムにおいて、異常な実行ノードを速やかに検出する。

【解決手段】同一の実行環境112を持つ複数の実行ノード110-1～110-n上で、同等の機能を持つが異なるバージョンのプログラム111-1～111-nを並列実行させることでプログラムレベルでの耐障害性を持たせた耐障害システムにおいて、異常な実行ノードが存在しない状態では、各実行ノードの環境情報の違いはプログラムのバージョンの相違に起因する差異に過ぎないが、何れかの実行ノードが異常になると、その異常になった実行ノードの環境情報だけが他の正常な実行ノードの環境情報と明確に差が生じる。情報取得手段121は、プログラムを並列実行中の複数の実行ノードの環境情報を取得し、比較検証手段122はこれら複数の環境情報を比較して異常ノードを検出する。

【選択図】 図1



【特許請求の範囲】

【請求項 1】

同一の実行環境上でバージョンが異なるプログラムを実行することにより同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備えることを特徴とする耐障害システム。

【請求項 2】

それぞれ異なる実行環境上で同一のプログラムを実行することにより同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備えることを特徴とする耐障害システム。 10

【請求項 3】

それぞれ異なる実行環境上でバージョンが異なるプログラムを実行することにより同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備えることを特徴とする耐障害システム。

【請求項 4】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備えることを特徴とする耐障害システム。 20

【請求項 5】

前記障害管理手段は、前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを含むことを特徴とする請求項 1 乃至 4 の何れか 1 項に記載の耐障害システム。

【請求項 6】

前記検出された異常な実行ノードを停止させる停止手段を備えることを特徴とする請求項 5 記載の耐障害システム。

【請求項 7】

前記複数の実行ノードの安定性を判定する安定性判定手段と、前記安定性判定手段で前記複数の実行ノードが安定していると判定されたときに既定数を超える数の実行ノードを停止対象として選択する選択手段とを備え、前記停止手段は前記選択手段で選択された実行ノードを停止させるものであることを特徴とする請求項 6 記載の耐障害システム。 30

【請求項 8】

異常な実行ノードが検出されたときに、新たな実行ノードを起動する追加手段を備えることを特徴とする請求項 5、6 または 7 記載の耐障害システム。

【請求項 9】

前記複数の実行ノード上の前記プログラムに同一の処理要求を分配する処理要求分配手段と、前記複数の実行ノードで実行された前記プログラムの処理結果からシステムとしての処理結果を導出する処理結果導出手段とを備えることを特徴とする請求項 1 乃至 8 の何れか 1 項に記載の耐障害システム。 40

【請求項 10】

前記処理要求分配手段は、処理サービスの要求メッセージを受信する手段と、その要求メッセージをシステム内で処理するかどうか判定する手段と、処理すると判定したら、転送先の実行ノードに転送するためのメッセージを生成する手段と、生成したメッセージを実行ノードに送信する手段と、システムの状態に応じて、判断基準となる情報を更新する手段とを含むことを特徴とする請求項 9 記載の耐障害システム。

【請求項 11】

前記処理結果導出手段は、処理サービスの応答もしくはデータベースアクセスなどの処理途中の別サービス要求メッセージを受信する手段と、その応答メッセージもしくは別サー 50

ビス要求メッセージをどのように処理するか判定し、判定に基づきメッセージの選択もしくは生成を行う手段と、応答メッセージもしくは別サービス要求メッセージをサービス要求元もしくは別サービス要求先に転送するために転送メッセージを生成する手段と、生成したメッセージをサービス要求元もしくは別サービス要求先に送信する手段と、システムの状態に応じて、判断基準となる情報を更新する手段とを含むことを特徴とする請求項10記載の耐障害システム。

【請求項12】

前記複数の実行ノードに通信経路を通じて接続された少なくとも1つの管理ノードを備え、前記管理ノードに前記障害管理手段を備えることを特徴とする請求項1乃至4の何れか1項に記載の耐障害システム。

10

【請求項13】

前記複数の実行ノードのそれぞれに前記障害管理手段を備えることを特徴とする請求項1乃至4の何れか1項に記載の耐障害システム。

【請求項14】

前記実行ノードに実マシンを用いることを特徴とする請求項1乃至13の何れか1項に記載の耐障害システム。

【請求項15】

前記実行ノードに仮想マシンを用いることを特徴とする請求項1乃至13の何れか1項に記載の耐障害システム。

【請求項16】

複数の実行ノードの環境情報を比較検証することで、異常な実行ノードを検出することを特徴とする耐障害システム。

20

【請求項17】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードに通信経路を通じて接続され、前記複数のプログラムを並列実行中の前記複数の実行ノードから環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備えることを特徴とする管理ノード。

【請求項18】

前記障害管理手段は、前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを含むことを特徴とする請求項17記載の管理ノード。

30

【請求項19】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する1つ以上の他実行ノードに通信経路を通じて接続され、前記1つ以上の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し且つ管理ノードとしても機能する実行ノードであって、前記1つ以上の他実行ノードおよび自ノードから、前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備えることを特徴とする実行ノード。

【請求項20】

前記障害管理手段は、前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを含むことを特徴とする請求項19記載の実行ノード。

40

【請求項21】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する1つ以上の他実行ノードおよび前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備える管理ノードに通信経路を通じて接続され、前記1つ以上の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し、且つ、前記管理ノードの前記障害管理手段に対して、自ノ

50

ードの環境情報を前記通信経路を通じて提供する情報提供手段を備えることを特徴とする実行ノード。

【請求項 2 2】

前記障害管理手段は、前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを含むことを特徴とする請求項 2 1 記載の実行ノード。

【請求項 2 3】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の他実行ノードに通信経路を通じて接続され、前記複数の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し、且つ、前記複数の他実行ノードのうち前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する管理ノードとして機能する他実行ノードの障害管理手段に対して、自ノードの環境情報を前記通信経路を通じて提供する情報提供手段を備えることを特徴とする実行ノード。

10

【請求項 2 4】

前記障害管理手段は、前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを含むことを特徴とする請求項 2 3 記載の実行ノード。

【請求項 2 5】

複数の実行ノードを備える計算機システムにおけるプログラム並列実行方法において、
(a) プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードで、前記プログラムを並列実行するステップ
(b) 前記複数の実行ノードの前記プログラムに同一の処理要求を処理させるステップ
(c) 前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出するステップ
(d) 前記複数の計算ノードで並列実行される前記複数のプログラムの前記処理要求に対する処理結果からシステムとしての処理結果を導出するステップ
を含むことを特徴とするプログラム並列実行方法。

20

30

【請求項 2 6】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードで前記複数のプログラムが並列実行されているときに前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを備えることを特徴とする耐障害システムの障害検出装置。

【請求項 2 7】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードを備える計算機システムを、前記複数のプログラムが並列実行されているときに前記複数の実行ノードの環境情報を取得する情報取得手段、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段、として機能させる耐障害システムプログラム。

40

【請求項 2 8】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードに通信経路を通じて接続された計算機を、前記複数のプログラムを並列実行中の前記複数の実行ノードから環境情報を取得して比較し異常な実行ノードを検出する障害管理手段、として機能させる管理ノード用プログラム。

【請求項 2 9】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互い

50

に相違するが同等の機能を提供する1つ以上の他実行ノードに通信経路を通じて接続され、前記1つ以上の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し且つ管理ノードとしても機能する計算機を、前記1つ以上の他実行ノードおよび自ノードから、前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段、として機能させる実行ノード用プログラム。

【請求項30】

プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する1つ以上の他実行ノードおよび前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備える管理ノードに通信経路を通じて接続され、前記1つ以上の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供する計算機を、前記管理ノードの前記障害管理手段に対して、自ノードの環境情報を前記通信経路を通じて提供する情報提供手段、として機能させる実行ノード用プログラム。

10

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は実行ノードを複数備えた計算機システムに関し、特にプログラムレベルでの耐障害性能を高めた耐障害システムに関する。

20

【0002】

【従来の技術】

耐障害システムは、構成要素の故障は避けられないという観点から、実行ノードの冗長化によって信頼性を高めた計算機システムである。例えば、後述する非特許文献1には、プロセッサとメモリからなる多数の実行ノードをバスに接続し、全ての処理を適当な大きさのタスクに分割して、各タスクを複数の実行ノードで同時に実行し、各タスクの実行結果から多数決によりシステムの出力を導出すると共に異常な実行ノードを検出する耐障害システムが示されている。

【0003】

類似する耐障害システムは、特許文献1にも記載されている。但し、特許文献1に記載されるシステムでは、クライアントがそれぞれ異なる複数のサーバに対して処理要求を出し、クライアント自身が複数のサーバから返される結果から多数決によって正しい結果を求めている。また、クライアントから複数のサーバに対して、処理要求を送ると同時に浮動小数点演算やファイルアクセス等の試験手続きの実行要求を送信し、各サーバから返されてくる処理要求に対する結果だけでなく試験手続きの実行結果をも比較して正しい処理結果の導出と障害サーバの検出を行う方法が記載されている。

30

【0004】

非特許文献1に記載される耐障害システムは、主にハードウェアレベルでの耐障害性の向上を狙ったものであり、実行ノードは全て同じ構成を有し、タスクは同一のプログラムにより各実行ノードで並列実行することを前提としている。このため、プログラムにバグがあると、並列実行された全てのプログラムに異常が発生する確率が高く、ソフトウェアレベルでの耐障害性の向上は難しい。大規模かつ複雑化の一途をたどるプログラムに、バグの存在しない完全性を求めることは、事実上困難であるため、プログラムレベルで耐障害性を持たせる必要がある。

40

【0005】

プログラムにバグが存在していても、障害が外部に影響を与えない、ソフトウェアレベルでの耐障害性を向上させる従来技術として、Nバージョン方式やその改良方式、およびリカバリブロック方式が知られている。

【0006】

Nバージョン方式は、後述する非特許文献2に記載されているように、複数の異なる開発

50

者によってそれぞれ異なる手法で作成された同一の機能を達成する複数のバージョン（版）のプログラムを計算機システムで並列に実行し、複数のプログラムの出力結果から多数決によりシステムの出力を導出する方式である。

【0007】

Nバージョンを改良した方式は、後述する特許文献2に記載されるように、1つのプログラムを機能単位で幾つかのモジュールに分割し、各モジュール毎に複数のバージョンの異なるモジュールを開発し、各モジュールを組み合わせることで、低い開発コストで、より多くのバージョンの異なるプログラムを開発する手法である。例えば、あるプログラムを{A、B、C}の3つのモジュールに分割し、各モジュール毎に2種類ずつのバージョンの異なるモジュールA1とA2、B1とB2、C1とC2を開発し、これらを組み合わせ

10

【0008】

一方、リカバリブロック方式は、後述する非特許文献3に記載されているように、プログラムの実行結果の正当性を検証する方式である。具体的には、Nバージョン方式と同じく、複数の異なるバージョンのプログラムを作成し、それぞれに順位づけを行う。第1順位のバージョンのプログラムの実行結果を、アクセプタンステスト（Acceptance Test：受け入れテスト）と呼ばれる予め定められた検証論理により検証し、検証結果が合格ならば、その実行結果を選択する。検証結果が不合格ならば、第2順位のバージョンのプログラムの実行結果を選択する。

20

【0009】

このように、従来の耐障害システムは、複数の異なるバージョンのプログラムによる多重実行によって、システムとしての正しい処理結果の導出と障害ノードの検出を行っている。

【0010】

他方、耐障害システムに限らず通常の計算機システムにおいても、ハートビート信号を用いて実行ノードが正常に動作しているかどうかを定期的に調べるping-pong方式などによって、システムに存在する個々のノードの障害発生の有無を検出しており、障害の発生したノードを停止するなどの障害対応処理を実施している。

30

【0011】

【非特許文献1】

社団法人情報処理学会編，「新版 情報処理ハンドブック」，第1版，株式会社オーム社，平成7年11月25日，p459－461

【非特許文献2】

「Fault tolerance by design diversity: Concepts and experiments」(Avizienis, A., Kelly, J. P. J., IEEE Computer vol. 17, no. 8, pp: 67-80, 1984.)

40

【非特許文献3】

「Distributed Execution of Recovery Blocks: An Approach for Uniform Treatment of Hardware and Software Faults in Real-Time Applications」(K. H. Kim, Howard O. Welch, IEEE Transactions on Computers, vol. 38, no. 5, pp: 626-636, 1989)

【特許文献1】

特開平7-306794号公報

50

【特許文献 2】

特開平 6-342369 号公報

【0012】

【発明が解決しようとする課題】

上述したようにプログラムレベルでの耐障害性能を高めた従来の耐障害システムにおいては、同等の機能を持つ複数のプログラムを並列実行し、それら複数のプログラムの実行結果から多数決方法やアクセプタンステストによる検証などによってシステムとしての正しい実行結果を導出すると同時に障害ノードを検出している。従って、最終的に実行結果が出揃うまで障害ノードを検出できず、障害ノードの速やかな検出が行えないという課題がある。ping-pong方式を用いて障害ノードの検出を並行して実施しても、全くpong応答が無くなるノードダウンといった障害は検出できるが、処理結果が他のノードと相違することになるような異常な実行ノードを早期に検出することは困難である。

【0013】

異常な実行ノードの早期検出が困難であると、異常な実行ノードの停止が遅れ、リソースの無駄な消費によって実行コストが上昇すると共に、異常な実行ノードが停止されずに動作し続けることによって他の正常な実行ノードに障害が波及し、耐障害性を劣化させる恐れがある。

【0014】

本発明の目的は、複数のプログラムを複数の実行ノードで並列実行する耐障害システムにおいて、異常な実行ノードの速やかな検出を可能にすることにある。

【0015】

本発明の別の目的は、異常な実行ノードが動作し続けることによる悪影響を速やかに除去することにある。

【0016】

本発明の他の目的は、耐障害システムの開発コスト、実行コストを低減することにある。

【0017】

【課題を解決するための手段】

本発明の耐障害システムは、複数の実行ノードの環境情報を比較検証することで、異常な実行ノードを検出することを基本とする。ここで、環境情報とは、プロセッサの使用状態、稼動中のプロセスの実行状態、メモリの使用状態、キャッシュの使用状態、仮想記憶の使用状態、スワップファイルの使用状態、ディスクの使用状態、入出力インタフェースの状態、ネットワークの状態、電源供給の有無、各デバイスの温度などの1つ又は複数あるいは全部の情報のことである。本発明の耐障害システムは、具体的には以下のような構成を備える。

【0018】

本発明の第1の耐障害システムは、同一の実行環境上でバージョンが異なるプログラムを実行することにより同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備える。

【0019】

本発明の第2の耐障害システムは、それぞれ異なる実行環境上で同一のプログラムを実行することにより同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備える。

【0020】

本発明の第3の耐障害システムは、それぞれ異なる実行環境上でバージョンが異なるプログラムを実行することにより同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備える。

【0021】

本発明の第４の耐障害システムは、プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードと、前記複数のプログラムを並列実行中の前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段とを備える。

【００２２】

本発明の第５の耐障害システムは、第１乃至第４の耐障害システムにおいて、前記障害管理手段は、前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を比較して異常な実行ノードを検出する比較検証手段とを含んで構成される。

【００２３】

本発明の第６の耐障害システムは、第５の耐障害システムにおいて、前記検出された異常な実行ノードを停止させる停止手段を備える。

【００２４】

本発明の第７の耐障害システムは、第６の耐障害システムにおいて、前記複数の実行ノードの安定性を判定する安定性判定手段と、前記安定性判定手段で前記複数の実行ノードが安定していると判定されたときに既定数を超える数の実行ノードを停止対象として選択する選択手段とを備え、前記停止手段は前記選択手段で選択された実行ノードを停止させるようにしている。

【００２５】

本発明の第８の耐障害システムは、第５、第６または第７の耐障害システムにおいて、異常な実行ノードが検出されたときに、新たな実行ノードを起動する追加手段を備える。

【００２６】

本発明の第９の耐障害システムは、第１乃至第８の耐障害システムにおいて、前記複数の実行ノード上の前記プログラムに同一の処理要求を分配する処理要求分配手段と、前記複数の実行ノードで実行された前記プログラムの処理結果からシステムとしての処理結果を導出する処理結果導出手段とを備える。

【００２７】

本発明の第１０の耐障害システムは、第９の耐障害システムにおいて、前記処理要求分配手段は、処理サービスの要求メッセージを受信する手段と、その要求メッセージをシステム内で処理するかどうか判定する手段と、処理すると判定したら、転送先の実行ノードに転送するためのメッセージを生成する手段と、生成したメッセージを実行ノードに送信する手段と、システムの状態に応じて、判断基準となる情報を更新する手段とを含んで構成される。

【００２８】

本発明の第１１の耐障害システムは、第１０の耐障害システムにおいて、前記処理結果導出手段は、処理サービスの応答もしくはデータベースアクセスなどの処理途中の別サービス要求メッセージを受信する手段と、その応答メッセージもしくは別サービス要求メッセージをどのように処理するか判定し、判定に基づきメッセージの選択もしくは生成を行う手段と、応答メッセージもしくは別サービス要求メッセージをサービス要求元もしくは別サービス要求先に転送するために転送メッセージを生成する手段と、生成したメッセージをサービス要求元もしくは別サービス要求先に送信する手段と、システムの状態に応じて、判断基準となる情報を更新する手段とを含んで構成される。

【００２９】

本発明の第１２の耐障害システムは、第１乃至第４の耐障害システムにおいて、前記複数の実行ノードに通信経路を通じて接続された少なくとも１つの管理ノードを備え、前記管理ノードに前記障害管理手段を備える。

【００３０】

本発明の第１３の耐障害システムは、第１乃至第４の耐障害システムにおいて、前記複数の実行ノードのそれぞれに前記障害管理手段を備える。

【００３１】

10

20

30

40

50

本発明の第14の耐障害システムは、第1乃至第13の耐障害システムにおいて、前記実行ノードに実マシンを用いる。

【0032】

本発明の第15の耐障害システムは、第1乃至第13の耐障害システムにおいて、前記実行ノードに仮想マシンを用いる。

【0033】

本発明の第1の管理ノードは、プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードに通信経路を通じて接続され、前記複数のプログラムを並列実行中の前記複数の実行ノードから環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備える。

10

【0034】

本発明の第2の管理ノードは、プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する1つ以上の他実行ノードに通信経路を通じて接続され、前記1つ以上の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し且つ管理ノードとしても機能する実行ノードであって、前記1つ以上の他実行ノードおよび自ノードから、前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備える。

【0035】

本発明の第1の実行ノードは、プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する1つ以上の他実行ノードおよび前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する障害管理手段を備える管理ノードに通信経路を通じて接続され、前記1つ以上の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し、且つ、前記管理ノードの前記障害管理手段に対して、自ノードの環境情報を前記通信経路を通じて提供する情報提供手段を備える。

20

【0036】

本発明の第2の実行ノードは、プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の他実行ノードに通信経路を通じて接続され、前記複数の他実行ノードとプログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が相違するが同等の機能を提供し、且つ、前記複数の他実行ノードのうち前記複数のプログラムを並列実行中の実行ノードの環境情報を取得して比較し異常な実行ノードを検出する管理ノードとして機能する他実行ノードの障害管理手段に対して、自ノードの環境情報を前記通信経路を通じて提供する情報提供手段を備える。

30

【0037】

また、本発明のプログラム並列実行方法は、複数の実行ノードを備える計算機システムにおけるプログラム並列実行方法において、(a)プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードで、前記プログラムを並列実行するステップ、(b)前記複数の実行ノードの前記プログラムに同一の処理要求を処理させるステップ、(c)前記複数の実行ノードの環境情報を取得して比較し異常な実行ノードを検出するステップ、(d)前記複数の計算ノードで並列実行される前記複数のプログラムの前記処理要求に対する処理結果からシステムとしての処理結果を導出するステップ、を含んで構成される。

40

【0038】

また、本発明の耐障害システムの障害検出装置は、プログラムのバージョン及び前記プログラムを実行する実行環境の少なくとも一方が互いに相違するが同等の機能を提供する複数の実行ノードで前記複数のプログラムが並列実行されているときに前記複数の実行ノードの環境情報を取得する情報取得手段と、取得された前記複数の実行ノードの環境情報を

50

比較して異常な実行ノードを検出する比較検証手段とを備える。

【0039】

【作用】

全く同一の実行環境を持つ複数の実行ノード上で全く同一のプログラムを並列実行させた場合、各実行ノードのプロセッサの使用状態やメモリの使用状態などの環境情報はほぼ全く等しくなる。

【0040】

これに対して、第1の耐障害システムのように、全く同一の実行環境を持つ複数の実行ノード上で、同等の機能を持つが異なるバージョンのプログラムを並列実行させた場合、異常となる実行ノードが発生していない状態では、各実行ノードの環境情報の違いはプログラムのバージョンの相違に起因する差異に過ぎないが、若し、何れかの実行ノードが異常になると、その異常になった実行ノードの環境情報だけが他の正常な実行ノードの環境情報と、プログラムのバージョンの違いに起因する差以上に明確に差が生じる。従って、プログラムを並列実行中の複数の実行ノードの環境情報を取得して比較することにより、実行結果が出揃う前に異常なノードの検出が行える。

10

【0041】

また、第2の耐障害システムのように、それぞれ異なる実行環境を持つ複数の実行ノード上で、同一のプログラムを並列実行させた場合、異常となる実行ノードが発生していない状態では、各実行ノードの環境情報の違いは実行環境の相違に起因する差異に過ぎないが、若し、何れかの実行ノードが異常になると、その異常になった実行ノードの環境情報だけが他の正常な実行ノードの環境情報と、実行環境の違いに起因する差以上に明確に差が生じる。従って、プログラムを並列実行中の複数の実行ノードの環境情報を取得して比較することにより、実行結果が出揃う前に異常なノードの検出が行える。

20

【0042】

また、第3の耐障害システムのように、それぞれ異なる実行環境を持つ複数の実行ノード上で、それぞれ異なるバージョンのプログラムを並列実行させた場合、異常となる実行ノードが発生していない状態では、各実行ノードの環境情報の違いはプログラムのバージョンおよび実行環境の相違に起因する差異に過ぎないが、若し、何れかの実行ノードが異常になると、その異常になった実行ノードの環境情報だけが他の正常な実行ノードの環境情報と、プログラムのバージョンおよび実行環境の違いに起因する差以上に明確に差が生じる。従って、プログラムを並列実行中の複数の実行ノードの環境情報を取得して比較することにより、実行結果が出揃う前に異常なノードの検出が行える。

30

【0043】

また、第4の耐障害システムのように、プログラムのバージョン及び実行環境の少なくとも一方が互いに相違する複数の実行ノードで前記複数のプログラムを並列実行させた場合、異常となる実行ノードが発生していない状態では、各実行ノードの環境情報の違いはプログラムのバージョンや実行環境の相違に起因する差異に過ぎないが、若し、何れかの実行ノードが異常になると、その異常になった実行ノードの環境情報だけが他の正常な実行ノードの環境情報と、プログラムのバージョンおよび／または実行環境の違いに起因する差以上に明確に差が生じる。従って、プログラムを並列実行中の複数の実行ノードの環境情報を取得して比較することにより、実行結果が出揃う前に異常なノードの検出が行える。

40

【0044】

【発明の第1の実施の形態】

図1参照すると、本発明の第1の実施の形態にかかる耐障害システムは、複数の実行ノード110-1～110-nと、これら複数の実行ノード110-1～110-nを管理する少なくとも1つの管理ノード120と、これら複数の実行ノード110-1～110-nおよび管理ノード120を互いに通信可能に接続する通信経路130とを含んで構成される。

【0045】

50

実行ノード 110-1~110-n は、プログラム 111-1~111-n と、そのプログラム 111-1~111-n を実行する実行環境 112 と、自ノードの環境情報を管理ノード 120 に提供する情報提供部 116 とを有する。

【0046】

本実施の形態において、各実行ノード 110-1~110-n の実行環境 112 はすべて同一であり、少なくとも 1 つの演算部 113 と記憶部 114 と入出力部 115 とを備えている。情報提供部 116 は、自ノードにおける実行環境 112 が備える演算部 113 や記憶部 114 や入出力部 115 などの状態を示す環境情報を管理ノード 120 に提供する。

【0047】

プログラム 111-1~111-n は、管理ノード 120 から与えられる処理要求を入力して要求される処理を実行し、その処理結果を管理ノード 120 に返却するアプリケーションプログラムである。本実施の形態においては、各実行ノード 110-1~110-n のプログラム 111-1~111-n は、同等の機能を提供するプログラムであるが、前述した N バージョン方式で 사용되는プログラムのように、互いにバージョン（版）が相違している。 10

【0048】

管理ノード 120 は、異常な実行ノードの検出などシステムの障害を検出する障害検出部 124 と、各実行ノード 110-1~110-n のプログラム 111-1~111-n に同一の処理要求を分配して並列に処理させる処理要求分配処理、障害検出部 124 で検出された障害に対応する障害対応処理、プログラム 111-1~111-n の処理結果からシステムとしての処理結果を導出する処理結果導出処理など、システム全体の実行状態を管理する実行管理部 125 と、種々のデータを記憶する記憶部 129 とを備える。特に本実施の形態では、検出した異常な実行ノードを外部に通知する通知手段 123 が実行管理部 125 に設けられている。 20

【0049】

障害検出部 124 は、各実行ノード 110-1~110-n における実行環境 112 が備える演算部 113、記憶部 114 および入出力部 115 などの状態を示す環境情報を、各実行ノード 110-1~110-n の情報提供部 116 から取得して記憶部 129 に記憶する情報取得手段 121 と、この情報取得手段 121 によって取得された各実行ノード 110-1~110-n における実行環境 112 の環境情報を記憶部 129 から読み出して比較検証し、その検証結果を記憶部 129 に記憶する比較検証手段 122 とを含んで構成される。 30

【0050】

通信経路 130 は、バス、シリアル、パラレル、LAN、無線 LAN、インターネット、公衆回線などの任意の通信回線であり、また複数を組み合わせた多重化通信回線であっても良い。

【0051】

ここで、個々の実行ノード 110-1~110-n および管理ノード 120 は、例えば 1 台のパーソナルコンピュータ（PC）や大型計算機の 1 つのセルなど、物理的に存在する実マシンであっても良いし、仮想マシンであっても良い。また、全ての実行ノード 110-1~110-n および管理ノード 120 を、全て実マシンで実現しても良いし、全て仮想マシンで実現しても良く、一部の実行ノードは仮想マシンで、残りの実行ノードは実マシンで実現する如く、仮想マシンと実マシンとが混在していても良い。 40

【0052】

仮想マシンは、図 2 の仮想マシン 211 に示されるように、物理的な計算機 212 とその上で実行される仮想マシンモニタと呼ばれる制御ソフトウェア 213 とで実現される仮想マシン実行環境 214 を利用して、ソフトウェアで実現される計算機であり、計算機シミュレータあるいは計算機エミュレータとも呼ばれる。仮想マシンは、実マシンと論理的には全く同じ動作をする。同じ仮想マシン実行環境 214 上に 1 つまたは複数の仮想マシン 211 を実現することができ、各々の仮想マシン 211 の機器構成は、その仮想マシンを 50

実現している計算機 2 1 2 上にあるファイルで指定される。仮想マシン 2 1 1 が備える演算部や記憶部や入出力部は仮想資源と呼ばれる。制御ソフトウェア 2 1 3 の最も基本的な機能は、仮想マシン 2 1 1 が備える仮想資源と計算機 2 1 2 が備える実資源とのマッピング機能である。このように仮想マシンはソフトウェアで実現されているので、必要な時にのみ存在する。つまり、必要な時に仮想マシンは生成され、不要となると消去される。そのため、実行ノード 1 1 0 - 1 ~ 1 1 0 - n に仮想マシンを使用すると、多数の計算機および機器を常時用意する必要がなく、コストの削減が可能となる。なお、実行ノード 1 1 0 - 1 ~ 1 1 0 - n のうちの全部または一部を仮想マシン 2 1 1 で実現する場合、それらは同じ仮想マシン実行環境 2 1 4 上に生成されたものであっても良いし、別の仮想マシン実行環境 2 1 4 上に生成されたものであっても良い。

10

【0053】

次に、図 3 のフローチャートを参照して、本実施の形態の耐障害システムにおけるシステム起動時の動作を説明する。

【0054】

システムが起動されると、複数の実行ノード 1 1 0 - 1 ~ 1 1 0 - n が同じ実行環境 1 1 2 でバージョンの異なるプログラム 1 1 1 - 1 ~ 1 1 1 - n が並列に実行されるように、各実行ノード 1 1 0 - 1 ~ 1 1 0 - n に実行環境 1 1 2 およびプログラム 1 1 1 - 1 ~ 1 1 1 - n の設定が行われる (S 1 0 1 ~ S 1 0 4)。具体的には、記憶部 1 2 9 には各実行ノード 1 1 0 - 1 ~ 1 1 0 - n 毎の実行環境設定データ、並列実行対象となるプログラム 1 1 1 - 1 ~ 1 1 1 - n、システム起動時に起動すべき実行ノードの数と起動対象とする実行ノードの指定情報などが予め記憶されている。管理ノード 1 2 0 の実行管理部 1 2 5 は、内部の変数 N を 1 に初期設定し (S 1 0 1)、1 番目の実行ノード 1 1 0 - 1 に注目し、その実行ノード 1 1 0 - 1 の実行環境設定データ等を記憶部 1 2 9 から読み込んで、実行環境 1 1 2 を設定する (S 1 0 2)。実行管理部 1 2 5 は、1 つの実行ノードに対する実行環境 1 1 2 の設定を終えると、変数 N を 1 加算し (S 1 0 4)、2 番目の実行ノード 1 1 0 - 2 に対して同様の処理を繰り返す。これを、記憶部 1 2 9 に予め設定されたシステム起動時に起動すべき既定数の全ての実行ノードに対する処理が終るまで (S 1 0 3 で Y E S)、繰り返す。システム起動時に起動すべき実行ノードの個数を定める規定数としては、3 以上の値が使用される。ステップ S 1 0 2 の処理例を以下に示す。

20

【0055】

(a) 実行ノード 1 1 0 - i (i = 1 ~ n) が実マシンの場合

実行管理部 1 2 5 は、実行ノード 1 1 0 - i に対して遠隔操作などで電源オンの命令を送信する。この命令によって起動した実行ノード 1 1 0 - i は、起動途中で、自身の実行環境を実行管理部 1 2 5 に問い合わせる。実行管理部 1 2 5 は、記憶部 1 2 9 から、その実行ノード 1 1 0 - i の実行環境設定データを読み出し、応答する。実行ノード 1 1 0 - i は、応答された実行環境設定データを用いて、自ノード 1 1 0 - i の実行環境 1 1 2 を設定する。具体的には、例えば実行ノード 1 1 0 - i が一般的な P C の場合、実行環境の設定を B I O S に保存して再起動し、再起動後、プログラム 1 1 1 - i の処理を開始させる。

30

【0056】

(b) 実行ノード 1 1 0 - i (i = 1 ~ n) が仮想マシンの場合

仮想マシンの実行環境は、その仮想マシンを実現している実マシン上にあるファイルで指定されるため、実行管理部 1 2 5 は、実行ノード (仮想マシン) 1 1 0 - i を実現している実マシン上の該当するファイルを、記憶部 1 2 9 から読み出した当該実行ノード用の実行環境設定データで更新し、その仮想マシンのプロセスを起動する。起動後、その実行ノード (仮想マシン) がプログラム 1 1 1 - i の処理を開始する。

40

【0057】

他の方法として、同じ実行環境 1 1 2 を持つ実行ノードを予め多数用意しておき、実行管理部 1 2 5 がどの実行ノードを活性化させるかを制御することにより、同じ実行環境 1 1 2 の複数の実行ノード 1 1 0 - 1 ~ 1 1 0 - n を起動するようにしても良い。この場合、実行ノードが実マシンのときは、遠隔操作による電源オンの命令の送信により起動され、

50

仮想マシンのときはそのプロセスの起動命令の送信により行われる。

【0058】

各実行ノード110-1~110-nの実行環境の設定後、システムは、各実行ノード110-1~110-nでプログラム111-1~111-nを並列実行する通常運用状態となる。

【0059】

通常運用状態では、実行管理部125および障害検出部124においてシステムの状態を取得し、取得したシステムの状態に応じた処理を行う。実行管理部125が取得するシステムの状態は、システム全体の終了命令が発行されたかどうか、プログラム111-1~111-nで処理すべき新たな処理要求が発生したかどうか、プログラム111-1~111-nで処理要求に対する処理が終了したかどうかなどの情報である。また、障害検出部124が取得するシステムの状態は、各実行ノード110-1~110-nにおける情報提供部116が提供する、実行環境112の状態を示す環境情報である。取得された環境情報は記憶部129に一旦記憶され、その後読み出されて異常ノードの検出に用いられる。この障害関連処理の詳細については後述する。

【0060】

システム全体の終了命令が発行された場合、終了処理が行われ、すべての実行ノードが停止する。システム全体の終了命令は、システム運用者からの入力、タイマーなどのイベントをトリガーとして発行される。システム全体の終了命令は、管理ノード120の実行管理部125で検出され、各実行ノード110-1~110-nに通知される。具体的な制御命令は、記憶部129に記憶されている各実行ノード110-1~110-nの実行環境設定データに応じて、実行管理部125から発行される。これにより、UPS装置などの電源装置を介して電源断の処理を行う、主記憶や仮想記憶の状態を保持したまま停止するSTR(Suspend to RAM)を行う、などの方法で各実行ノード110-1~110-nが停止する。

【0061】

新たな処理要求が発生した場合、管理ノード120の実行管理部125は、その処理要求を複数の実行ノード110-1~110-n上のプログラム111-1~111-nに処理させるための処理分配処理を実行する。また、プログラム111-1~111-nで処理要求に対する処理が終了した場合、実行管理部125は、システムとしての処理結果を導出する処理を行う。この処理は、例えば、異常ノードとして検出された実行ノード以外の実行ノードのプログラムの処理結果から多数決法によってシステムとしての処理結果を求める手法を使って行うことができる。また、数値計算の結果のような明確な実行結果が無い、あるいは得られない、あるいは得るのが困難なプログラム111-1~111-nの場合、多数決法の適用は困難なので、異常ノードとして検出された実行ノード以外の複数の実行ノードのプログラムで得られた処理結果から、任意の1つをシステムとしての処理結果として選択する手法などを使って、システムとしての処理結果を導出する。勿論、これらの手法に限定されず、任意の手法を用いることが可能である。

【0062】

次に、本実施の形態にかかる耐障害システムにおける障害関連処理の一例について、その流れを示す図4のフローチャートを参照して説明する。

【0063】

複数の実行ノード110-1~110-nにおいてプログラム111-1~111-nが並列実行されているとき、障害検出部124の情報取得手段121は、定期的に、各実行ノード110-1~110-nにおける実行環境112のほぼ同一時点の環境情報を通信経路130を通じて各実行ノード110-1~110-nの情報提供部116から取得し、それを記憶部129の環境情報記憶域に記憶する(ステップS201)。

【0064】

以下、管理ノード120が各実行ノード110-1~110-nから取得する環境情報について詳細に説明する。

【0065】

取得する環境情報の例を、以下に示す。

- (a) プロセッサの使用状態
- (b) 稼動中のプロセスの実行状態
- (c) メモリの使用状態
- (d) キャッシュの使用状態
- (e) 仮想記憶の使用状態
- (f) スワップファイルの使用状態
- (g) ディスクの使用状態
- (h) 入出力インタフェースの状態
- (i) ネットワークの状態
- (j) 各デバイスの温度

10

【0066】

上記のbはソフトウェアにかかる環境情報の例であり、それ以外のa、c～jはハードウェアにかかる環境情報の例である。本発明においては、前記a～jの何れか1種類だけを取得するようにしても良いし、前記a～jの内の任意の複数種類あるいは全部を取得するようにしても良い。前記bだけを取得する構成では、ソフトウェアにかかる環境情報だけを取得する構成になり、前記a、c～jの少なくとも1つを取得する構成では、ハードウェアにかかる環境情報だけを取得する構成になる。勿論、各実行ノードの電源供給の有無や消費電力など、前記a～jに例示した以外の環境情報を取得するようにしても良い。

20

【0067】

前記a～jの環境情報を取得する方法としては、次の2つの方法がある。

【0068】

◇環境情報の取得方法1

管理ノード120の情報取得手段121から通信経路130経由で各実行ノード110-1～110-nにログインし、コマンドラインより様々な調査コマンドを発行してその結果を解析したり、あるいは／proc以下の仮想ファイルシステムを調べる方法である。この場合、実行ノード110-1～110-nの情報提供部116は、管理ノード120からの調査コマンドを受信して実行し結果を返すコマンド実行部として実現される。

30

【0069】

◇環境情報の取得方法2

実行ノード110-1～110-nの情報提供部116を、前記a～jの環境情報を取得するハードウェアデバイスであるサービスプロセッサで実現し、管理ノード120の情報取得部121が各実行ノード110-1～110-nのサービスプロセッサから環境情報を取得する。

【0070】

以下、前記a～jの環境情報の詳細とその取得方法の詳細について説明する。なお、取得方法としては前記取得方法1を使用し、LinuxOS(Redhat7.3)が各ノードに実装されている場合を想定する。

40

【0071】

- (a) プロセッサの使用状態
- (ア) ロードアベレージ(プロセッサの使用率)

これは、uptimeコマンド、あるいはtopコマンドを実行して結果を解析することで取得できる。

- (イ) プロセッサ上で動いているプロセス数

これは、psコマンド、あるいはtopコマンドを実行して結果を解析することで取得できる。

【0072】

- (b) 稼動中のプロセスの実行状態

- (ア) プロセスの実行状態(実行状態(running)、割込み可能な休眠状態(sleeping))

50

ee p i n g i n a n i n t e r r u p t i b l e w a i t)、割込み不可な待機状態 (w a i t i n g i n u n i n t e r r u p t i b l e d e s k s l e e p)、親プロセスが死んでいるゾンビ状態 (z o m b i e)、トレースされている状態 (t r a c e d)、シグナルにより停止している状態 (s t o p p e d)、ページング中 (p a g i n g))

(イ) プロセスが使用しているメモリ領域

(ウ) プロセスのロードアベレージ

(エ) プロセスの実行時間 (u s e r m o d e / k e r n e l m o d e)

(オ) プロセスの実行優先度 (p r i o r i t y)

(カ) プロセスのメモリ消費量

10

(キ) プロセスに割り当てられた仮想メモリのサイズ

(ク) プロセスのページフォールト数 (m a j o r f a u l t / m i n o r f a u l t)

(ケ) プロセスからのディスクの読み込み、書き込み回数／サイズ

(コ) シグナルのビットマップ

これらは、プロセスIDがXXXXとすると、／p r o c / x x x x / s t a t、／p r o c / x x x x / s t a t u s、／p r o c / x x x x / s t a t m以下の仮想ファイルを参照することで取得できる。

【0073】

(c) メモリの使用状態

20

(ア) メモリのサイズ

(イ) メモリの使用量

これらは、f r e eコマンドあるいはt o pコマンドを実行し、結果を解析することで取得できる。また、／p r o c / m e m i n f oを調べることで取得できる。

【0074】

(d) キャッシュの使用状態

(ア) キャッシュのサイズ

(イ) キャッシュヒット率

これらは、f r e eコマンドを実行して結果を解析することで取得できる。また、／p r o c / m e m i n f oを参照することで取得できる。

30

【0075】

(e) 仮想記憶の使用状態

(ア) 仮想記憶のサイズ

(イ) 仮想記憶の使用量

これらは、f r e eコマンド、あるいはt o pコマンドを実行して結果を解析することで取得できる。また、／p r o c / m e m i n f oを参照することで取得できる。

【0076】

(f) スワップファイルの使用状態

(ア) スワップのサイズ

これは、f r e eコマンド、あるいはt o pコマンドを実行して結果を解析することで取得できる。また、／p r o c / m e m i n f oを参照することで取得できる。

40

(イ) スワップされたページ数

これらは、プロセス毎のページフォールト数の合計値として求めることができる。

【0077】

(g) ディスクの使用状態

(ア) ディスクのサイズ

(イ) ディスクの使用量

これらは、d fコマンドを実行して結果を解析することで取得できる。

【0078】

(h) 入出力インタフェースの状態

50

(ア) 各デバイス (I D E、S C S I、P C I、N I C等) の有無

(イ) 各デバイスの状態 (正常に動作しているか)

これらは、`l s p c i` コマンドを実行して結果を解析することで取得できる。また、`N I C` の場合、`i f c o n f i g` コマンドを実行すると、より詳細な情報が得られる。

【0079】

(i) ネットワークの状態

(ア) 接続の有無

(イ) 送受信パケット数

これらは、`i f c o n f i g` コマンドの実行結果を解析することで取得できる。

【0080】

(j) 各デバイスの温度

(ア) プロセッサの温度

これは、`B I O S` が対応していれば `/ p r o c / c p u i n f o` を参照して取得できる。なお、他にも電源装置やマシン内部の温度も取得するようにしてよい。

【0081】

続いて、障害検出部 124 の比較検証手段 122 の動作を説明する。障害検出部 124 の比較検証手段 122 は、記憶部 129 に記憶された各実行ノード 110-1 ~ 110-n における実行環境 112 のほぼ同一時点の環境情報に基づいて、大多数の他の実行ノードの環境情報と明確に相違する環境情報となっている実行ノードを異常な実行ノードとして検出する (S202) 。

【0082】

例えば、プロセスの異常終了が多い異常な実行ノードは、他の正常な実行ノードに比べて、実行状態がゾンビのプロセスが増える。また、プログラムが無限ループなどに陥って暴走した異常な実行ノードは、他の正常な実行ノードに比べて、プロセッサのロードアベレージが高くなる。プロセスのフォークが無限ループ状態になった異常な実行ノードは、他の正常な実行ノードに比べて、プロセス数が増える。メモリリークのバグが発生した異常な実行ノードは、他の正常な実行ノードに比べて、メモリ使用量、仮想記憶使用量、スワップ数が増える。異常によってログが増えすぎた異常な実行ノードは、他の正常な実行ノードに比べて、ディスク使用量が多くなる。ワーム型ウィルスに感染した異常な実行ノードは、他の正常な実行ノードに比べて、ネットワークの送信パケットが増える。プロセッサが熱暴走した異常な実行ノードは、他の正常な実行ノードに比べて、プロセッサの温度が上がる。比較検証手段 122 は、このような経験則を利用して異常ノードを検出する。具体的には、比較検証手段 122 は、多数決法あるいは外れ値検出法を用いて比較検証することで、他と明確に異なる環境情報を持つ実行ノードを、異常のある実行ノードとして同定する。なお、比較検証の手法については、これら 2 つの手法に限定するものではない。

【0083】

ここで、多数決法とは、多数決論理に基づき、大多数を正常、少数を異常とする手法である。また、外れ値検出法とは、例えば " Detecting cellular fraud using adaptive prototypes " (P. Burge, J. Shawe-Taylor, Proceedings of AI Approaches to Fraud Detection and Risk Management, pp: 9-13, 1997) や " On-line Unsupervised Outlier Detection Using Finite Mixtures with Discounting Learning Algorithms " (K. Yamanishi et. al., Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, pp: 320-324, 2000) に記載されている統計的外れ値検出のように、データセットにおいて、大多数のデータが従う確率分布から外

10

20

30

40

50

れたデータ、すなわち発生しにくいデータを「統計的外れ値」として異常と判定する手法である。ここで、環境情報には上記 a～j のように種類の異なる情報が含まれているため、同じ種類どうしの比較に加えて、総合的な比較も行う。具体的には、例えば、各環境情報に重みを付けておき、外れ値の度合い×重みの総和で、総合的な判定を行う。重みの付け方は、経験に基づき予め手動で設定する、過去の障害履歴を参照して設定する等の任意の方法が可能である。

【0084】

比較検証手段122による検証結果は記憶部129に記憶される。実行管理部125は、記憶部129に記憶された検証結果を参照して異常ノードが検出されたかどうかを判定し、異常ノードが検出されていれば、障害対応処理の一環として、検出された異常な実行ノードを電子メールなどの方法で予め定められたユーザに通知する(S203)。勿論、通知先はユーザだけでなく、例えばこの通知結果をもって何らかの制御を行うシステムでも構わない。また、異常ノードが検出された場合にのみ結果を通知するだけでなく、比較検証の結果を常に通知しても構わない。

10

【0085】

情報取得手段121による各実行ノード110-1～110-nからの環境情報の取得とその取得された環境情報に基づく比較検証手段122により比較検証は、少なくとも各実行ノード110-1～110-nに処理要求が分配されてから結果が出るまでの期間中、定期的に繰り返される。繰り返しの周期は、短く設定するほど障害を早期に検出でき高い耐障害性を確保できるが、周期が短くなると処理量が増大するので、実際は、両者を考慮した周期が設定される。また、各実行ノード110-1～110-nの実行結果が出揃った場合には、従来のNバージョン方式のように実行結果だけを比較検証して異常ノードを検出するようにしても良いし、環境情報と実行結果の双方を比較検証して異常ノードを検出するようにしても良い。

20

【0086】

このように本実施の形態では、プログラムを並列実行中の各実行ノードから取得した環境情報を比較検証することで異常な実行ノードを検出するため、最終的な実行結果が出揃う前、つまり障害が実行結果として発現する前に速やかに異常ノードを検出することが可能となる。また、実行結果を取得しなくても検出できるため、数値計算の結果のような明確な実行結果が無い、あるいは得られない、あるいは得るのが困難なプログラムに対しても容易に適用できる。更に、予め異常値を検出する閾値を固定して異常を検出する手法に比べ、柔軟性が高く、想定していない障害も検出できる。

30

【0087】

【発明の第2の実施の形態】

図5を参照すると、本発明の第2の実施の形態にかかる耐障害システムは、管理ノード520の実行管理部525が図1の通知手段123の代わりに、異常な実行ノードを停止させる停止手段526を備えている点で第1の実施の形態と構成が異なり、その他の構成は第1の実施の形態と同じである。また、本実施の形態にかかる耐障害システムの動作は、障害関連処理を除いて第1の実施の形態と同じである。

【0088】

次に、本実施の形態にかかる耐障害システムにおける障害関連処理の一例について、その流れを示す図6のフローチャートを参照して説明する。

40

【0089】

システム起動後、複数の実行ノード110-1～110-nにおいてプログラム111-1～111-nが並列実行される通常運用状態になると、障害検出部124は、プログラム111-1～111-nを実行している実行ノード110-1～110-nの個数が予め定められた既定数X以上であるかどうかを判定し(S601)、既定数X以上であれば、情報取得手段121により、各実行ノード110-1～110-nにおける実行環境112のほぼ同一時点の環境情報を通信経路130を通じて取得し、それを記憶部129の環境情報記憶域に記憶する(ステップS602)。取得する環境情報は第1の実施の形態

50

におけるものと同じである。なお、規定数Xとしては、多数決論理により異常ノードあるいは正常ノードが判明する最小の値「3」を用いるが、3より大きな値にしても良い。

【0090】

次に、障害検出部124の比較検証手段122は、記憶部129に記憶された各実行ノード110-1~110-nにおける実行環境112の環境情報を読み込み、第1の実施の形態と同様に多数決法あるいは外れ値検出法を用いて比較検証することで、他と明確に異なる環境情報を持つ実行ノードを、異常のある実行ノードとして同定する(S603)。比較検証手段122による検証結果は記憶部129に記憶される。

【0091】

次に、実行管理部525は、記憶部129から検証結果を読み出し、異常と判定された実行ノードがあるかどうかを判定し(S604)、異常と判定された実行ノードがある場合、停止手段526により、その異常な実行ノードを停止させる(S605)。実行ノードを停止させる制御命令は、記憶部129に保持されたその実行ノードの環境設定データに応じて、通信経路130経由で停止手段526から発行される。これにより、異常な実行ノードが実マシンの場合、例えば電源装置を介して電源断の処理が行われ、仮想マシンの場合、仮想マシンを実現しているプロセスが停止する。

【0092】

以上のような環境情報の取得と比較検証、異常ノードの停止処理は、プログラム111-1~111-nを実行している実行ノード110-1~110-nの個数が予め定められた既定数X以上である間、定期的に実行される。そして、異常ノードの停止によって、プログラム111-1~111-nを実行している実行ノード110-1~110-nの個数が予め定められた既定数Xより少なくなると(S601でNO)、図6に示される処理を停止する。

【0093】

このように本実施の形態によれば、第1の実施の形態で得られる効果に加えて、異常と判定された実行ノードを停止するため、無駄なリソースの消費がなく、実行コストを低減できる。また、障害が実行結果として発現する前に異常ノードの検出とその停止が行えるため、最終的な実行結果が出揃った時点で異常ノードの検出とその停止を行う場合に比べて、実行コストの削減効果が高く、さらに異常ノードが動作し続けることによる正常ノードへの障害の波及を防止することができる。

【0094】

【発明の第3の実施の形態】

図7を参照すると、本発明の第3の実施の形態にかかる耐障害システムは、管理ノード520の実行管理部525が図1の通知手段123の代わりに、新たな実行ノードを追加する追加手段929を備えている点で第1の実施の形態と構成が異なり、その他の構成は第1の実施の形態と同じである。また、本実施の形態にかかる耐障害システムの動作は、障害関連処理を除いて第1の実施の形態と同じである。

【0095】

次に、本実施の形態にかかる耐障害システムにおける障害関連処理の一例について、その流れを示す図8のフローチャートを参照して説明する。システム起動後、複数の実行ノード110-1~110-nにおいてプログラム111-1~111-nが並列実行される通常運用状態になると、障害検出部124の情報取得手段121は、定期的に、各実行ノード110-1~110-nにおける実行環境112のほぼ同一時点の環境情報を通信経路130を通じて各実行ノード110-1~110-nの情報提供部116から取得し、それを記憶部129の環境情報記憶域に記憶する(ステップS701)。取得する環境情報は第1の実施の形態におけるものと同じである。

【0096】

次に、障害検出部124の比較検証手段122は、記憶部129に記憶された各実行ノード110-1~110-nにおける実行環境112の環境情報を読み込み、第1の実施の形態と同様に多数決法あるいは外れ値検出法を用いて比較検証することで、他と明確に異

なる環境情報を持つ実行ノードを、異常のある実行ノードとして同定する（S702）。比較検証手段122による検証結果は記憶部129に記憶される。

【0097】

次に、実行管理部525は、記憶部129から検証結果を読み出し、異常と判定された実行ノードがあるかどうかを判定し（S703）、異常と判定された実行ノードがある場合、追加手段929により、新たな実行ノードを起動する（S704）。起動する実行ノードの個数、実際に起動する実行ノードの情報、各実行ノードの実行環境設定データは記憶部129に事前に記憶されており、追加手段929は記憶部129からそれらの情報を参照して新しい実行ノードを起動し、他の実行ノードと異なるバージョンのプログラムの処理を開始させる。新しく起動する実行ノードの数は、例えば直前のステップS703で異常と判定された実行ノードの個数と同じに設定されるが、それ以上の数を設定しておくことも可能である。 10

【0098】

このように本実施の形態によれば、第1の実施の形態で得られる効果に加えて、実行ノードが異常と判定される毎に新たな実行ノードが追加されるため、正常な実行ノードの数が多数決論理で必要な最低値3より少なくなる事態を回避でき、耐障害性が向上する。

【0099】

【発明の第4の実施の形態】

図9を参照すると、本発明の第4の実施の形態にかかる耐障害システムは、管理ノード720の実行管理部725が停止手段526に加えて、実行ノードの安定性を判定する安定性判定手段727と、安定性が確認されたら既定数X（Xは3以上の予め定められた値）を超える余分な実行ノードを停止対象ノードとして選択する選択手段728を備えている点で第2の実施の形態と構成が異なり、その他の構成は第2の実施の形態と同じである。また、本実施の形態にかかる耐障害システムの動作は、障害関連処理を除いて第2の実施の形態と同じである。 20

【0100】

次に、本実施の形態にかかる耐障害システムにおける障害関連処理の一例について、その流れを示す図10のフローチャートを参照して説明する。

【0101】

システム起動後、複数の実行ノード110-1～110-nにおいてプログラム111-1～111-nが並列実行される通常運用状態になると、障害検出部124は、プログラム111-1～111-nを実行している実行ノード110-1～110-nの個数が予め定められた既定数X以上であるかどうかを判定し（S801）、既定数X以上であれば、情報取得手段121により、各実行ノード110-1～110-nにおける実行環境112のほぼ同一時点の環境情報を通信経路130を通じて情報提供部116から取得し、それを記憶部129の環境情報記憶域に記憶する（ステップS802）。取得する環境情報は第1の実施の形態におけるものと同じである。 30

【0102】

次に、障害検出部124の比較検証手段122は、記憶部129に記憶された各実行ノード110-1～110-nにおける実行環境112の環境情報を読み込み、第1の実施の形態と同様に多数決法あるいは外れ値検出法を用いて比較検証することで、他と明確に異なる環境情報を持つ実行ノードを、異常のある実行ノードとして同定する（S803）。比較検証手段122による検証結果は記憶部129に記憶される。 40

【0103】

次に、実行管理部725は、記憶部129から検証結果を読み出し、異常と判定された実行ノードがあるかどうかを判定し（S804）、異常と判定された実行ノードがある場合、停止手段526により、第2の実施の形態と同様に、その異常な実行ノードを停止させる（S805）。

【0104】

他方、異常と判定された実行ノードがない場合、実行管理部725は、安定性判定手段7 50

27により、異常と判定されずに残っている正常な実行ノード110-1～110-nの安定性を判定する(S806)。安定性の判定方法としては、異常ノードが検出されなくなってから既定時間が経過したら安定と判断する方法、比較検証手段122による比較検証において既定回数連続で異常ノードが検出されなければ安定と判断する方法などを使用する。勿論、これらの方法に限定するものではない。安定性の判定結果は記憶部129に記憶される。

【0105】

次に、実行管理部725は記憶部129から安定性の判定結果を読み出し、安定しているとの判定結果が出ていない場合(S807でNO)、制御を障害検出部124に移し、障害検出部124は再び環境情報の取得と比較検証の処理を繰り返す。他方、安定しているとの判定結果が出ている場合(S807でYES)、選択手段728により、既定数Xを超える数だけの実行ノードを停止させる実行ノードとして選択する(S808)。停止させる実行ノードを選択する方法としては、偏りを無くすために完全にランダムで選択する方法、事前に定義された順位がより高いものから順に選択する方法、事前に定義されたローテーションスケジュールに従って選択する方法などを使用する。勿論、これらの方法に限定するものではない。選択手段728で選択された実行ノードの情報は記憶部129に記憶される。

【0106】

次に、停止手段526は、記憶部129から停止対象となる実行ノードの情報を読み出し、それらに対して通信経路130経由で制御命令を発行することにより、選択手段728で選択された全ての実行ノードを停止させる。以後、図10に示した処理は停止する。

【0107】

このように本実施の形態によれば、第2の実施の形態で得られる効果に加えて、すべての実行ノードが安定した状態になった時点で、不要な分だけ実行ノードを停止させるため、実行コストをより低減することができる。

【0108】

【発明の第5の実施の形態】

図11を参照すると、本発明の第5の実施の形態にかかる耐障害システムは、管理ノード920の実行管理部925が停止手段526、安定性判定手段727および選択手段728に加えて、新たな実行ノードを追加する追加手段929を備えている点で図9の第4の実施の形態と構成が異なり、その他の構成は第4の実施の形態と同じである。また、本実施の形態にかかる耐障害システムの動作は、障害関連処理を除いて第4の実施の形態と同じである。

【0109】

次に、本実施の形態にかかる耐障害システムにおける障害関連処理の一例について、その流れを示す図12のフローチャートを参照して説明する。

【0110】

システム起動後、規定数X(Xは3以上の値)の複数の実行ノード110-1～110-nにおいてプログラム111-1～111-nが並列実行されている通常運用状態になると、障害検出部124は、情報取得手段121により、各実行ノード110-1～110-nにおける実行環境112のほぼ同一時点の環境情報を通信経路130を通じて情報提供部116から取得し、それを記憶部129の環境情報記憶域に記憶する(ステップS1001)。取得する環境情報は第1の実施の形態におけるものと同じである。次に、障害検出部124の比較検証手段122は、記憶部129に記憶された各実行ノード110-1～110-nにおける実行環境112の環境情報を読み込み、第1の実施の形態と同様に多数決法あるいは外れ値検出法を用いて比較検証することで、他と明確に異なる環境情報を持つ実行ノードを、異常のある実行ノードとして同定する(S1002)。比較検証手段122による検証結果は記憶部129に記憶される。次に、実行管理部925は、記憶部129から検証結果を読み出し、異常と判定された実行ノードがあるかどうかを判定する(S1003)。異常と判定された実行ノードが無い場合、制御はステップS100

1に戻り、一定期間毎に環境情報の取得と比較検証する処理を再び繰り返す。

【0111】

他方、異常と判定された実行ノードがある場合、実行管理部925は、追加手段929により、新たな実行ノードを起動する(S1004)。起動する実行ノードの個数、実際に起動する実行ノードの情報、各実行ノードの実行環境設定データは記憶部129に事前に記憶されており、追加手段929は記憶部129からそれらの情報を参照して新しい実行ノードを起動し、他の実行ノードと異なるバージョンのプログラムの処理を開始させる。新しく起動する実行ノードの数は、例えば直前のステップS1003で異常と判定された実行ノードの個数と同じに設定されるが、それ以上の数を設定しておくことも可能である。

10

【0112】

その後、制御はステップS1005に移る。ステップS1005～S1013は、図10のステップS801～S809と同じである。これにより、プログラム111-1～111-nを実行する実行ノード110-1～110-nの数が既定数X以上あることを条件に、環境情報の取得と比較検証による異常ノードの同定の処理が一定期間毎に繰り返され、異常ノードが発生した場合にはその実行ノードが停止される。また、実行ノードが安定すると、既定数Xを超える余分な実行ノードが停止される。但し、図10の処理と異なり、余分な実行ノードを停止した後、またはプログラム111-1～111-nを実行する実行ノード110-1～110-nの数が既定数Xより少なくなると、制御はステップS1021に移行する。従って、実行ノードの個数が規定数Xより少ない場合(S1021

20

【0113】

このように本実施の形態によれば、第4の実施の形態で得られる効果に加えて、平常時は実行ノードの数を減らすことで実行コストを低減し、障害を検知したら実行ノードを追加して起動、実行することで耐障害性を向上させることができる。従って、必要に応じて適切な耐障害性を提供することができる。

【0114】

30

【発明の第6の実施の形態】

図13を参照すると、本発明の第6の実施の形態にかかる耐障害システムは、管理ノード1120の実行管理部1125が停止手段526、安定性判定手段727、選択手段728および追加手段729に加えて、外部からの要求メッセージを受けて、実行ノード110-1～110-nに分配し、その応答メッセージを収集し、外部に返信するメッセージ管理手段1127を備えている点で図11の第5の実施の形態と構成が異なり、その他の構成は第5の実施の形態と同じである。また、本実施の形態にかかる耐障害システムの動作は、メッセージ管理手段1127に関連する処理を除いて第5の実施の形態と同じである。

【0115】

40

図14を参照すると、メッセージ管理手段1127は、要求メッセージ受信手段1211、要求メッセージ処理判断手段1212、要求メッセージ生成手段1213、要求メッセージ送信手段1214、要求メッセージ処理判断情報記憶部1215および要求メッセージ転送先情報記憶部1216から構成される処理要求分配系と、応答メッセージ受信手段1221、応答メッセージ処理判断手段1222、応答メッセージ生成手段1223、応答メッセージ送信手段1224、応答メッセージ処理判断情報記憶部1225および応答メッセージ転送先情報記憶部1226から構成される処理結果導出系と、実行管理部1125の停止手段526による実行ノードの停止および追加手段729による実行ノードの追加に応じて、各記憶部1215、1216、1225、1226を更新する情報更新手段1231とを含んで構成され、サービスを要求するクライアントなどといった要求メッ

50

セージ送信元 1202 に接続されると共に、実行ノード 110-1 ~ 110-n に相当する実行ノード群 1201 に接続される。なお、図 14 に示される応答メッセージ送信先 1203 は、要求メッセージ送信元 1202 あるいは実行ノード群 1201 に相当する。

【0116】

次に、メッセージ管理手段 1127 の動作を説明する。

【0117】

要求メッセージ送信元 1202 から本実施の形態の耐障害システムに要求メッセージが送信されてくると、要求メッセージ受信手段 1211 で受信する。要求メッセージ処理判断手段 1212 では、要求メッセージ処理判断情報記憶部 1215 に記憶された情報を参照して、受信した要求メッセージの転送の可否および転送する場合には転送先の実行ノード 110-i (i=1~n) を判断する。転送先が無いなどの理由で転送しないと判断した場合は、要求メッセージを破棄するか、受信できない旨のメッセージを要求メッセージ送信元 1202 に返す。転送すると判断した場合、要求メッセージ処理判断手段 1212 は、転送先の実行ノード 110-i を指定して要求メッセージを要求メッセージ生成手段 1213 に伝達する。転送先の実行ノードが複数ある場合、それらの全てを指定する。

【0118】

要求メッセージ生成手段 1213 は、要求メッセージ転送先情報記憶部 1216 を参照して、要求メッセージの宛先を転送する実行ノード 110-i に書き換えることで、実際に転送する要求メッセージを生成する。ここで、転送先の実行ノードが複数あるのであれば、要求メッセージ生成手段 1213 は、要求メッセージを複製し、それぞれの宛先を転送先の実行ノードに書き換えることで、複数の転送する要求メッセージを生成する。

【0119】

要求メッセージ送信手段 1214 は、要求メッセージ生成手段 1213 で生成された要求メッセージを、通信経路 130 を通じて、実行ノード群 1201 内の該当する実行ノード 110-i に送信する。要求メッセージの転送を受けた実行ノード 110-i は、その要求メッセージをプログラム 111-i に処理させる。

【0120】

各実行ノード 110-i のプログラム 111-i は、要求された処理を行い、応答メッセージを送信する。要求メッセージ送信元 1202 からの要求メッセージをプログラム 111-i で一度処理するだけで、そのプログラム 111-i から最終的な処理結果の応答メッセージが出される場合だけでなく、最終的な処理結果を得る過程で別の実行ノードや外部のシステムにデータベースアクセスなどの処理要求を行う応答メッセージ（別サービス要求メッセージ）が返される場合がある。応答メッセージ受信手段 1221 は、このような実行ノード 110-i からの応答メッセージを受信し、内部のバッファに一時的に蓄積する。

【0121】

応答メッセージ処理判断手段 1222 は、応答転送メッセージ生成に必要な処理結果が応答メッセージ受信手段 1221 のバッファに蓄積されたかどうかを判断し、未だ蓄積されていないときは今回の処理を終了する。応答転送メッセージ生成に必要な処理結果が蓄積されていれば、応答メッセージ処理判断手段 1222 は、応答メッセージ処理判断情報記憶部 1225 を参照して、応答転送メッセージの生成可否および転送先の判断を行い、応答転送メッセージを生成する。例えば、蓄積されている応答メッセージが別サービス要求メッセージであって、別サービス要求メッセージを分配する実行ノードが 1 つも無いなどの理由で転送しないと判断した場合、おおもとの要求メッセージを破棄するか、受信できない旨のメッセージを要求メッセージ送信元 1202 に送信する。また、蓄積されている応答メッセージが別サービス要求メッセージであって分配先が存在するために転送すると判断した場合および蓄積されている応答メッセージが別サービス要求メッセージでなく最終的な応答メッセージの場合、応答メッセージが一つであれば、そのメッセージをそのまま転送するメッセージとする。複数の実行ノード 110-i のプログラム 111-i から複数の応答メッセージがある場合、応答メッセージ処理判断手段 1222 は、応答メッセ

ージ処理判断情報記憶部1225を参照して、以下のような応答メッセージの選択あるいは生成を行い、転送先を指定して応答メッセージ生成部1223に伝達する。転送先は、最終的な応答であれば、要求メッセージ元であり、別サービス要求メッセージであれば、実行ノード等である。

【0122】

(a) 任意の一つの実行ノード110-iの計算機プログラム111-iからの応答メッセージを選択し、これを応答転送メッセージとする。このとき、最も早く到着した応答メッセージを選択するようにしても良い。

(b) 予め設定されたタイムアウト時間（例えば1分）内に受信した複数の応答メッセージから多数決論理により1つの応答メッセージを選択し、これを応答転送メッセージとする。 10

(c) 予め定められた既定個数の応答メッセージが到着次第、多数決により1つの応答メッセージを選択し、これを応答転送メッセージとする。

(d) 予め設定されたタイムアウト時間（例えば1分）内に受信した複数の応答メッセージの平均値を取るなどの統計処理を行い、統計処理で得られた値をもつ応答転送メッセージを生成する。

(e) 異常と検出された実行ノード以外の任意の実行ノードの結果を出力する。

【0123】

応答メッセージの選択あるいは生成の方法は、これらに限定されるものではなく、その他任意の方法を使用することができる。 20

【0124】

応答メッセージ生成手段1223は、応答メッセージ転送先情報記憶部1226を参照して、応答転送メッセージの宛先を応答メッセージ送信先1203に書き換えることで、実際に転送する応答メッセージを生成する。応答メッセージ送信手段1224は、生成された応答メッセージを応答メッセージ送信先1203に送信する。応答メッセージ送信先1203が、実行ノードである場合、その実行ノードのプログラムは要求された処理を行い、その処理結果を応答メッセージで送信する。また、応答メッセージ送信先1203が、要求メッセージ送信元1202である場合、システムとしての処理結果が要求メッセージ送信元1202に返却されたことになる。

【0125】

他方、情報更新手段1231は、実行ノードが起動されたり、停止されたりしてシステムの状態が変化すると、その情報を元に、要求メッセージ処理判断情報記憶部1215、要求メッセージ転送先情報記憶部1216、応答メッセージ処理判断情報記憶部1225および応答メッセージ転送先情報記憶部1226に格納された情報を適宜更新する。 30

【0126】

このように本実施の形態によれば、第5の実施の形態で得られる効果に加えて、実行ノードと外部とのやり取りにNバージョン方式と同様な耐障害性を付与することができる。

【0127】

【発明の第7の実施の形態】

従来の耐障害システムおよびこれまでに説明した本発明の第1～第6の実施の形態にかかる耐障害システムは、全ての実行ノードが同一の実行環境を持ち、その同一の実行環境上で、異なるバージョンのプログラムを実行することで、耐障害性を付与していたが、本実施の形態にかかる耐障害システムは、全ての実行ノードが互いに異なる実行環境を持ち、その異なる実行環境上で、同一のプログラムを実行することで、耐障害性を持たせる。一般にソフトウェアフォールトは、ソフトウェアであるプログラムに存在する特定のバグが特定の状態に陥った際に活性化して、障害として発現するため、複数の実行ノードの実行環境をそれぞれ異ならせることで、プログラムに存在する特定のバグが活性化する特定の状態が全ての実行ノードで同時に起き難くなり、同一のプログラムを使用してソフトウェアレベルでの耐障害性を持たせることができる。このような耐障害手法を、本明細書では環境多様型ソフトウェアフォールトトレランス（略して環境多様SFT）と呼ぶ。 40 50

【0128】

図15を参照すると、本発明の第7の実施の形態にかかる耐障害システムは、複数の実行ノード1310-1～1310-nと、これら複数の実行ノード1310-1～1310-nを管理する少なくとも1つの管理ノード1320と、これら複数の実行ノード1310-1～1310-nおよび管理ノード1320を互いに通信可能に接続する通信経路130とを含んで構成される。実行ノード1310-1～1310-nは、同一のプログラム1311と、そのプログラム1311を実行するそれぞれ異なる実行環境1312-1～1312-nと、情報提供部116とを有する実マシンあるいは仮想マシンである。それぞれの実行環境1312-1～1312-nは、少なくとも1つの演算部1313-1～1313-nと記憶部1314-1～1314-nと入出力部1315-1～1315-nとを備えている。また、管理ノード1320は、図13に示した第6の実施の形態における管理ノード1120と同様の構成を備えている。つまり、本実施の形態は、図13に示した第6の実施の形態において、プログラムを全て同じバージョンにし、実行環境をそれぞれ異ならせたものに相当する。

【0129】

実行環境は、ハードウェア環境、ソフトウェア環境、外部接続機器環境およびプログラム起動環境の4種類に大別され、この4種類のうちの少なくとも1種類が相違すると、異なる実行環境となる。

【0130】

ハードウェア環境とは、計算機の主記憶容量、仮想記憶容量、メモリアクセスタイミング、プロセッサ速度、バス速度、バス幅、プロセッサ数、リードキャッシュメモリサイズ、ライトキャッシュメモリサイズ、キャッシュの有効・無効状態、プロセッサやメモリの種類などのことである。これらハードウェア環境を規定する計算機の主記憶容量などをハードウェア環境パラメータと呼ぶ。ハードウェア環境パラメータの1つでも互いに相違するハードウェア環境は異なるハードウェア環境となる。

【0131】

ソフトウェア環境とは、OS、基本ソフトウェア、各種デバイスドライバ、各種ライブラリのバージョンのことである。基本ソフトウェアとは、インターネットブラウザ、ファイアーウォール、ウィルスチェッカーなどのように、常時実行されるために、計算機プログラムの動作に影響を与えるもののことである。各種デバイスドライバとは、ディスクドライバなどのことである。各種ライブラリとは、動的リンクライブラリなどのことである。また、これらのソフトウェアに対する修正パッチ、セキュリティパッチの適用・未適用の設定もソフトウェア環境の一つである。さらに、OS、基本ソフトウェア、ライブラリの動作を規定するレジストリ、セキュリティポリシー、アクセスポリシーなどのコンフィグレーションファイルの設定状態もソフトウェア環境に含まれる。これらソフトウェア環境を規定するOSのバージョンなどをソフトウェア環境パラメータと呼ぶ。ソフトウェア環境パラメータの1つでも互いに相違するソフトウェア環境は異なるソフトウェア環境となる。

【0132】

外部接続機器環境とは、計算機に接続されている外部記憶装置、表示装置、入力装置、通信装置などの仕様のことである。例えば、シリアルポート、ディスプレイ、ハードディスク、ネットワーク、キーボード、マウス、タブレット、プリンタ、スピーカーなどの仕様である。これら外部接続機器環境を規定する外部記憶装置の仕様を外部接続機器環境パラメータと呼ぶ。外部接続機器環境パラメータの1つでも互いに相違する外部接続機器環境は異なる外部接続機器環境となる。

【0133】

プログラム起動環境とは、計算機プログラムの起動方法の種類であり、具体的には、計算機プログラムのサスペンド・ツー・ラムによる一時停止と再起動、計算機プログラムのサスペンド・ツー・ディスクによる一時停止と再起動、計算機プログラムのOSによる停止と再起動、計算機プログラムとOSのシャットダウン処理と再起動、計算機プログラムと

OSの強制終了と再起動、計算機プログラムとOSの再インストール後の再起動、計算機プログラムとOSのクリアインストール後の再起動といったプログラム起動方法の種類である。これらプログラム起動環境を規定する起動方法の種類はプログラム起動環境のパラメータの1つである。また、実行ノードが活性化する条件もプログラム起動環境のパラメータに含まれる。例えば、ある実行ノードは、システムの運用開始と同時に活性化し、別の実行ノードは、別の実行ノードの停止または負荷増加のイベント通知を受けて活性化するなどの如きものである。さらに、起動時刻、停止タイミング、停止時刻、連続稼動時間といった運用環境も、ここではプログラム起動環境パラメータに含める。プログラム起動環境パラメータの1つでも互いに相違するプログラム起動環境は異なるプログラム起動環境となる。

10

【0134】

本実施の形態においては、管理ノード1320の記憶部129に記憶されている各実行ノード毎の実行環境設定データは互いに異なり、第1の実施の形態と同様に図3に示した手順によってシステム起動時に各実行ノードの実行環境を設定する際、管理ノード1320の実行管理部1325は、記憶部129に記憶された各実行ノード毎にそれぞれ異なる実行環境設定データを使って、各実行ノード1310-1～1310-nに実行環境1312-1～1312-nおよび同一のプログラム1311の設定を行う。これにより、同一のプログラム1311がそれぞれ異なる実行環境1312-1～1312-n上で並列実行される通常運用状態となる。

【0135】

20

ここで、プログラム1311に存在する特定のバグが活性化する特定の状態を全ての実行ノード1310-1～1310-nで同時に起き難くするためには、実行ノード1310-1～1310-nどうしの実行環境1312-1～1312-nの相違度を或る程度以上大きくするのが望ましい。何故なら、他の環境やパラメータは全て同じで、主記憶容量だけが数バイト乃至数十バイト程度相違するだけでは殆ど効果がないからである。そこで、本実施の形態では、複数の実行ノードの任意の2つの実行ノード間の実行環境の相違度は、予め定められた相違度以上に相違するように定められている。以下、この点について説明する。

【0136】

今、2つの実行ノードX、Yの実行環境の相違度を D_{XY} とし、次式に示すように定義する。

30

$$D_{XY} = aD_H + bD_S + cD_G + dD_K \quad \cdots (1)$$

【0137】

ここで、 D_H は実行ノードX、Yのハードウェア環境の相違度、 D_S は実行ノードX、Yのソフトウェア環境の相違度、 D_G は実行ノードX、Yの外部接続機器環境の相違度、 D_K は実行ノードX、Yのプログラム起動環境の相違度をそれぞれ示す。 a 、 b 、 c 、 d は重み付けの変数であり、全て同じ値を使えば各環境の相違度を公平に考慮することができ、他のものに比べて大きな値を用いればその環境の相違度を重要視できる。これらの重み付け変数は、実験や経験、過去の計算機資源の使用履歴の解析結果などに基づいて設定される。

40

【0138】

ハードウェア環境の相違度 D_H は、ハードウェア環境のパラメータ数を r 個とすると、次式で定義される。

$$D_H = h_1 |D_{HX1} - D_{HY1}| + h_2 |D_{HX2} - D_{HY2}| + \cdots + h_r |D_{HXr} - D_{HYr}| \quad \cdots (2)$$

【0139】

ここで、 D_{HXi} ($i = 1 \sim r$)は実行ノードXのハードウェア環境の i 番目のパラメータ、 D_{HYi} ($i = 1 \sim r$)は実行ノードYのハードウェア環境の i 番目のパラメータ、 $|D_{HXi} - D_{HYi}|$ は実行ノードX、Yのハードウェア環境の i 番目どうしのパラメ

50

ータの相違度で、パラメータの種類に応じて定められる。例えば、記憶部 1 3 1 4 - 1 ~ 1 3 1 4 - n の相違度は、2 倍未満の容量差であれば相違度 0（実質的な相違なし）、2 倍以上 4 倍未満の容量差があれば相違度 1、4 倍以上 8 倍未満の容量があれば相違度 2 の如く定められる。また、演算部（プロセッサ）1 3 1 3 - 1 ~ 1 3 1 3 - n の相違度は、種類が同じで且つバージョンが同じであれば相違度 0、種類が同じであるがバージョンが異なれば相違度 1、種類が異なれば相違度 2 の如く定められる。また、キャッシュの有効、無効のパラメータなど、取り得る状態が 2 つしか無いパラメータの場合、両者が同じ状態であれば相違度 0、異なっていれば相違度 1 とする。 h_i ($i = 1 \sim r$) は重み付けの変数であり、全て同じ値を使えば各パラメータの相違度を公平に考慮することができ、他のものに比べて大きな値を用いればそのパラメータの相違度を重要視できる。これらの重み付け変数は、実験や経験、過去の計算機資源の使用履歴の解析結果などに基づいて設定される。

10

【0140】

ソフトウェア環境の相違度 D_s は、ソフトウェア環境のパラメータ数を s 個とすると、次式で定義される。

$$D_s = s_1 |D_{sx1} - D_{sy1}| + s_2 |D_{sx2} - D_{sy2}| + \dots + s_s |D_{sx_s} - D_{sy_s}| \quad \dots (3)$$

【0141】

ここで、 $D_{s_{X_i}}$ ($i = 1 \sim s$) は実行ノード X のソフトウェア環境の i 番目のパラメータ、 $D_{s_{Y_i}}$ ($i = 1 \sim s$) は実行ノード Y のソフトウェア環境の i 番目のパラメータ、 $|D_{s_{X_i}} - D_{s_{Y_i}}|$ は実行ノード X、Y のソフトウェア環境の i 番目どうしのパラメータの相違度で、パラメータの種類に応じて定められる。例えば、OS の相違度は、同じ種類で且つバージョンが同じであれば相違度 0、同じ種類の OS であるがバージョンが異なれば相違度 1、異なる種類の OS であれば相違度 2 の如く定められる。ウィルスチェッカーなど、実装の有無も相違する場合、例えば、共に実装されていてバージョンが同じであれば相違度 0、同じ種類のプログラムであるがバージョンが異なれば相違度 1、異なる種類のプログラムであれば相違度 2、実装の有無が相違すれば相違度 3 の如く定められる。 s_i ($i = 1 \sim s$) は重み付けの変数であり、全て同じ値を使えば各パラメータの相違度を公平に考慮することができ、他のものに比べて大きな値を用いればそのパラメータの相違度を重要視できる。これらの重み付け変数は、実験や経験、過去の計算機資源の使用履歴の解析結果などに基づいて設定される。

20

30

【0142】

外部接続機器環境の相違度 D_g は、外部接続機器環境のパラメータ数を t 個とすると、次式で定義される。

$$D_g = g_1 |D_{gx1} - D_{gy1}| + g_2 |D_{gx2} - D_{gy2}| + \dots + g_t |D_{gx_t} - D_{gy_t}| \quad \dots (4)$$

【0143】

ここで、 $D_{g_{X_i}}$ ($i = 1 \sim t$) は実行ノード X の外部接続機器環境の i 番目のパラメータ、 $D_{g_{Y_i}}$ ($i = 1 \sim t$) は実行ノード Y の外部接続機器環境の i 番目のパラメータ、 $|D_{g_{X_i}} - D_{g_{Y_i}}|$ は実行ノード X、Y の外部接続機器環境の i 番目どうしのパラメータの相違度で、パラメータの種類に応じて定められる。例えば、ハードディスクの相違度は、2 倍未満の容量差であれば相違度 0（実質的な相違なし）、2 倍以上 4 倍未満の容量差があれば相違度 1、4 倍以上 8 倍未満の容量があれば相違度 2 の如く定められる。また、プリンタの相違度は、共に実装されていて同じ種類であれば相違度 0、共に実装されているが種類や仕様が異なれば相違度 1、実装の有無が相違すれば相違度 3 の如く定められる。 g_i ($i = 1 \sim t$) は重み付けの変数であり、全て同じ値を使えば各パラメータの相違度を公平に考慮することができ、他のものに比べて大きな値を用いればそのパラメータの相違度を重要視できる。これらの重み付け変数は、実験や経験、過去の計算機資源の使用履歴の解析結果などに基づいて設定される。

40

【0144】

50

プログラム起動環境の相違度 D_K は、プログラム起動環境のパラメータ数を u 個とすると、次式で定義される。

$$D_K = k_1 |D_{KX1} - D_{KY1}| + k_2 |D_{KX2} - D_{KY2}| + \dots + k_u |D_{KXu} - D_{KYu}| \quad \dots (5)$$

【0145】

ここで、 $D_{KX i}$ ($i = 1 \sim u$) は実行ノード X のプログラム起動環境の i 番目のパラメータ、 $D_{KY i}$ ($i = 1 \sim u$) は実行ノード Y のプログラム起動環境の i 番目のパラメータ、 $|D_{KX i} - D_{KY i}|$ は実行ノード X、Y のプログラム起動環境の i 番目どうしのパラメータの相違度で、パラメータの種類に応じて定められる。例えば、再起動方法の相違度は、種類が同じであれば相違度 0、種類が異なれば相違度 1 の如く定められる。 k_i ($i = 1 \sim u$) は重み付けの変数であり、全て同じ値を使えば各パラメータの相違度を公平に考慮することができ、他のものに比べて大きな値を用いればそのパラメータの相違度を重要視できる。これらの重み付け変数は、実験や経験、過去の計算機資源の使用履歴の解析結果などに基づいて設定される。

【0146】

各実行ノード 1310-1 ~ 1310-n の実行環境 1312-1 ~ 1312-n の選定に際しては、式 (1) で計算される任意の 2 つの実行ノード間の実行環境の相違度が、予め定められた閾値の相違度よりも大きくなるように選定される。これにより、他の環境は全て同じで、主記憶容量だけが高々数バイト乃至数十バイト程度相違するような 2 つの実行環境を設定するような事態を防止することができる。また、閾値を高く設定すれば、複数の実行ノードでプログラムに存在するバグが同時に発現する確率をより小さくすることができる。

【0147】

通常運用状態における動作は、図 13 に示した第 6 の実施の形態の耐障害システムとほぼ同じである。但し、環境多様 SFT では、全ての実行ノードが正常な状態においても最初から環境情報に違いがあるため、比較検証ではこの当初の違いを考慮して、異常ノードを検出する必要がある。以下、この点について詳しく説明する。

【0148】

まず、実行ノードの個数を実行ノード 1、2、3 の 3 台とし、たとえば比較検証する値を、各実行ノード 1、2、3 における或るプロセスの消費メモリ [KB] : x_1 、 x_2 、 x_3 、そのプロセスの CPU 使用率 [%] : y_1 、 y_2 、 y_3 、その実行ノードにおける CPU の温度 [度] : z_1 、 z_2 、 z_3 の 3 つの種類の環境情報を示すパラメータを用いるものとする。

【0149】

また、パラメータ x_1 の、集合 $\{x_1, x_2, x_3\}$ に対する外れ値を計算する関数を下記 (6)、パラメータ y_1 の、集合 $\{y_1, y_2, y_3\}$ に対する外れ値を計算する関数を下記 (7)、パラメータ z_1 の、集合 $\{z_1, z_2, z_3\}$ に対する外れ値を計算する関数を下記 (8) に示すものとする。但し、これらの関数はパラメータが環境の影響を受けない場合のものである。

$$f(x_1, \{x_1, x_2, x_3\}) \quad \dots (6)$$

$$g(y_1, \{y_1, y_2, y_3\}) \quad \dots (7)$$

$$h(z_1, \{z_1, z_2, z_3\}) \quad \dots (8)$$

【0150】

また、実行ノード 1 の総合的な外れ度 $H(M1)$ を、前記 (6) ~ (8) の計算式を使って次式で計算するものとする。

$$\begin{aligned} H(M1) = & \alpha f(x_1, \{x_1, x_2, x_3\}) \\ & + \beta g(y_1, \{y_1, y_2, y_3\}) \\ & + \gamma h(z_1, \{z_1, z_2, z_3\}) \quad \dots (9) \end{aligned}$$

ここで、 α 、 β 、 γ は定数である。

【0151】

さて、実際は、上記のパラメータ x 、 y 、 z のうち、プロセス毎の消費メモリ： x は、基本的に環境の差異の影響を受けないパラメータであり、プロセス毎のCPU使用率（ロードアベレージ）： y は、CPUの処理性能に影響を受けるため環境の差異の影響を受けるパラメータである。また、CPU温度： z は、環境の差異（CPUの種類など）の影響を受けるパラメータであるが、該当ノードのBIOSがCPU温度の取得に対応していない場合には取得すらできないものであるため、値としては未定義値となる場合がある。このため、環境の差異の影響を受けないパラメータ x 、環境の差異の影響を受けるパラメータ y 、取得できないことのあるパラメータ z 毎に、実際には以下のように計算していく。 10

【0152】

◇環境の差異の影響を受けないパラメータ

基本的に環境の差異の影響を受けないパラメータ x は、そのまま式（6）を計算する外れ値計算関数に渡して計算し、その計算結果を式（9）に適用する。

【0153】

◇環境の差異の影響を受けるパラメータ

プロセス毎のCPU使用率（ロードアベレージ） y は、CPUの処理性能に影響を受けるため、以下のように処理する。

【0154】

例えば仕事量10のプロセスは、処理性能100のCPUの場合、使用率は $10 \div 100 = 10\%$ となるが、処理性能500のCPUの場合、使用率は $10 \div 500 = 2\%$ となる。このようなパラメータは、この環境の差異の影響を打ち消す関数を用いて、外れ値を計算する。具体的には、実行ノード1の y に関する外れ値の計算式（7）を、環境差異を打ち消す関数を I とすると、次式（7'）のように変形して使用する。 20

$$\beta g(I(y1, e1), \{I(y1, e1), I(y2, e2), I(y3, e3)\}) \dots (7')$$

ここで、 e は y に影響を与える環境をあらわすパラメータである。また、関数 I は、例えば前述のCPU使用率の場合、次式に示すものとなる。

$$I(y, p) = y \times p \div 100 \dots (10) \quad 30$$

ここで、 y はCPU使用率、 p はCPU処理性能である。

【0155】

◇パラメータが取得できない場合

CPU温度などは、値として未定義値となり得るパラメータの場合、環境差異を打ち消す関数 I' として、以下のようなものを使用する。

$$I' = \begin{cases} A & \text{for } z = \text{未定義} \\ F(z, e) & \text{for } z \neq \text{未定義} \end{cases} \dots (11)$$

40

ここで、 A は定数、 F は z が定義された場合の環境差異を打ち消す関数、 e は z に影響を与える環境をあらわすパラメータである。また、未定義の場合に与えられる値 A では、外れ値計算の信頼性が高くない場合、外れ値計算関数に掛かる定数の値を小さくすることで、その影響を少なくする。具体的には、前記式（9）における定数 γ を小さくする、あるいは0にして、 z に関する外れ値計算の影響を小さくする、あるいは無視する。

【0156】

結局、実行ノード1の総合的な外れ度 $H(M1)$ の計算式は以下のようになる。

$$\begin{aligned}
 H(M1) = & \alpha f(x1, \{x1, x2, x3\}) \\
 & + \beta g(I(y1, e1), \{I(y1, e1), I(y2, e2), I(y3, e3)\}) \\
 & + \gamma h(I'(z1, e1), \{I'(z1, e1), I'(z2, e2), I'(z3, e3)\}) \\
 & \dots (12)
 \end{aligned}$$

【0157】

以上のように本実施の形態によれば、第6の実施の形態で得られる効果に加えて、使用するプログラム1311は1種類のため、プログラムの開発コストを軽減できる。また、実行ノード1310-1～1310-nに仮想マシンを使用すれば、主記憶メモリのサイズなど各種のパラメータ設定の変更が容易であり、多様な実行環境を容易に実現でき、また、常時多様な実行環境を実現する計算機および機器を用意する必要がないので、実行コストの削減が可能である。

【0158】

本実施の形態は、図13に示した第6の実施の形態において、プログラムを全て同じバージョンにし、実行環境をそれぞれ異ならせたものであるが、他の実施の形態として、図1に示した第1の実施の形態、図5に示した第2の実施の形態、図7に示した第3の実施の形態、図9に示した第4の実施の形態において、図11に示した第5の実施の形態において、プログラムを全て同じバージョンにし、実行環境をそれぞれ異ならせることも考えられる。

【0159】

【発明の第8の実施の形態】

これまで説明した実施の形態では、各実行ノードの環境情報の取得と比較検証による異常ノードの同定、異常ノード発生時の障害対応処理といった障害関連処理を、管理ノードで集中的に行う構成を採用したが、障害関連処理を複数の実行ノードで分散的に行う構成にすることも可能である。また、障害関連処理だけでなく、処理要求の分配処理や並列実行されるプログラムの処理結果からシステムとしての処理結果を導出する処理など、他の処理も複数の実行ノードで分散的に行う構成にすることも可能である。一例として、図11に示した第5の実施の形態にかかる耐障害システムを分散管理方式に変更した耐障害システムの構成例を図16に示す。勿論、第5の実施の形態以外の他の実施の形態についても同様に分散管理方式にすることが可能である。

【0160】

図16を参照すると、本発明の第8の実施の形態にかかる耐障害システムは、複数の実行ノード1410-1～1410-nと、これら実行ノード1410-1～1410-nを接続する通信経路130とを含んで構成される。

【0161】

実行ノード1410-1～1410-nは、図11に示される第5の実施の形態における実行ノード110-1～110-nの構成において、実行環境112に代えて、実行環境112が有する演算部113、記憶部114および入出力部115を有し且つ管理部1420を有する実行環境1412を備える点で異なる。

【0162】

管理部1420は、実行ノード1410-1～1410-nの環境情報を取得する情報取得手段1431および環境情報を比較検証する比較検手段1432を有する障害検出部1430と、他の実行ノードと協調して実行ノード1410-1～1410-nの実行管理を行う協調実行管理部1440とを備えている。また、協調実行管理部1440は、異常として検出された実行ノードや安定状態における余分な実行ノードを停止させる停止手段1441と、実行ノード1410-1～1410-nの安定性を判定する安定性判定手段1442と、安定状態になったときに既定数Xを超える余分な実行ノードを停止対象の実行ノードとして選択する選択手段1443と、新たな実行ノードをシステムに追加する追加手段1444とを含む。

10

20

30

40

50

【0163】

本実施の形態の動作は、本発明の第5の実施の形態とほぼ同じである。ただし第5の実施の形態における管理ノード920は備えず、各実行ノード1410-1~1410-nの管理部1420が通信経路130を介して情報をやり取りし、協調して動作する点が異なる。特に協調実行管理手段1440が異常と同定した実行ノードを停止したり、新たに実行ノードを起動する場合には、それらの動作は排他的に実行される必要がある。なぜなら同時に全ての実行ノードが停止したり、同一の実行ノードに対して複数の起動命令が発行され、コンフリクトを発生する可能性があるからである。

【0164】

協調動作手法としては、以下の2つの手法が考えられる。

10

【0165】

◇手法1 各操作毎に他の全ての実行ノードとネゴシエーションを行う手法である。

具体的には、制御命令を発行するノードをA、制御対象となるノードをB、その他の関連ノードをC（複数存在する）とすると、以下のような手順となる。

【0166】

(1) ノードAは、ノードCに、ノードBを制御する制御許可の申請を通知する。

(2) 通知を受け取ったノードCは、特に問題が無ければ（自分がノードBに何らかの制御をしようとしなければ）、制御命令許可を返信し、問題がある場合は不許可の旨を返信する。

(3) ノードAは一定時間待機して不許可通知がなければ、ノードBの制御を行う。他方、不許可通知があれば、失敗となる。この場合、一定時間（ランダム）待機後、(1)から再試行を行う。一定回数失敗したら、あるいは一定時間経過したら、制御を放棄する。

20

【0167】

◇手法2 関連ノードから暫定的に管理ノードを選択する手法である。具体的には以下のような手順となる。

【0168】

(1) 関連する全てのノードの中から、何らかの方針（例えば、もっとも負荷が少ないノード、等）、あるいはランダムで1つのノードを選び、以降そのノードを管理ノードとする。

(2) 以降の管理制御は、選ばれた管理ノードが一手に行う。

30

(3) 選ばれた管理ノードが、なんらかの理由で停止した場合、残ったノードから同様の方法でノードを選び、新たな管理ノードとする。

【0169】

以上のように本実施の形態によれば、第5の実施の形態で得られる効果に加え、単一の管理ノードを持たないため、単一故障点が存在せず、より高い耐障害性を提供できる。

【0170】

【発明の他の実施の形態】

本発明の第1~第6、第8の実施の形態にかかる耐障害システムは、全ての実行ノードが同一の実行環境を持ち、その同一の実行環境上で、異なるバージョンのプログラムを実行することで、耐障害性を持たせた。また、第7の実施の形態にかかる耐障害システムは、全ての実行ノードがそれぞれ異なる実行環境を持ち、その異なる実行環境上で同一バージョンのプログラムを実行することで、耐障害性を持たせた。本発明の他の実施の形態として、全ての実行ノードがそれぞれ異なる実行環境を持ち、その異なる実行環境上でそれぞれ異なるバージョンのプログラムを実行することで、耐障害性を持たせることもできる。また、一部の実行ノードどうしは実行環境が相違するがプログラムのバージョンは同じであり、他の一部の実行ノードどうしはプログラムのバージョンが相違するが実行環境は同一であるというようにすることで、耐障害性を持たせる実施の形態も考えられる。つまり、本発明の耐障害システムにおける複数の実行ノードは、プログラム及びこのプログラムを実行する実行環境から構成され、前記プログラムのバージョンおよび前記実行環境の少なくとも一方が互いに相違するが全て同等の機能を提供する複数の実行ノードであれば良

50

い。

【0171】

【実施例】

次に本発明の実施例について図面を参照して詳細に説明する。以下の実施例は、本発明の第5の実施の形態の一実施例である。

【0172】

（構成の説明）

図17を参照すると、本実施例は、実行ノード1710-1～1710-n、管理ノード1720およびそれらを相互に通信可能に接続する通信経路130を備える。

【0173】

実行ノード1710-1～1710-nは、インテルアーキテクチャによる演算部1713、記憶部1714、入出力部1715を備える実行環境1712において動作するLinuxマシンであり、アプリケーションプログラム1711-1～1711-nが動作する。また、各実行ノード1710-1～1710-nは、自ノードの環境情報を通信経路1730経由で管理ノード1720に提供する情報提供部1716を有する。

【0174】

管理ノード1720もまたLinuxマシンであり、異常な実行ノードの検出などシステムの障害を検出する障害検出部1721、システム全体の実行状態を管理する実行管理部1724、記憶部1729を備える。

【0175】

障害検出部1721は、各実行ノード1710-1～1710-nにおける実行環境1712の状態を示す環境情報を情報提供部1716から取得して記憶部1729に記憶する情報取得手段1722と、この情報取得手段1722によって取得された各実行ノード1710-1～1710-nにおける実行環境1712の環境情報を記憶部1729から読み出して比較検証し、その検証結果を記憶部1729に記憶する比較検証手段1723とを備える。

【0176】

実行管理部1724は、新たな実行ノードを起動する追加手段1725、指定された実行ノードを停止させる停止手段1726、各実行ノード1710-1～1710-nの安定性を判定する安定性判定手段1727、停止する実行ノードを選択する選択手段1728

【0177】

記憶部1729は、情報取得手段1722が取得する環境情報のリストである取得環境情報リストM1701、この取得環境情報リストM1701に記載された各環境情報毎の取得方法を記述した環境情報取得方法M1702、情報取得手段1722で取得された環境情報M1703、比較検証手段1723による比較検証結果M1704、追加手段1725が新たな実行ノードを追加する際に参照する追加ノード起動情報M1705、安定性判定手段1727の判定結果である安定性判定結果M1706、安定状態に至ったときに選択手段1728が停止対象として選択した実行ノードの情報である選択結果M1707を保持する。なお、記憶部1729には、管理ノード1720がシステム起動時に参照する、起動すべき実行ノードの数と起動対象とする実行ノードの指定情報、各ノードの実行環境のデータおよび並列実行対象となるプログラムも記憶されているが、図17では図示を省略してある。

【0178】

（動作の説明）

次に本実施例による耐障害システムの動作について詳細に説明する。

【0179】

第5の実施の形態における説明個所で説明したように、システムが起動されると、管理ノード1720は、複数の実行ノード1710-1～1710-nが同じ実行環境1712でバージョンの異なるプログラム1711-1～1711-nが並列に実行されるように

10

20

30

40

50

、各実行ノード1710-1~1710-nに実行環境1712およびプログラム1711-1~1711-nの設定を行い、通常運用状態に移行する。本実施例では、3つの実行ノード1710-1~1710-3において、プログラム1711-1~1711-3が並列実行されている通常運用状態になったものとする。

【0180】

この通常運用状態においては、第5の実施の形態における説明個所で説明したように、実行管理部1724および障害検出部1721においてシステムの状態を取得し、取得したシステムの状態に応じた処理を行う。実行管理部1724が取得するシステムの状態は、システム全体の終了命令が発行されたかどうか、プログラム1711-1~1711-3で処理すべき新たな処理要求が発生したかどうか、プログラム1711-1~1711-3で処理要求に対する処理が終了したかどうかなどの情報である。また、障害検出部1721が取得するシステムの状態は、各実行ノード1710-1~1710-3における情報提供部1716が提供する、実行環境1712の状態を示す環境情報である。取得された環境情報は記憶部1729に一旦記憶され、その後読み出されて異常ノードの検出に用いられる。この障害関連処理の実施例については後に詳述する。

【0181】

システム全体の終了命令が発行された場合、終了処理が行われ、すべての実行ノードが停止する。新たな処理要求が発生した場合、管理ノード1720の実行管理部1724は、その処理要求を複数の実行ノード1710-1~1710-3上のプログラム1711-1~1711-3に処理させるための処理分配処理を実行する。また、プログラム1711-1~1711-3で処理要求に対する処理が終了した場合、実行管理部1724は、システムとしての処理結果を導出する処理を行う。この処理は、例えば、異常ノードとして検出された実行ノード以外の実行ノードのプログラムの処理結果から多数決法によってシステムとしての処理結果を求める手法や、異常ノードとして検出された実行ノード以外の複数の実行ノードのプログラムで得られた処理結果から、任意の1つをシステムとしての処理結果として選択する手法などを使って、システムとしての処理結果を導出する。勿論、これらの手法に限定されず、任意の手法を用いることが可能である。

【0182】

次に、本実施の形態にかかる耐障害システムにおける障害関連処理の一例について、その流れを示す図18のフローチャートを参照して説明する。

【0183】

システム起動後、本実施例では前述したように、3つの実行ノード1710-1~1710-3において、プログラム1711-1~1711-3が並列実行されている通常運用状態になる。

【0184】

まず、ステップS1801では、実行管理部1724が、動作している実行ノード1710-1~1710-3の数が規定数X未満かどうかを判定する。本実施例では、規定数Xを3とする。そのため、システム起動直後のステップS1801では、実行ノードは規定数X以上であるため、ステップS1802は実行されず、ステップS1803へと進む。

【0185】

ステップS1803では、障害検出部1721における情報取得手段1722により、各実行ノード1710-1~1710-3の環境情報を取得する。取得すべき環境情報は、取得環境情報リストM1701として記憶部1729に格納されている。本実施例では、取得環境情報リストM1701に、HTTPデーモン（プロセス名httpd）のプロセスの数、ノード全体のメモリ消費量、ノード全体のCPU使用率の3つの環境情報を取得するよう設定されているものとする。また、各環境情報の取得方法は、環境情報取得方法M1702として記憶部1729に格納されている。本実施例では各環境情報は以下のよう

【0186】

○HTTPDのプロセスの数：

10

20

30

40

50

- (1) 対象となる実行ノードに `telnet` でログインする。
- (2) `ps -e | grep httpd | wc` というコマンドを実行する。
- (3) 標準出力として得られた結果を、解析する。

○ノード全体のメモリ消費量：

- (1) 対象となる実行ノードに `telnet` でログインする。
- (2) `free` というコマンドを実行する。
- (3) 標準出力として得られた結果を、解析する。

○ノード全体のCPU使用率：

- (1) 対象となる実行ノードに `telnet` でログインする。
- (2) `uptime` というコマンドを実行する。
- (3) 標準出力として得られた結果を、解析する。

10

【0187】

この結果得られた環境情報を以下の通りとする。

□実行ノード1710-1：

HTTPDのプロセスの数…14 [個]

ノード全体のメモリ消費量…127912 [KB]

ノード全体のCPU使用率…1.12 [%]

□実行ノード1710-2：

HTTPDのプロセスの数…17 [個]

ノード全体のメモリ消費量…183381 [KB]

ノード全体のCPU使用率…62.80 [%]

□実行ノード1710-3：

HTTPDのプロセスの数…15 [個]

ノード全体のメモリ消費量…131104 [KB]

ノード全体のCPU使用率…1.40 [%]

これらの環境情報は記憶部1729に環境情報M1703として格納される。

20

【0188】

障害検出部1721における比較検証手段1723は、記憶部1729に記憶された各実行ノード1710-1～1710-3における実行環境1712の環境情報M1703を読み込み、比較検証することで、他と明確に異なる環境情報を持つ実行ノードを、異常のある実行ノードとして同定する（ステップS1804）。 30

【0189】

比較検証手段1723における比較検証方法として、本実施例ではまず個々のデータの外れ値検出法に、スミルノフ・グラブス検定による外れ値検出を用いる。スミルノフ・グラブス検定による外れ値検出とは、あるデータの集合に対し、対立仮説H1として「データのうち、平均値からもっとも離れたデータは外れ値である」という前提をおき、また、これを否定する帰無仮説H0として「データは同じ母集団のものである」という前提をおき、これらの前提のもと、平均値からもっとも離れたデータXiについて、 $T_i = |X_i - \bar{X}| / \sqrt{U}$ （Xはデータの集合の平均値、Uは不変分散）を求め、Tiに対し有意点tとの差によって帰無仮説を採択あるいは棄却することで、Xiを任意の有意水準で外れ値か否か検定する方法である。 40

【0190】

まずHTTPDのプロセス数については、 $X_1 = 14$ 、 $X_2 = 17$ 、 $X_3 = 15$ であるため、データの集合の平均値X、不変分散Uは、それぞれ下記ようになる。

$$X = \sum_{i=1}^n (X_i) / n = 15.333$$

$$U = \sum_{i=1}^n (X_i - X)^2 / (n-1) = 1.528$$

【0191】

また、各データXiの平均値Xとの差分Siは、

$$S_1 = |X_1 - X| = 1.333$$

$$S_2 = |X_2 - X| = 1.667$$

50

$$S_3 = |X_3 - X| = 0.333$$

であり、もっとも離れたデータは X_2 である。

【0192】

したがって X_2 について検定を行うと、

$$T_2 = |X_2 - X| / \sqrt{U} = 1.091$$

となる。

【0193】

ここで有意水準 5% で検定を行うとすると、 $n = 3$ なので、有意点 t は $t = 1.153$ (統計数値表から求める) となる。したがって、 $T_2 < t$ であるため帰無仮説 H_0 を採択する。すなわち X_2 は外れ値ではない。したがって外れ値は存在しない。

10

【0194】

同様にメモリ消費量については、

$$X_1 = 127912$$

$$X_2 = 183381$$

$$X_3 = 131104$$

であり、

$$X = 147465.700$$

$$U = 31144.510$$

であり、

$$S_1 = 19553.700$$

$$S_2 = 35915.330$$

$$S_3 = 16361.700$$

であるため、 X_2 について検定を行うと

$$T_2 = 1.153$$

となる。したがって、 $T_2 \geq t$ のため、帰無仮説 H_0 を棄却する。すなわち X_2 が外れ値である。

20

【0195】

同様に CPU 使用率については、

$$X_1 = 1.12$$

$$X_2 = 62.80$$

$$X_3 = 1.40$$

であり、

$$X = 21.773$$

$$U = 35.530$$

であり、

$$S_1 = 20.653$$

$$S_2 = 41.027$$

$$S_3 = 20.373$$

であるため、 X_2 について検定を行うと

$$T_2 = 1.155$$

となる。したがって、 $T_2 \geq t$ のため、帰無仮説 H_0 を棄却する。すなわち X_2 が外れ値である。

40

【0196】

次にノード自体の比較検証手法として、以下の式を用いて、総合的な外れ度 H を計算する。

$$\begin{aligned} H_i = & \alpha f(X_i, \{X_1 \dots X_n\}) \\ & + \beta f(Y_i, \{Y_1 \dots Y_n\}) \\ & + \gamma f(Z_i, \{Z_1 \dots Z_n\}) \end{aligned}$$

50

$\alpha \beta \gamma$ は定数であり、本実施例では 1 とする。

関数 $f(X_i, \{X_1, \dots, X_n\})$ は、 X_i が集合 $\{X_1, \dots, X_n\}$ において外れ値であれば 1、外れ値でなければ 0 を返す。

X_i は実行ノード 1710-i の H T T P D のプロセス数、

Y_i は実行ノード 1710-i のメモリの消費量、

Z_i は実行ノード 1710-i の C P U 使用率である。

【0197】

したがって各実行ノードの総合的な外れ度 H は、

$$H_1 = 0 + 0 + 0 = 0$$

$$H_2 = 0 + 1 + 1 = 2$$

$$H_3 = 0 + 0 + 0 = 0$$

となる。

【0198】

総合的な外れ度 H が 2 以上の実行ノードは異常であるとする、 $H_2 = 2$ のため、実行ノード 1710-2 は異常と同定される。この比較検証結果は、比較検証結果 M1704 として記憶部 1729 に格納される。

【0199】

ステップ S1805 では、実行管理部 1724 が比較検証結果 M1704 を読み出し、異常と判定された実行ノードの有無を判定する。異常と判定された実行ノードが無い場合、制御はステップ S1801 に戻り、一定期間毎に環境情報の取得と比較検証する処理を再び繰り返す。

【0200】

前述の例では実行ノード 1710-2 が異常と同定されたため、ステップ S1806 では、追加手段 1725 が新たな実行ノードを起動する。どのように、いくつ実行ノードを起動するかは記憶部 1729 に格納された追加ノード起動情報 M175 に記憶されている。本実施例では 3 つの実行ノード 1710-4、1710-5、1710-6 を起動するとする。そして、ステップ S1807 へ移行する。

【0201】

ステップ S1807 では、実行管理部 1724 が、実行ノード 1710-1 ~ 1710-6 の数が規定数より多いかどうか判定する。実行ノード 1710-1 ~ 1710-6 の数が規定数以下の場合、制御はステップ S1701 に戻る。前述した例では、この時点で実行ノード 1710-1 ~ 1710-6 の 6 個が動作しているため、ステップ S1808 に制御が移る。

【0202】

ステップ S1708 およびステップ S1709 では、再度各実行ノードの環境情報を取得して比較検証することで、異常な実行ノードを検出する。このステップ S1808 およびステップ S1809 の動作は、ステップ S1803 およびステップ S1804 の動作と同一である。その結果、ステップ S1803 で異常ノードと同定された実行ノード、先の例では実行ノード 1710-2 が、再度異常ノードと同定される。

【0203】

ステップ S1810 では、実行管理部 1724 が比較検証結果 M1704 を読み出し、異常と判定された実行ノードの有無を判定する。異常な実行ノードがある場合、停止手段 1726 が、その実行ノードを停止する（ステップ S1811）。今の場合、異常な実行ノードと同定された実行ノード 1710-2 が停止させられる。そしてステップ S1807 に戻り、環境情報の取得と比較検証が繰り返される。

【0204】

この環境情報の取得と比較検証が何度か繰り返された状況で、異常な実行ノードが無い場合、安定性判定手段 1727 が、残った実行ノード 1710-1 ~ 1710-6 の安定性を判定し、その結果を安定性判定結果 M1706 として記憶部 1729 に格納する（ステップ S1812）。本実施例では、ステップ S1810 において異常ノードが検出されな

10

20

30

40

50

くなってから、ステップS1808～S1810、S1812、S1813のループにおいて一定時間、ステップS1810において異常ノードが検出されない場合に、安定したと判定する。

【0205】

ステップS1813では、実行管理部1724が安定性判定結果M1706を読み出し、安定していないと判定されていた場合は、制御はステップS1808に戻る。

【0206】

一方、安定していると判定されていた場合、選択手段1728は規定数X(=3)を超えている数だけ、停止させる実行ノードを選択し、その結果を選択結果M1707として記憶部1729に格納する(ステップS1814)。今の例では、この時点で動作している実行ノードは、実行ノード1710-1、1710-3～1710-6の5個であるので、2個の実行ノードを停止させる実行ノードとして選択する。また、本実施例では、恣意的な選択による偏りをなくすために、停止させる実行ノードをランダムで選択する。

【0207】

次にステップS1815において、停止手段1726は、記憶部1729より選択結果M1707を読み出し、選択された実行ノードを停止させる。そして、制御をステップS1801に戻す

【0208】

なお、ステップS1807～S1811のループが何度か繰り返されて、実行ノード数が規定数X(=3)未満になると、ステップS1801に移行し、このステップS1801の判定結果がYESとなる。このため、追加手段1725は、実行ノード1710-1～1710-nの数が規定数Xになるように、新たな実行ノードを起動する(ステップS1702)。どのように実行ノードを起動するかは記憶部1729に格納された追加ノード起動情報M1705に記憶されている。

【0209】

以上本発明の実施の形態および実施例について説明したが、本発明は以上の実施の形態および実施例に限定されず、その他各種の付加変更が可能である。また、本発明の各実施の形態および実施例における実行ノードおよび管理ノードは、その有する機能をハードウェア的に実現することは勿論、コンピュータとプログラムとで実現することができる。プログラムは、磁気ディスクや半導体メモリ等のコンピュータ可読記録媒体に記録されて提供され、コンピュータの立ち上げ時などにコンピュータに読み取られ、そのコンピュータの動作を制御することにより、そのコンピュータを前述した各実施の形態および実施例における実行ノードおよび管理ノードとして機能させる。

【0210】

【発明の効果】

以上説明したように、本発明によれば、次のような効果が得られる。

【0211】

複数のプログラムを複数の実行ノードで並列実行する耐障害システムにおいて、異常な実行ノードを速やかに検出できる。その理由は、プログラムを並列実行中の複数の実行ノードの環境情報を取得して比較することにより、実行結果が出揃う前に異常なノードの検出が行えるからである。

【0212】

事前に想定していない障害も検出することができる。その理由は、予め異常値を検出する閾値を固定して異常を検出するのではなく、複数の実行ノードの環境情報を比較して異常なノードの検出を行うからである。

【0213】

耐障害システムの実行コストを低減することができる。その理由は、異常ノードを早期に検出してそのノードを停止させるため、無駄なリソースの消費が無くなるからである。また、複数の実行ノードが安定したら、余分な実行ノードを停止させるからである。

【0214】

要求に応じた無駄のない耐障害性を提供できる。その理由は、複数の実行ノードが安定したら余分な実行ノードを停止させて実行コストを下げ、再び異常なノードが検出されたら新たな実行ノードを追加して耐障害性を確保するからである。

【0215】

それぞれ異なる実行環境を持つ複数の実行ノードで同一のプログラムを並列実行させる構成にあっては、プログラムが1種類で済むため開発コストを低減することができる。

【0216】

それぞれ異なる実行環境を持つ複数の実行ノードを仮想マシンで実現する構成にあっては、多種多様な実行環境を実現する実マシンを常時用意しておく必要がなく、多様な実行環境をソフトウェアによって実現できるため、低コストと高い耐障害性の両立が可能である 10

【図面の簡単な説明】

【図1】本発明の第1の実施の形態にかかる耐障害システムのブロック図である。

【図2】実行ノードや管理ノードに使用する仮想マシンの説明図である。

【図3】本発明の第1の実施の形態にかかる耐障害システムにおけるシステム起動時の処理例を示すフローチャートである。

【図4】本発明の第1の実施の形態にかかる耐障害システムにおける障害関連処理のフローチャートである。

【図5】本発明の第2の実施の形態にかかる耐障害システムのブロック図である。

【図6】本発明の第2の実施の形態にかかる耐障害システムにおける障害関連処理のフローチャートである。 20

【図7】本発明の第3の実施の形態にかかる耐障害システムのブロック図である。

【図8】本発明の第3の実施の形態にかかる耐障害システムにおける障害関連処理のフローチャートである。

【図9】本発明の第4の実施の形態にかかる耐障害システムのブロック図である。

【図10】本発明の第4の実施の形態にかかる耐障害システムにおける障害関連処理のフローチャートである。

【図11】本発明の第5の実施の形態にかかる耐障害システムのブロック図である。

【図12】本発明の第5の実施の形態にかかる耐障害システムにおける障害関連処理のフローチャートである。 30

【図13】本発明の第6の実施の形態にかかる耐障害システムのブロック図である。

【図14】本発明の第6の実施の形態にかかる耐障害システムにおけるメッセージ管理手段のブロック図である。

【図15】本発明の第7の実施の形態にかかる耐障害システムのブロック図である。

【図16】本発明の第8の実施の形態にかかる耐障害システムのブロック図である。

【図17】本発明の第5の実施の形態の一実施例にかかる耐障害システムのブロック図である。

【図18】本発明の第5の実施の形態の一実施例にかかる耐障害システムにおける障害関連処理のフローチャートである。

【符号の説明】 40

110-1～110-n…実行ノード

111-1～111-n…プログラム

112…実行環境

113…演算部

114…記憶部

115…入出力部

116…情報提供部

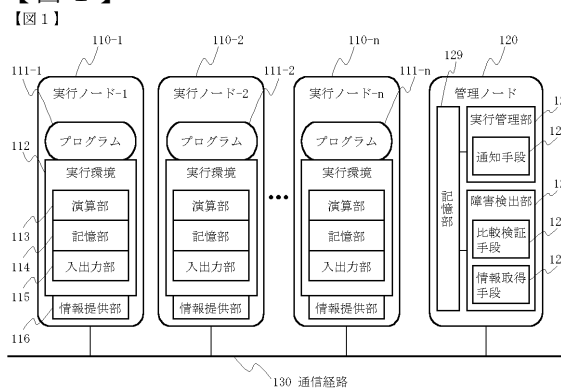
120…管理ノード

121…情報取得手段

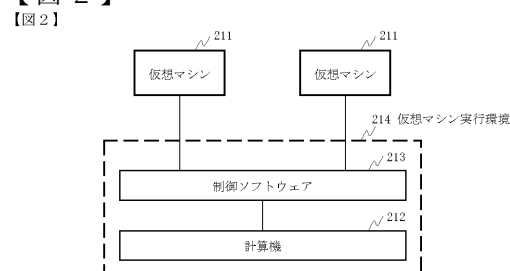
122…比較検証手段 50

- 1 2 3 … 通知手段
- 1 2 4 … 障害検出部
- 1 2 5 … 実行管理部
- 1 2 9 … 記憶部
- 1 3 0 … 通信経路

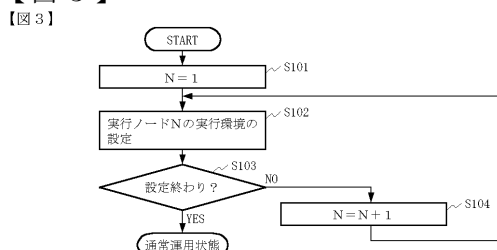
【図 1】



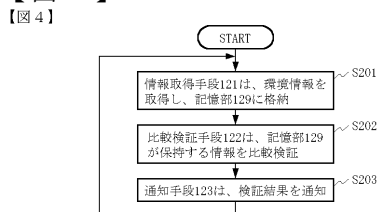
【図 2】



【図 3】

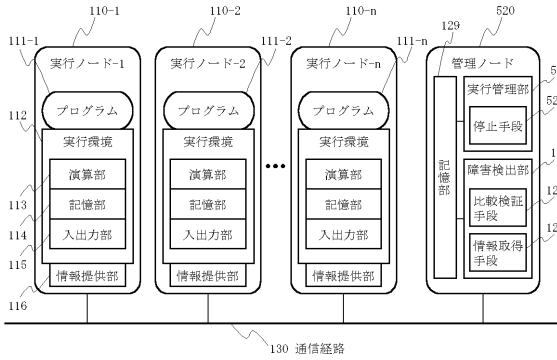


【図 4】



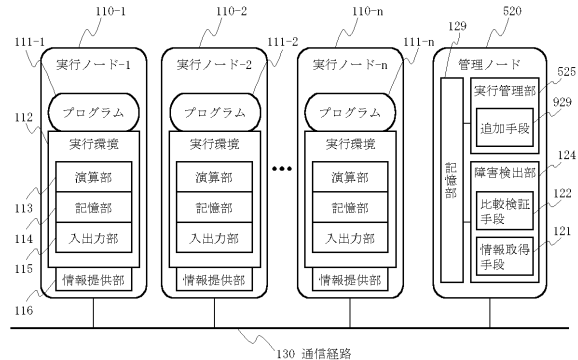
【図 5】

【図 5】



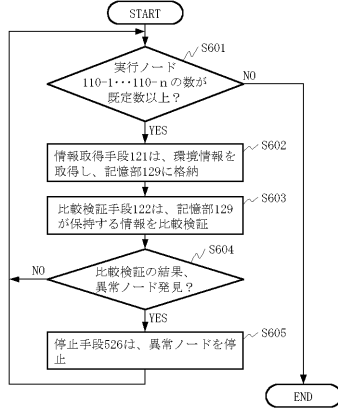
【図 7】

【図 7】



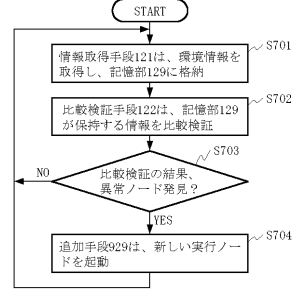
【図 6】

【図 6】



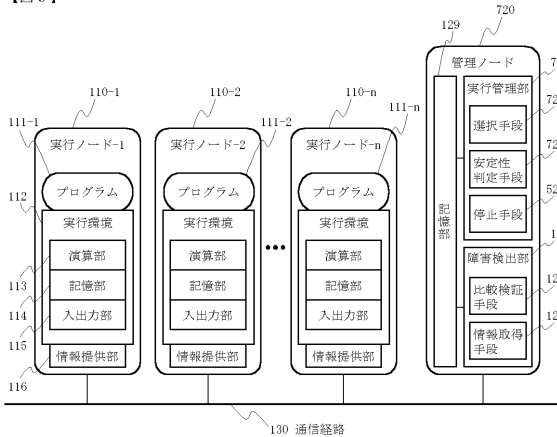
【図 8】

【図 8】



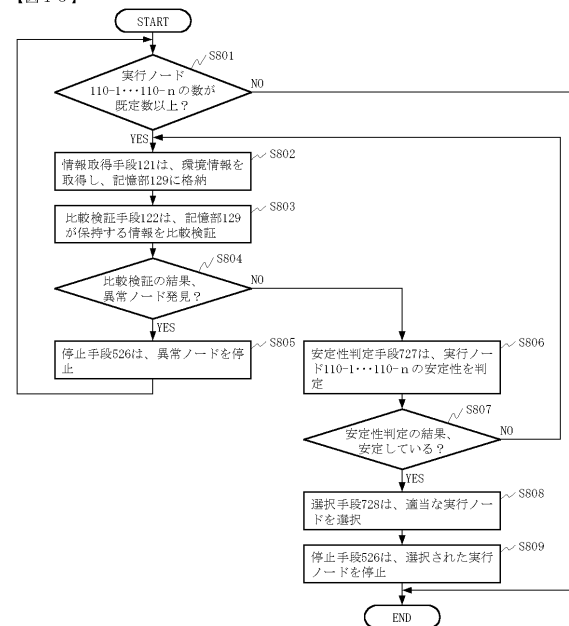
【図 9】

【図 9】



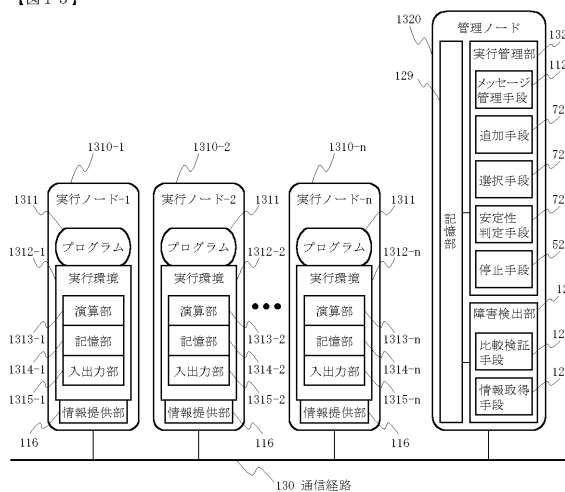
【図 10】

【図 10】



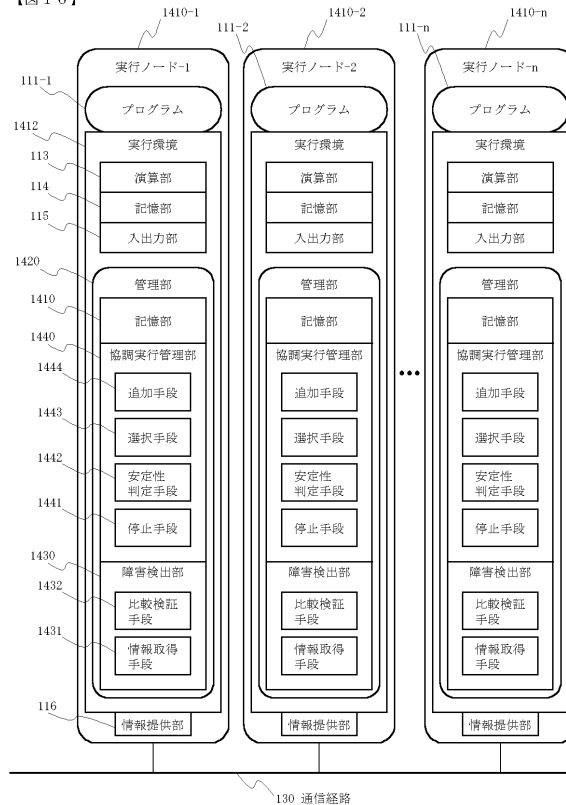
【图 15】

【図 15】



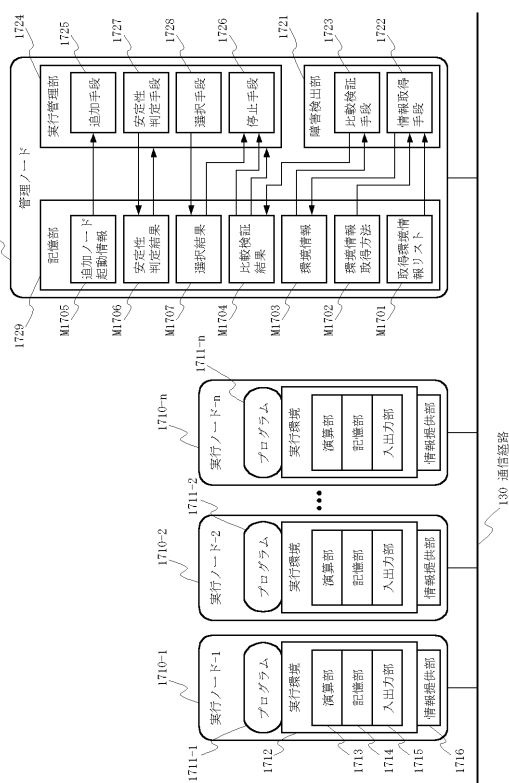
【 図 1 6 】

【图 1-6】



【図 17】

【図 17】



【 図 1 8 】

【図 18】

