(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4405511号
(P4405511)

(45)発行日　平成22年1月27日(2010.1.27)　(24)登録日　平成21年11月13日(2009.11.13)

(51)Int.Cl.　　F I
G06F 11/20 (2006.01)　　G06F 11/20 310F
G06F 15/177 (2006.01)　　G06F 15/177 A

請求項の数 5　(全 17 頁)

(21)出願番号　特願2006-521850 (P2006-521850)
(86)(22)出願日　平成16年6月30日 (2004.6.30)
(65)公表番号　特表2007-500895 (P2007-500895A)
(43)公表日　平成19年1月18日 (2007.1.18)
(86)国際出願番号　PCT/US2004/020906
(87)国際公開番号　WO2005/015346
(87)国際公開日　平成17年2月17日 (2005.2.17)
審査請求日　平成19年6月26日 (2007.6.26)
(31)優先権主張番号　10/604,585
(32)優先日　平成15年7月31日 (2003.7.31)
(33)優先権主張国　米国(US)

早期審査対象出願

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】複数のサービス・ポイントを有する自律コンピューティングにおける動的に構成可能な耐障害性

(57)【特許請求の範囲】
【請求項１】
複数のプロセッサを有するシステムであって、各々が１つのサービス・ポイントを持つ少なくとも１つのクラスタを備えたシステム、を提供するように該システムを構成する方法であって、
前記システムにおける各プロセッサから他のプロセッサまでの距離を計算するステップと、
各合計距離が１つのプロセッサに関連する、複数の合計距離を計算するステップと、
前記複数の合計距離から最小合計距離を決定するステップと、
前記最小合計距離に関連するプロセッサをサービス・ポイントとして割り当てるステップと、
前記システムを複数のクラスタに区分するステップであって、
各プロセッサに関連した合計距離に従ってプロセッサを分類するステップと、
前記分類された各プロセッサを分類順に２つのクラスタの一方に割り当てるステップと、
各クラスタにおけるプロセッサに対する最小合計距離を、当該クラスタにおけるプロセッサに関連した複数の合計距離に従って決定するステップと、
各クラスタに対するサービス・ポイントとして、当該クラスタにおける最小合計距離に関連するプロセッサを割り当てるステップと、
を含むステップと、

前記２つのクラスタの１つを２つのクラスタに分割し、それによって前記システムを３つのクラスタに区分するステップと、
前記３つのクラスタにおけるプロセッサに関連した複数の合計距離に従って、前記３つのクラスタの各々におけるプロセッサに対する最小合計距離を決定するステップと、
前記最小合計距離に対応するクラスタ構成を形成するように前記３つのクラスタにプロセッサを割り当てるステップと、
前記３つのクラスタの各々に対するサービス・ポイントとして、当該クラスタにおける最小合計距離に関連するプロセッサを割り当てるステップと、
を含む、方法。
【請求項２】
前記構成する動作は、プロセッサがシステムに加えられるとき、動的に実行される、請求項１に記載の方法。
【請求項３】
前記構成する動作は、プロセッサがシステムから取り除かれるとき、動的に実行される、請求項１に記載の方法。
【請求項４】
前記システムの区分は、プロセッサがシステムから取り除かれるとき、動的に変更される、請求項３に記載の方法。
【請求項５】
別のプロセッサをバックアップ・サービス・ポイントとして割り当てるステップを更に含む、請求項１に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、自律コンピューティング（autonomic computing）に関し、詳しく云えば、効率的な耐障害性の自己構成（self-configuring）および自己修正（self-healing）オペレーションのためのシステムにおいて、プロセッサをクラスタ化し、サービス・ポイントを割り当てるための方法に関するものである。
【背景技術】
【０００２】
自律コンピューティングは、一般に、自己監視し、自己構成し、耐障害性であり、および自己修正するマルチプロセッサ・コンピュータ・システムを意図したものを指すが、それはかなり原理的および実用的な観点からの見方である。良好な自律コンピューティング・システムを構築する場合の１つの重要な考察点は、自己修正機構を拡張するためにシステム自体にそのシステムの耐障害性を組み込むことである。自己修正機構は、障害時に、システムが直ちにその障害の性質を検出し、その障害の是正を試みることを必要とするであろう。それが障害を是正できなかった場合、システムは、障害を起こしたプロセッサのタスクを１つまたは複数の他のプロセッサに割り振ることによって、続いて起るパフォーマンス低下を最小限にするであろう。一般的なコンピュータ・アーキテクチャでは、障害検出および管理に関するこのタスクは、プロセッサの１つによって、またはマスタ・プロセッサによって行われる。
【０００３】
自律コンピューティング・システムを構成するプロセッサは地理的に広い領域に分散されることがある。更に、プロセッサは多くの種々のタイプのものがあり、多くの種々のタイプのオペレーティング・システムを稼動させ、分散ネットワークによって接続されることがある。種々のプロセッサが地理的にクラスタ状態で配列されることが多い。そのような配列は、１つのマスタ・プロセッサにシステム全体の耐障害性を管理させることを許容するものではない。従って、幾つかのプロセッサに障害管理を行わせることは有益である。本願では、これらのプロセッサをサービス・ポイントと呼ぶことにする。
【０００４】

図１には、サービス・ポイントを利用する代表的なシステムが示される。システム１は複数の相互接続されたプロセッサ２を含み、それらのプロセッサの１つがサービス・ポイント１０となるように割り当てられる。一般に、サービス・ポイントは、そのプロセッサが他のプロセッサまでの最小距離を有するものとなるように選ばれる。本願では、「距離」という用語は、プロセッサ間の必要な通信時間の尺度として使用される。サービス・ポイントは、それ自身の正規のコンピューティング負荷に加えて、次のようないくつかのタスクを有する。

（１）システム内の別の場所における障害プロセッサを検出する、

（２）そのプロセッサのタスクを他のプロセッサに再割り振りすることによって障害プロセッサを置換する、

（３）他のプロセッサによって実行されるタスクを監視する、および

（４）最適のパフォーマンスを保証するためにシステムにおける負荷を平衡させる。

図２は、プロセッサ３がサービス・ポイント１０によって障害を検出されたシステムから取り除かれてしまったこと、および、システムにおける残りのプロセッサが動作を継続していることを示す。

【０００５】

冗長な計算を使用する耐障害性が時々使用されるが、現用の自律計算システムの自己修正および自己構成のフィーチャは、例えば、次のようないくつかの新たな問題点を提起している。

（１）すべてのプロセッサ（遠隔地に設置されたものを含む）が同様のものであり且つ交換可能であるとき、自律システムの自己構成可能および自己調整可能フィーチャは非常に良好に作用する。これは、サービス・ポイントが特別のプロセッサである必要がなく、むしろ、余分な負荷を操作しているプロセッサの同じセットから選択されることを意味する。

（２）通常、並列およびスケーラブル・コンピュータ・アーキテクチャでは、プロセッサ・ポイントの数が一定であり、アクティブ・プロセッサの数分の一として指定することができない。しかし、少な過ぎるサービス・ポイントを持つことは自己修正機構を低速にし過ぎる。多過ぎるサービス・ポイントを持つことはシステムの全体的なパフォーマンスを低下させる。

（３）自律コンピューティング・システムは動的な環境で働くので、システム・パフォーマンスを最適化するためにクラスタ化およびサービス・ポイントの割当てを動的に最適化することが重要である。オンデマンド・コンピューティング環境では、プロセッサの合計数（従って、クラスタの構成およびサービス・ポイントの割当て）がコンピューティング負荷に応答して常に変化している。

【０００６】

自律コンピューティング・システムの自己構成環境では、サービス・ポイントを事前割当てすることは一般に不可能である。従って、その状況に関する要件に従って、いずれの現用のプロセッサもサービス・ポイントになるように動的に割り当てることが可能である。一方、多過ぎるサービス・ポイントを設けることはシステムにおける大きい計算上の負荷を導くことになる。従って、サービス・ポイントの数を稼動プロセッサの数の数分の一に限定して保つことが望ましい。

【０００７】

従って、現在の課題は、分散および動的環境におけるプロセッサのセット、および稼動プロセッサの合計数に対するサービス・プロセッサの最大数の比率を表す数を考慮して、サービス・ポイントおよび各サービス・ポイントがサービスするプロセッサを決定することである。

【０００８】

クラスタ化という概念は多くの他の分野にうまく適用されている。しかし、上述の適用分野では、クラスタの数を経験的に指定することができない。余分のサービス・ポイントが常に制限されるよう、クラスタの数に上限を設定することが必要である。一定の限界を

伴うクラスタ化の問題点は、一般には対処し難いもの、即ち、効率的な最適の解決方法が存在しないもの、と思われている。しかし、次善の効率的な解決方法に対する要求は依然として存在する。更に詳しく云えば、システムの最適なパフォーマンス（自己構成および自己修正を含む）を保証するために、システムにおける種々のプロセッサをクラスタに動的に割り当て、各クラスタ内のサービス・ポイントを割り当てるための効果的な手順に対する要求が存在する。

【発明の開示】

【発明が解決しようとする課題】

【０００９】

本発明の課題は、複数のプロセッサを有するシステムであって、各々が１つのサービス・ポイントを持つ少なくとも１つのクラスタを備えたシステム、を提供するようにそのシステムを構成する方法を提供することにより、上記の要求に対処することである。

【課題を解決するための手段】

【００１０】

本発明によれば、これは、システムにおける各プロセッサから他のプロセッサまでの距離を計算することによって行われる。そこで、複数の合計距離が計算される。その場合、各合計距離が１つのプロセッサに関連している。それらの複数の合計距離から、最小の合計距離が決定される。１つのプロセッサが、サービス・プロセッサとなるように割り当てられる。このプロセッサは、最小の合計距離に関連するプロセッサである。

【００１１】

本発明のもう１つの実施例によれば、本発明の方法は、システムを複数のクラスタに区分することを含む。この区分するプロセスは、各プロセッサに関連する合計距離に従ってプロセッサを分類すること、各プロセッサを２つのクラスタのうちの一方に割り当てること、各クラスタにおけるプロセッサに対する最小の合計距離を、当該クラスタにおけるプロセッサに関連する複数の合計数に従って決定すること、および、各クラスタに対するサービス・プロセッサとして、当該クラスタにおける最小合計距離に関連するプロセッサを割り当てること、を含む。

【００１２】

本発明の更なる実施例によれば、２つのクラスタの１つが２つのクラスタに再分割され、それによってシステムを３つのクラスタに区分する。３つのクラスタにおけるプロセッサに関連する複数の合計距離に従って、３つのクラスタの各々におけるプロセッサに対する最小合計距離が決定される。その最小合計距離に従って、プロセッサが３つのクラスタに割り当てられる。各クラスタにおける最小合計距離に関連するプロセッサがそのクラスタに対するサービス・ポイントとして割り当てられる。

【００１３】

本発明の更なる実施例によれば、プロセッサは異なるタイプのプロセッサであってもよく、それらのタイプに従ってクラスタに割り当てられる。

【００１４】

システムは、プロセッサがそのシステムに加えられるかまたはそのシステムから取り除かれるとき、動的に構成することが可能である。更に、システムの区分化は、プロセッサがシステムから取り除かれるとき、動的に変更することが可能である。

【００１５】

各クラスタにおいて、サービス・ポイントがシステムから取り除かれる場合、そのサービス・ポイントの機能を引き受けるために他のプロセッサをバックアップ・サービス・ポイントとして割り当てることが可能である。

【００１６】

本発明の他の実施例によれば、コンピュータ可読記憶媒体が提供される。その記憶媒体は、複数のプロセッサを有するシステムであって、各々が１つのサービス・ポイントを持つ少なくとも１つのクラスタを備えたシステム、を提供するようにそのシステムを構成する方法を実行するための命令を記憶している。この方法は、システムにおける各プロセッ

サから他のプロセッサまでの距離を計算するステップ、各合計距離が１つのプロセッサに関連する複数の合計距離を計算するステップ、複数の合計距離から最小合計距離を決定するステップと、最小合計距離に関連するプロセッサをサービス・ポイントとして割り当てるステップを含む。

【発明を実施するための最良の形態】

【００１７】

以下の説明では、システムがＮ個のプロセッサｃ１、ｃ２、．．．．．．ｃＮのセットを有するものと仮定する(図３参照)。d(i,i)=0 および d(i,j) が三角不等式を満足するよう、即ち、d(i,j)+d(j,k) が、d(i,k) よりも大きいかまたはそれに等しくなるよう、プロセッサｃｉおよびｃｊの間の距離関数d(i,j) が定義される。プロセッサ間の通信の観点から、これは、j 番目のプロセッサを介した通信が i 番目のプロセッサおよび k 番目のプロセッサの間の直接通信よりも遅くなければならないことを意味する。距離d(i,j) は、通信ネットワークの速度、データ交換およびデータ量の比率、およびプロセッサの地理的位置の関数である。

【００１８】

Ｎ個のプロセッサのうち、サービス・ポイントとして作用するという余分な負荷を引き受けることができるプロセッサの最大分数として、分数値 ｆ が定義される。数値ｐ＝Ｎ＊ｆは、サービス・ポイントとして作用するプロセッサの最大数を表す。従って、システムは、最大ｐ個のクラスタを有することが可能である。

【００１９】

本発明によれば、Ｎ個のプロセッサから成り、ｐ個のクラスタを有するシステムにおいて、クラスタおよびサービス・ポイントを割り当てるための方法が、複雑さを増すいくつかのケースとして下記のように提供される。

【００２０】

Ａ．ケース１：同様のプロセッサの静的システム

プロセッサｃ１、ｃ２、．．．．．．、ｃＮのシステムは、ｐ個の異なるクラスタに分けられるべきものである。この場合、各クラスタは、当該クラスタに属するすべてのプロセッサの、当該クラスタの対応するサービス・ポイントまでの最大距離の和が最小になるよう、サービス・ポイントによって識別される。例えば、図４は、ｐ＝４個のクラスタに分けられるべきＮ＝２０個のプロセッサを有するシステム３０を示す。この例では、分数 ｆ は、１／５であり、実際には、ｆ は一般に１／５０乃至１／１００の範囲の非常に小さい値である。

【００２１】

先ず、ｐ＝１およびｐ＝２の時の単純なケース、即ち、１個および２個のクラスタに関する方法を説明することにする。これらの２つのケースは非自明なケースの基礎を形成する。ｐ＝１に関して、即ち、単一のクラスタに関して、次のようなアルゴリズムが使用される。

（ａ）アルゴリズム（Ｎ,１）

ｉ＝１乃至ｉ＝Ｎに対して下記の動作を行う。

　ｊ＝１乃至Ｎ（ｊ≠ｉ）に対してd_i=SUM{d(i,j)} をセットする。

ｉ＝１乃至Ｎに対して d_i が最小となるようにｉを出力する。

【００２２】

図５は、Ｎ＝１０個のプロセッサを有し、プロセッサｃ１に対して距離d(1,j) が計算されるシステムを示す。距離 d(1,2)、d(1,3)、．．．．．．．、d(1,10) の和がプロセッサｃ１に対する合計距離である。

【００２３】

図６は、アルゴリズム（Ｎ,１）を示すフローチャートである。第１プロセッサに関して開始すると（ステップ３０１におけるｉ＝１）、そのプロセッサと他のプロセッサの各々との間の距離が計算される（ステップ３０２）。これらの距離の和がクラスタにおけるｉ番目のプロセッサに対する合計距離である（ステップ３０３）。合計距離が最小である

プロセッサ（ステップ３０４）がそのクラスタに対するサービス・ポイントになるであろう。

【００２４】

ｐ＝２、即ち、２つのクラスタに関して、下記のアルゴリズム、即ち、アルゴリズム（Ｎ,２）が使用される。そのアルゴリズムでは、アルゴリズム（Ｎ,１）が繰返し使用される。初期ステップのように、Ｎ個のプロセッサのセット全体が単一のクラスタとして扱われ、上記のアルゴリズム（Ｎ,１）が各プロセッサに対する合計距離を計算するために使用される。即ち、Ｎ個のプロセッサの各々が当該プロセッサから他のプロセッサまでの距離の和を表す合計距離を有する。ｉ番目のプロセッサに対するこの距離はd(i,1) として示される。そこで、Ｎ個のプロセッサをこの距離に従って分類することが可能である。

（ｂ）アルゴリズム（Ｎ,２）

１．プロセッサを距離 d(i,1) によって分類する。

２．２つのクラスタＬ１およびＬ２を初期化する。なお、Ｌ１はＮ個のプロセッサすべてを含み、Ｌ２は０個のプロセッサを含む。

２.１　アルゴリズム（Ｎ,１）を使ってＬ１に対するサービスのポイントおよびそれの最小距離を計算する。

２.２　d_0 をそれの距離に初期化する。

３．プロセッサｃ１、ｃ２、．．．．．、ｃＮをそれらの分類順に走査する。

３.１　各プロセッサｃ_ｉ に関して、それをＬ１から取り除き、それをＬ２に入れる。

３.２　Ｌ１およびＬ２対するサービスのポイントおよび距離 d_1iおよび d_2i を計算する。

３.３　距離の和 d_i=d_1i+d_2i をセットする。

４．d_i が最小となるよう ｉ＝１乃至Ｎに対する距離 di を選定し、その距離に対するクラスタＬ１およびＬ２を出力する。

【００２５】

図７は、Ｎ＝１０個のプロセッサを有し、２つのクラスタＬ１およびＬ２に分けられたシステム３０の概略図である。図８は、アルゴリズム（Ｎ,２）を図解したフローチャートを示す。上述のように、先ず、アルゴリズム（Ｎ,１）を使用してシステムの各プロセッサに対する合計距離が計算され（ステップ３２１）、しかる後、この距離に従って、プロセッサが分類される（ステップ３２２）。クラスタのその初期割り当てでは、１０個のプロセッサすべてがクラスタＬ１にあり、クラスタＬ２にはクラスタがない。従って、初期最小合計距離d_0 は、アルゴリズム（Ｎ,１）を使用して前に計算された値と同じである（ステップ３２３）。各プロセッサ（最小距離で始まり、アルゴリズム（Ｎ,１）を使用して計算されたように距離による分類順序で生じる）がクラスタＬ１からクラスタＬ２に順次移動し（ステップ３２４）、各繰返しに対してアルゴリズム（Ｎ,１）を使用して各クラスに対する最小合計距離が計算される（ステップ３２５）。ｉ番目の繰り返しにおいて、最小合計距離が、クラスタＬ１に対するd_1i およびクラスタＬ２に対する d_2i として示される。そこで、結合された最小距離 d_i=d_1i+d_2i が計算される（ステップ３２６）。Ｎ個の繰返しの１つにおいて、d_1の値が最小になるであろう。この値は、対応するクラスタ割振りの際にシステムに対する合計距離として選定される（ステップ３２７）。

【００２６】

ｐ個のクラスタの非自明なケースに関して、上記のアルゴリズム（Ｎ,２）が繰り返し使用される。各ステージにおいて、アルゴリズム（Ｎ,２）を使用して、作成されたクラスタの各々が２つの部分に分けられる。そこで、合計距離関数を最小にするクラスタ割り当てが識別される。

【００２７】

例えば、アルゴリズム（Ｎ,２）が実行されるとき、図７に示されたＮ＝１０個のプロセッサのシステムが２つのクラスタＬ１およびＬ２に分けられる。そこで、次のステージ

において、Ｌ２はそのままにして、クラスタＬ１が２つのクラスタＬ１.１およびＬ１.２に分けられる（図９）。次に、合計距離（d2.1として示される）が、Ｌ１.１、Ｌ２.１、およびＬ２のクラスタ距離の和として計算される。次に、クラスタＬ１をそのままにして、クラスタＬ２が２つのクラスタＬ２.１およびＬ２.２に分けられる（図１０）。再び、合計距離（d2.2として示される）が、Ｌ１、Ｌ２.１、およびＬ２.２のクラスタ距離の和として計算される。d2.1 および d2.2 の小さい方が、それに対応するクラスタ構成と共に選ばれる。従って、d2.1<d2.2である場合、クラスタＬ１.１、Ｌ１.２、およびＬ２が選ばれる。再分割であるＬ１.１およびＬ１.２はそれぞれＬ１およびＬ３として再命名され、クラスタＬ３のこのセットに対する合計距離がd2.1 に等しくセットされる。

【００２８】

同様に、いずれの繰返しステージｑ＋１（但し、ｑ＝２、．．．．、Ｎ－１）においても、クラスタＬ１、．．．．、Ｌｑが存在する。ｑ＋１個のクラスタのｑ個のセットが、他のすべてのクラスタをそのままにして、クラスタＬ１、．．．．、Ｌｑの各々を２つの部分Ｌ_{j.1} およびＬ_{j.2} （但し、ｊ＝１，．．．、ｑ）に一時に１つずつ分けることによって作成される。そこで、これらのクラスタの各々に対して、距離d_{q.1}、．．．、{d_{q.q}} が計算される。これらの距離のうちの最小距離が、対応するクラスタ構成と共に選ばれる。d_{m.j}=min{d_{q.1}、．．．．、{d_{q.q}}であると仮定する。そこで、対応するクラスタ｛Ｌ１、Ｌ２，．．．．、Ｌｑ｝が選ばれる。次に、距離 d_{q+1} が d_{m.j} に等しくセットされ、Ｌｍ＝Ｌｍ.１およびＬ_{q+1}＝Ｌｍ.２にセットされる。このプロセスが、それぞれが自身のサービス・ポイントを有するｑ＋１個のクラスタのセットを生じる。

【００２９】

ｐ個のクラスタのためのアルゴリズムは次のようになる。

（ｃ）アルゴリズム（Ｎ,ｐ）

０．アルゴリズム（Ｎ,２）を使用してプロセッサのセットを２つのクラスタに分ける。

１．ｉ＝２乃至ｐに対して下記の動作を行う。

１.１　Ｊ＝１乃至ｉに対して下記の動作を行う。

１.１.１　アルゴリズム（Ｎ,２）を用いてＬｊをＬｊ.１およびＬｊ.２に分ける。

１.１.２ クラスタの合計距離を合算し、それを d_{i.j} と呼ぶ。

１.２　d_{i+1}=min{d_{i.j}} をセットする（j=1乃至 i）。

最小値が生じたｊの値をｊ＝ｍにすると、

１.３　Ｌｍ＝Ｌｍ.１をセットする。

１.４　L_{i+1}＝Ｌｍ.２をセットする。

２．結果Ｌ１、．．．．、Ｌｐを戻す

【００３０】

図１１は、上記のアルゴリズム（Ｎ,ｐ）に関するフローチャートを示す。先ず、アルゴリズム（Ｎ,２）を使用してシステムが２つのクラスタに分けられる（ステップ３５１）。次に、繰返しが行われ、アルゴリズム（Ｎ,２）を用いてｊ番目のクラスタが２つに分けられ（ステップ３５２）、すべてのクラスタ（２つの再分割されたクラスタを含む）に対する合計距離が計算される（ステップ３５３）。これらの合計距離の最小値が見つけられ（ステップ３５４）、それに対応するクラスタ構成が選ばれる（ステップ３５５）。そこで、それらのクラスタが、図９および図１０に関連して上述したように、再命名される。ｐ個のクラスタのセットが見つけられるまで、クラスタが再分割される。しかる後、クラスタは、各々が１つのサービス・ポイントを有するｐ個のクラスタのセットを有することになる。このプロセスの結果が、Ｎ＝２０個のプロセッサおよびｐ＝４個のクラスタに関して、図１２に概略的に示される（図４と比較されたい）。システム３０は、それぞれのサービス・ポイント３１１、３１２、３１３、３１４を有する４つのクラスタ３１、３２、３３、３４に分けられる。

【００３１】

Ｂ．ケース２：非同一プロセッサの静的システム

図１３は、プロセッサすべてが必ずしも同一のものではないが、それらのオペレーティング・システムおよび／またはテクノロジに従って事前にグループ分けされており、或るタイプのプロセッサに対するサービス・ポイントは同じタイプのものでなければならない、というシステム４０の概略図である。システム４０では、プロセッサが２つのタイプ４１および４２のものである。種々のタイプのプロセッサが、ＴＣＰ／ＩＰのようなプロセッサに無関係のプロトコルを実行するネットワークを介して相互に通信を行うことが可能である。例えば、インテル社のチップ上で稼動するウィンドウズ・ベースのプロセッサのセットおよびＡＩＸ（ＩＢＭコーポレーションの商標）を実行するサーバのセットを使用して、自律コンピューティング環境を設定することも可能である。その場合、インテル社のシステムに対するサービス・ポイントはインテル社のシステムでなければならず、ＡＩＸに関するサービス・ポイントはＡＩＸベースのシステムでなければならない。従って、システム４０は、１つのクラスタ内のすべてのプロセッサが同じタイプのものとなるよう、クラスタに分けられなければならない。

【００３２】

このケースに対する解決方法を、上記のケース１から得ることが可能である。ｍ個の異なるタイプのプロセッサが存在し、これらの異なるタイプの各々に対するサービス・ポイントがそれ自身の種類のみから選ばれたものであることが可能であると仮定する。この場合、先ずｍ＜ｐであるかどうかがチェックされる。但し、ｐはクラスタの最大数である。ｍ≧ｐである（実用的なシステムでは極めてありそうもないことである）場合、クラスタは、単に、プロセッサのタイプに従ってグループ分けされる。ｍ＜ｐに関しては、各々が同じタイプのプロセッサだけを含むｍ個のクラスタＬ１、．．．．、Ｌｍにプロセッサをグループ分けすることによって、クラスタが初期設定される。しかる後、アルゴリズム（Ｎ,ｐ）がこれらのｍ個のクラスタに適用される。これらのプロセスの結果が、Ｎ＝２０、ｍ＝２、およびｐ＝４に関連して図１４に概略的に示される。プロセッサ４１はクラスタ４１１、４１２、４１３にグループ化され、一方、プロセッサ４２はクラスタ４２０にグループ化される。

【００３３】

Ｃ．ケース３：動的システム（プロセッサ編入・離脱システム）

このケースはケース２の拡張であり、プロセッサの数が動的に変化する。プロセッサは、障害のために、またはそれらがオンデマンド環境で動作しているために、システムに編入することまたはシステムから離脱することが可能である。更に、プロセッサの数の変化がクラスタの数または配列の変化に通じることがあり、従って、クラスタも動的に変化する。更に、プロセッサが複数のタイプのものである（従って、複数の異なるグループにある）場合、グループの数が動的に変化することもある。例えば、図１５は、クラスタ５０におけるプロセッサ５０１がシステムから取り除かれることになる状況を示す。そのクラスタは、図１６に示されるように、クラスタ５１として再構成される。プロセッサ５０１を取り除いた結果、クラスタ５１のサービス・ポイントの再割当てが生じることがあり得る。

【００３４】

クラスタ内のプロセッサとサービス・ポイントとの間の合計距離は、システムのパフォーマンスの損失を回避するために動的に最小にされなければならない。１つまたは複数のプロセッサがシステムに加えられるかまたはシステムから取り除かれるたびに、アルゴリズム（Ｎ,ｐ）を使用することによってそのクラスタを再計算することが可能である。この方法は、計算のオーバヘッドによって非常に高価となり、大きなパフォーマンス低下を引き起こすことがある。従って、（ｉ）システムにプロセッサを加えること、および（ｉｉ）システムからプロセッサを取り除くこと、の２つのケースの各々に関してシステムを動的に更新することが必要である。これらの２つのケースに関する手順は後述される。アルゴリズム（Ｎ,ｐ）を使用してクラスタの最適なセットが事前に見つかっているものと仮定する。

【００３５】

（i）新たなプロセッサがシステムに加えられる。この場合、すべての既存のクラスタにおけるすべてのサービス・ポイントに対比してその加えられたプロセッサがテストされる。しかる後、その新たなプロセッサは、合計距離が最小であるクラスタに加えられる。その新たなプロセッサが加えられるクラスタでは、新たなサービス・ポイントを見つけるために、アルゴリズム（N,１）が使用される。この手順は必ずしもシステムを最適化するものではないが、非常に高速であることに留意すべきである。この手順は図１７に概略的に示される。図１７では、システム５５が２つのクラスタ５５１および５５２を含み、新たなプロセッサ５６０を加えることが望ましい。プロセッサ５６０はクラスタ５５２に加えられるであろう。それは、システムの合計距離がこれによって最小に保たれるためである。

【００３６】

（ii）既存のプロセッサがシステムから取り除かれる。この場合、どちらのクラスタからプロセッサが取り除かれるかが先ず決定される。この状況に関しては、次の２つのケースが存在する。

（ａ）取り除かれたプロセッサがサービス・ポイントから最も遠いものではない場合、それはそのクラスタに関する最大距離に影響を与えない。その場合、そのプロセッサは取り除かれ、このクラスタに関してアルゴリズム（N,１）を使用することにより、新たなサービス・ポイントが再計算される。この状況が図１８に示される。そこでは、システム５７がクラスタ５７１および５７２を含み、プロセッサ５７４が取り除かれるべきものである。その結果、クラスタ構成に関する変化は生じない。

（ｂ）取り除かれたプロセッサがサービス・ポイントから最も遠いものである場合、それの取り除きはクラスタにおけるサービス・ポイントからの最大距離に影響を与える。その場合、プロセッサが先ずクラスタから取り除かれる。一般に、システムは、最適なパフォーマンスを維持するために再平衡化されなければならない。プロセッサがクラスタから取り除かれるとき、他のクラスタから当該クラスタに他のプロセッサを組み入れることによって、システムを更に効率的にすることが可能であろう。この状況が図１９に示される。そこでは、プロセッサ５７７がシステムから取り除かれている（図１８と比較されたい）。その結果、システムの合計距離を最小にするように、新たなクラスタ５７８および５７９が形成される。これを達成するために、クラスタの最終的セットを形成するためのアルゴリズム（N,ｐ）の各ステップにおいて、前に形成されたクラスタが、アルゴリズム（N,２）を使って分割される。従って、クラスタを形成するプロセス全体をバイナリ・ツリーで表現することが可能である。最終的なクラスタはすべてバイナリ・ツリーのリーフ・ノードである。

【００３７】

プロセッサがクラスタから取り除かれるとき、そのプロセッサが取り除かれるクラスタの他の同胞（sibling）ノードが考察される。その同胞ノードにおけるプロセッサは、プロセッサを同胞ノードから当該クラスタに移動させることがシステムの全体的な距離を最小にするかどうかに関して調べられる。システム全体の全体的な距離が影響を受けない場合、アクションが取られる必要はない。しかし、プロセッサを移動させることによって全体的な距離が減少する場合、プロセッサが同胞ノードから取り除かれて当該クラスタに組み入れられ、従って全体的な距離が再び最小にされる。同胞ノードがリーフ・ノードまたは既存のクラスタである場合、更なるアクションを取る必要はない。そうでない場合、プロセッサの取り除き後、それのチルドレン・ノードを平衡化するためにアルゴリズム（N,２）が使用される。

【００３８】

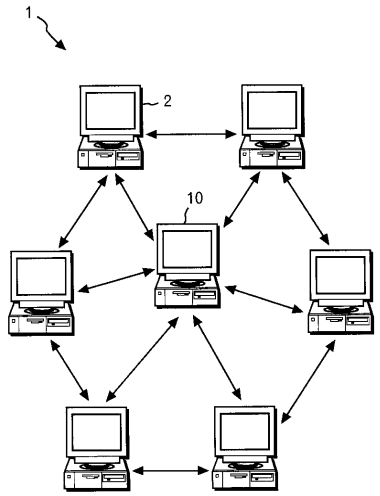
上記の調整は本来ローカル的なことであり、非常に高速で行うことが可能である。しかし、それらはシステム全体を総体的に平行化するものではない。従って、多くの動的な調整後、システムはかなり平衡を失ったものになることがあり、時々アルゴリズム（N,ｐ）を使用してシステムを最適化し直すことが必要であろう。

【００３９】

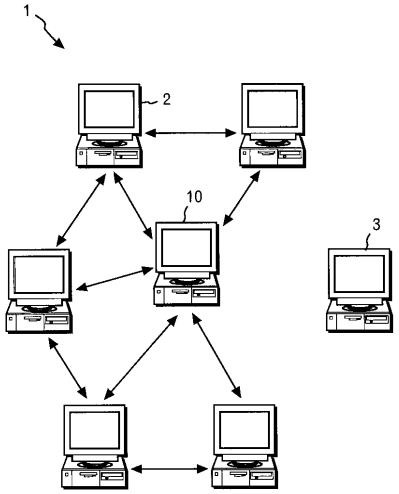
Ｄ．ケース４：サービス・ポイントに対するバックアップ
システムのフェールセーフ動作を保証するために、本発明は、各クラスタにデュアル・サービス・ポイントを提供するための方法も含む。この方法では、バックアップ・サービス・ポイントがアクティブなサービス・ポイント内に記憶された情報のミラー情報を保持し、更新時にサービス・ポイントが障害を生じた場合、アクティブなサービス・ポイントとして機能し始める。それらのサービス・ポイントがシステムの普通のプロセッサとは全く異ならないことは明らかであろう。従って、サービス・ポイントが障害を生じ、システムから取り除き得ることは可能である。この場合のシステム障害を防ぐために、サービス・ポイントのサービス情報が同じクラスタにおける他のプロセッサに記憶される。例えば、図２０に示されたシステム６０は４つのクラスタ６１、６２、６３、および６４を含む。クラスタ６１のサービス・ポイントはプロセッサ６０１を含み、プロセッサ６０２はバックアップ・プロセッサである。アクティブ・サービス・プロセッサ６０１における障害の場合、バックアップ・サービス・ポイント６０２がサービス・ポイントの機能を引き受ける。そこで、アルゴリズム（Ｎ,１）を、必要な場合にクラスタを平衡化し直すために使用することが可能である。
【００４０】
Ｅ．動的な数のサービス・ポイント
更に一般的な問題点は、サービス・ポイントの数ｐに関する最大の許容可能な制限も変更されるという状況である。これは、かなりの数のプロセッサが一時にシステムに加えられるかまたはシステムから取り除かれるときに起こり得る。これは、分数値ｆ（ｐ＝Ｎ＊ｆに従って、サービス・ポイントの最大の許容可能な数ｐを与える）が変更される場合にも起こり得る。次のような考慮すべき２つのケースが存在する。
（ｉ） 最大の許容可能なサービス・ポイントの新たな数ｐ１がｐよりも大きい。この場合、アルゴリズム（Ｎ,ｐ）は現在のクラスタ数と共に使用されるが、そのアルゴリズムは、新しいクラスタ数ｐ１を用いて実行される。これは、ケース１において説明された解決方法と同じであり、第１ステップにおいて、ｐ個のクラスタの各々が１回分割され、一方、他のクラスタはそのままにされる。各ケースにおいて、最も近いサービス・ポイントまでの距離の総和が計算され、その総和を最小にするクラスタ構成が選ばれる。このプロセスは、ｐ＋１、．．．．．、ｐ１に対して繰り返し実行される。
（ｉｉ）最大の許容可能なサービス・ポイントの新たな数ｐ２がｐよりも小さい。この場合、ｐ個のクラスタの各々を生成した命令を再コールすること（ケース１の上記説明参照）が必要である。クラスタは、ｐ２個のクラスタだけが残るまで、逆順に再結合される。
【００４１】
本発明を特定の実施例によって説明したが、当業者にとっては多くの代替、修正、および変更が明らかであることは、上記の説明から鑑みて明白である。従って、本発明は、本発明および「特許請求の範囲」の技術的範囲内にあるそのような代替、修正、および変更をすべて包含することを意図するものである。
【図面の簡単な説明】
【００４２】
【図１】サービス・ポイントを有するシステムにおけるプロセッサの代表的な配列を示す概略図である。
【図２】障害のあるプロセッサがシステムから取り除かれる後の図１のシステムを示す概略図である。
【図３】プロセッサ間の距離が本発明に従って決定される場合の複数のプロセッサを有するシステムの概略図である。
【図４】本発明の実施例に従って、サービス・ポイントを有するクラスタに構成されるべき複数のプロセッサを有するシステムの概略図である。
【図５】本発明に従って、システムにおける１つのプロセッサから他のプロセッサまでの合計距離を計算するための手順を示す概略図である。
【図６】本発明の実施例に従ってアルゴリズム（Ｎ,１）におけるステップを詳細に示す

フローチャートである。
【図７】２つのクラスタに分割されたシステムを示す概略図である。
【図８】本発明の実施例に従ってアルゴリズム（Ｎ,２）におけるステップを詳細に示すフローチャートである。
【図９】本発明の実施例に従って、クラスタが再分割されるシステムを示す概略図である。
【図１０】システムにおけるクラスタの別の再分割を示す概略図である。
【図１１】本発明の実施例に従ってアルゴリズム（Ｎ,ｐ）におけるステップを詳細に示すフローチャートである。
【図１２】図４のシステムにおいてクラスタを構成し、サービス・ポイントを割り当てるためのプロセスの結果を示す概略図である。
【図１３】本発明の更なる実施例に従って、サービス・ポイントを有するクラスタに構成されるべき異なるタイプの複数のプロセッサを有するシステムを示す概略図である。
【図１４】図１３のシステムにおいてクラスタを構成し、サービス・ポイントを割り当てるためのプロセスの結果を示す概略図である。
【図１５】複数のプロセッサを有し、１つのプロセッサがシステムから取り除かれるべきシステムを示す概略図である。
【図１６】プロセッサが取り除かれた後の図１５のシステムを示す概略図である。
【図１７】複数のプロセッサを有し、１つのプロセッサがシステムに加えられるべきシステムを示す概略図である。
【図１８】２つのクラスタを有し、１つのプロセッサがシステムから取り除かれるべきシステムを示す概略図である。
【図１９】クラスタが再構成された場合、プロセッサが取り除かれた後の図１８のシステムを複数のプロセッサを有し、１つのプロセッサがシステムから取り除かれるべきシステムを示す概略図である。
【図２０】クラスタのサービス・ポイントがシステムから取り除かれるべき場合、各々がサービス・ポイントを有するクラスタに構成された複数のプロセッサを有するシステムを示す概略図である。

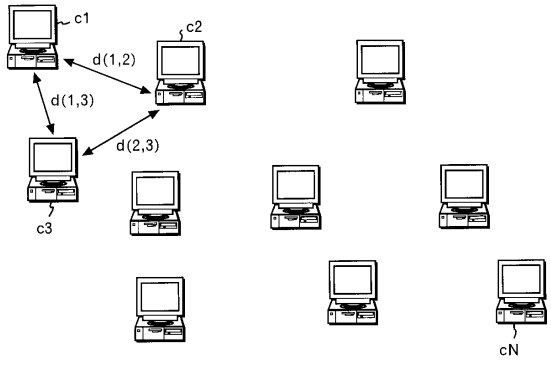
【図１】



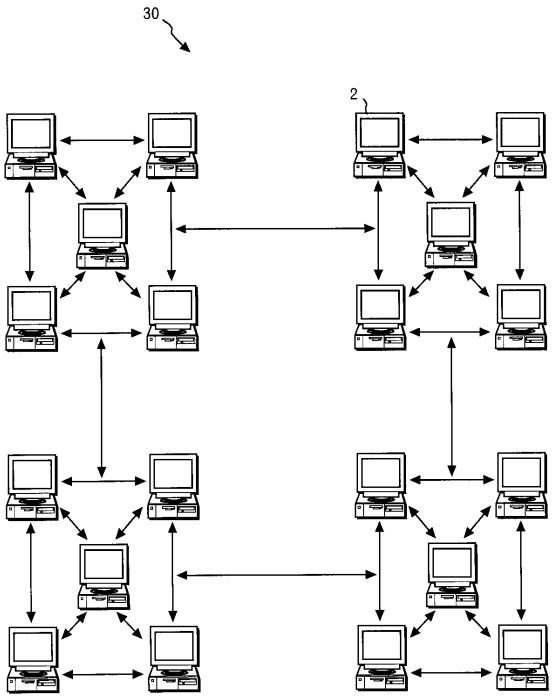
【図２】



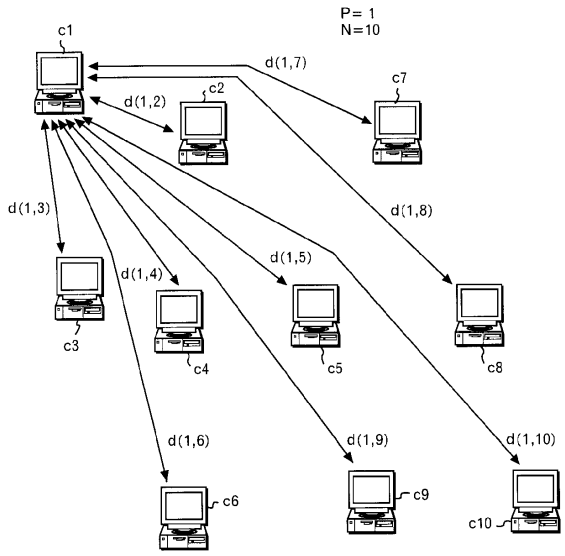
【図３】



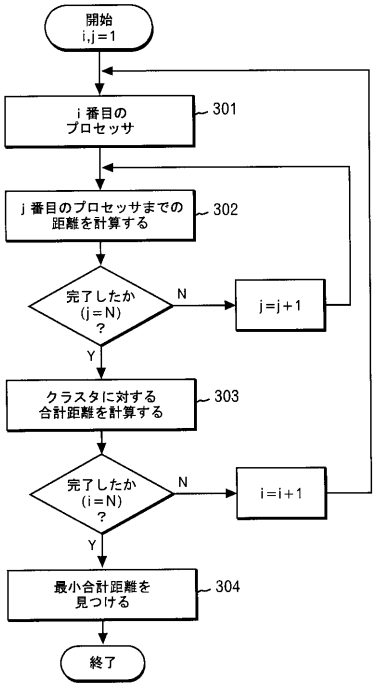
【図４】



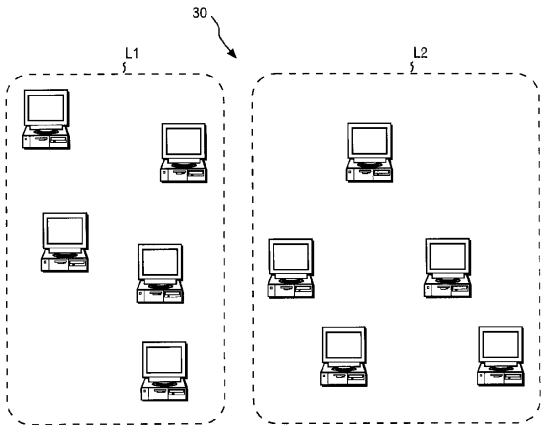
【図５】


P= 1
N=10
c1
c2
c3
c4
c5
c6
c7
c8
c9
c10
d(1,2)
d(1,3)
d(1,4)
d(1,5)
d(1,6)
d(1,7)
d(1,8)
d(1,9)
d(1,10)

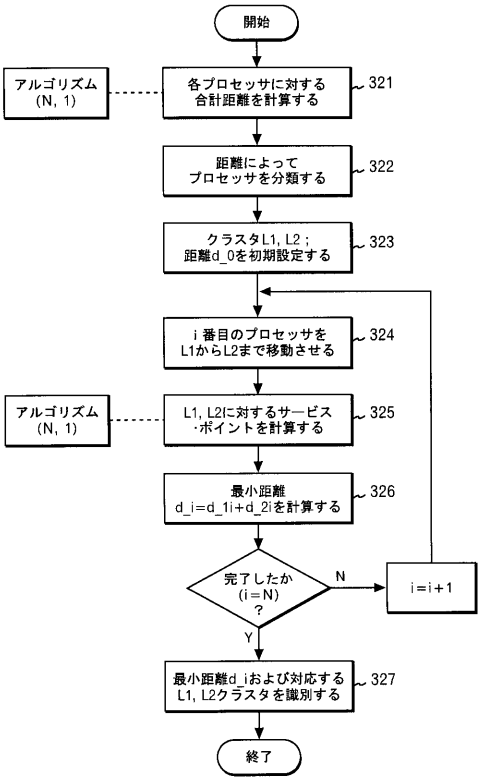
【図６】


開始
i,j=1
i 番目の
プロセッサ
301
j 番目のプロセッサまでの
距離を計算する
302
完了したか
(j=N)
?
N
j=j+1
Y
クラスタに対する
合計距離を計算する
303
完了したか
(i=N)
?
N
i=i+1
Y
最小合計距離を
見つける
304
終了

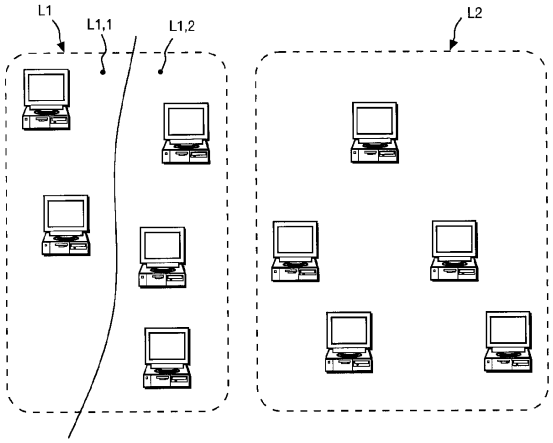
【図７】


30
L1
L2

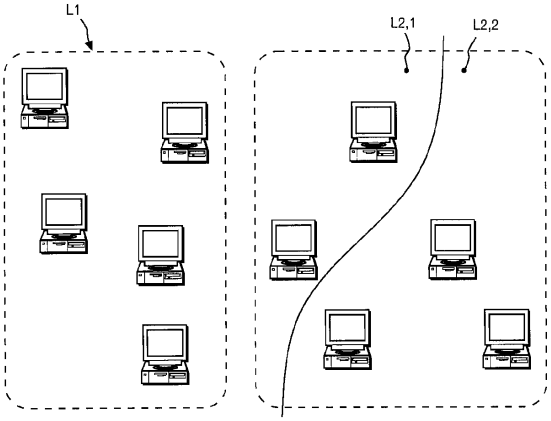
【図８】


開始
アルゴリズム
(N, 1)
各プロセッサに対する
合計距離を計算する
321
距離によって
プロセッサを分類する
322
クラスタL1, L2 ;
距離d_0を初期設定する
323
i 番目のプロセッサを
L1からL2まで移動させる
324
アルゴリズム
(N, 1)
L1, L2に対するサービス
・ポイントを計算する
325
最小距離
d_i=d_1i+d_2iを計算する
326
完了したか
(i=N)
?
N
i=i+1
Y
最小距離d_iおよび対応する
L1, L2クラスタを識別する
327
終了

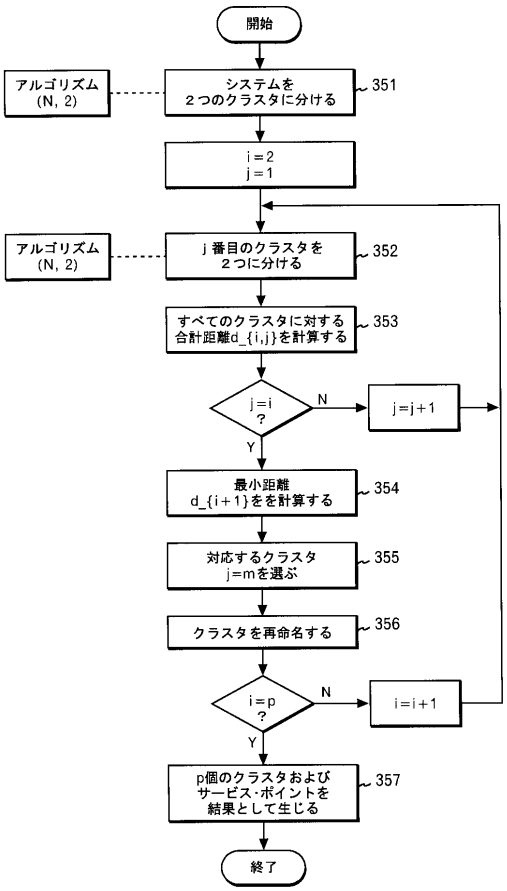
【図９】



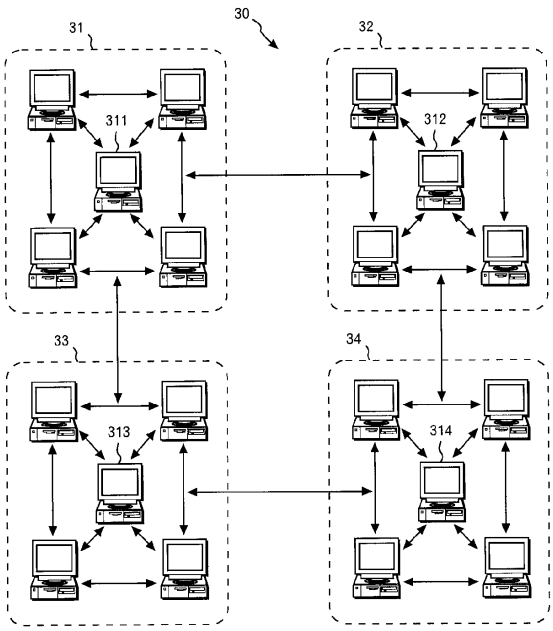
【図１０】



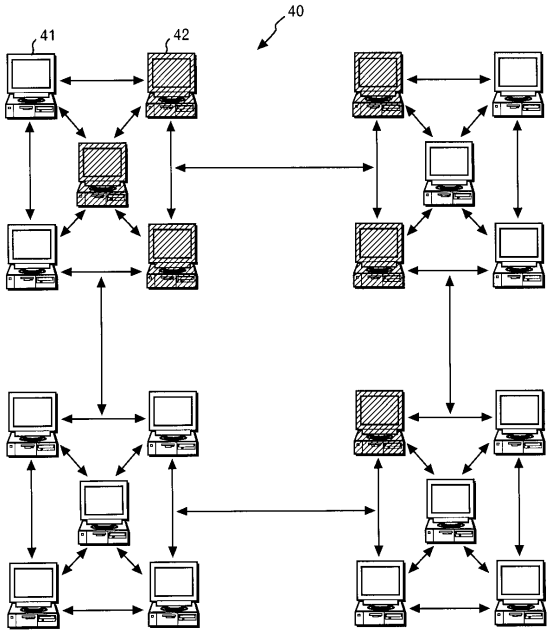
【図１１】



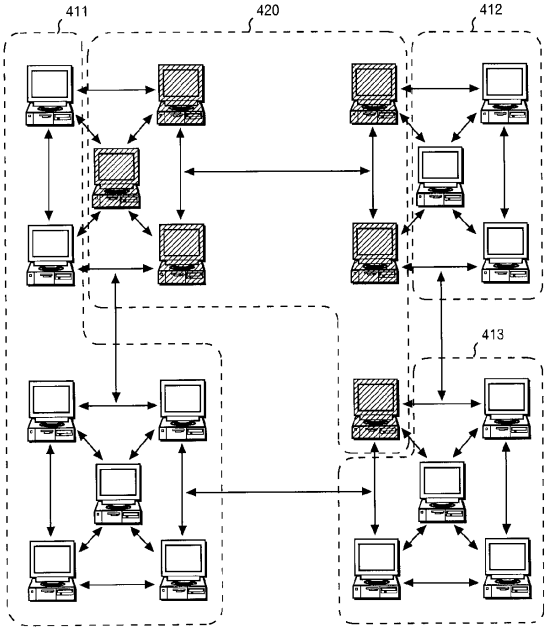
【図１２】



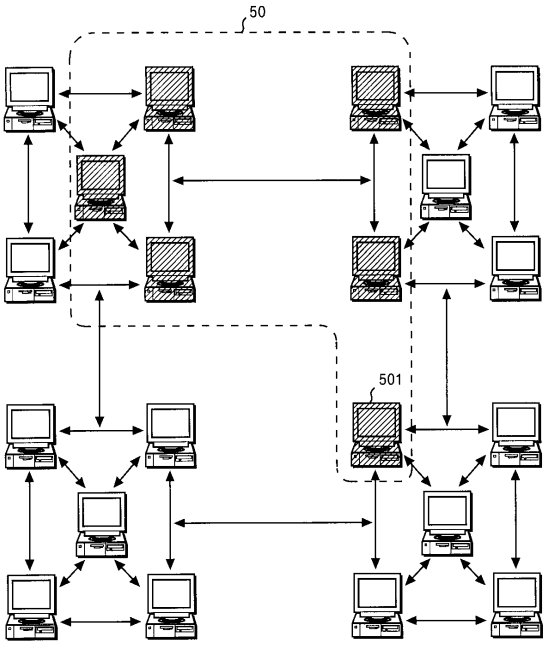
【図１３】



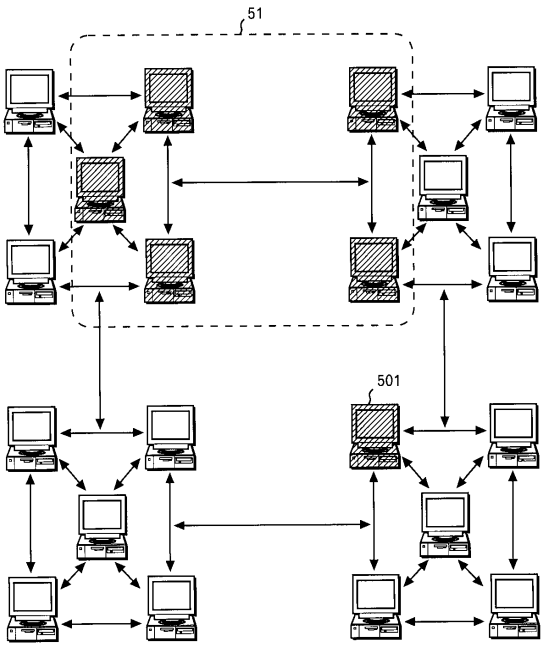
【図１４】



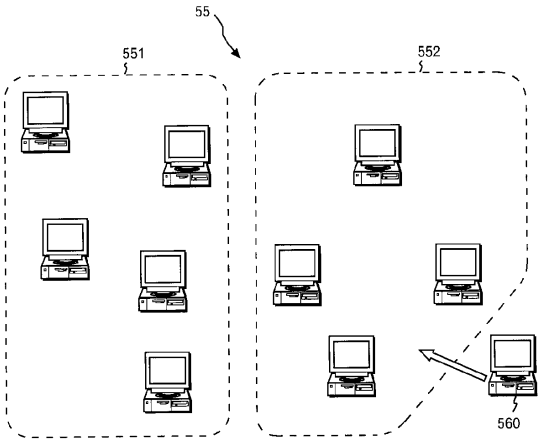
【図１５】



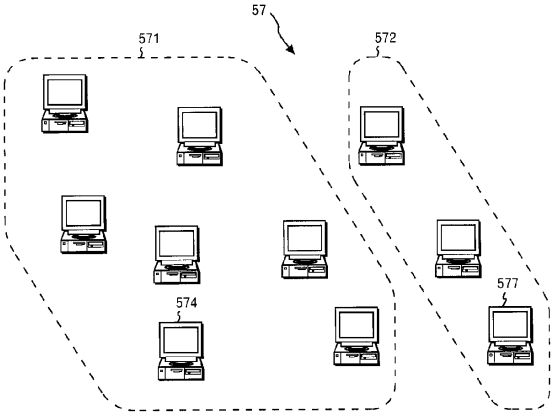
【図１６】



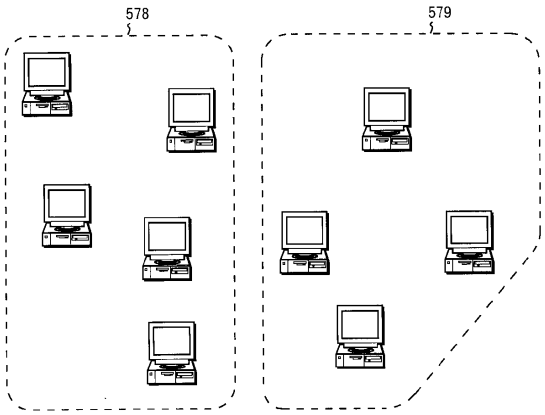
【図１７】



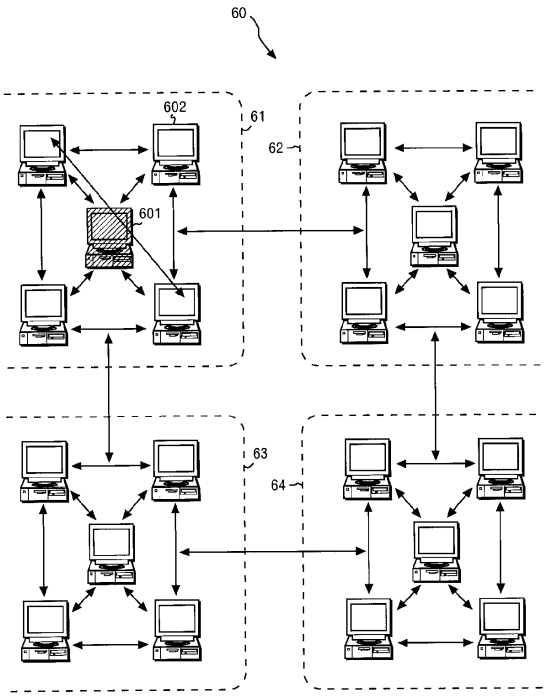
【図１８】



【図１９】



【図２０】



フロントページの続き

(74)代理人 100086243
弁理士 坂口 博
(72)発明者 ムクハージー、マハラージャ
アメリカ合衆国１２５９０、ニューヨーク州ワピンジャーズ・フォールス、ブラザーズ・ロード１２

審査官 久保 正典

(56)参考文献 米国特許第０６４００６９２（ＵＳ，Ｂ１）
特開平１０－０４０２２７（ＪＰ，Ａ）
特開２００３－０３０１６５（ＪＰ，Ａ）
特開平０３－２３２０５５（ＪＰ，Ａ）
Liu J et al., Distributed Distance Measurement for Large-Scale Networks, Computer Networks, Elsevier Science, ２００３年 2月 5日, vol.41, no.2, p177-192
Tillett, J. et al., Cluster-head identification in ad hoc sensor networks using particle swarm optimization, Personal Wireless Communications, 2002 IEEE International Conference on, IEEE, ２００２年１２月１５日, p201-205

(58)調査した分野(Int.Cl., ＤＢ名)
G06F11/16-11/20
G06F15/177