(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7182836号
(P7182836)

(45)発行日　令和4年12月5日(2022.12.5)

(24)登録日　令和4年11月25日(2022.11.25)

(51)国際特許分類

		FI		
G06F	*9/50 (2006.01)*	G06F	9/50	120Z
G06F	*9/48 (2006.01)*	G06F	9/48	300C
		G06F	9/50	120B

請求項の数　10　（全32頁）

(21)出願番号　特願2020-522300(P2020-522300)
(86)(22)出願日　平成30年11月5日(2018.11.5)
(65)公表番号　特表2021-504777(P2021-504777A)
(43)公表日　令和3年2月15日(2021.2.15)
(86)国際出願番号　PCT/IB2018/058672
(87)国際公開番号　WO2019/102286
(87)国際公開日　令和1年5月31日(2019.5.31)
審査請求日　令和3年4月23日(2021.4.23)
(31)優先権主張番号　15/819,225
(32)優先日　平成29年11月21日(2017.11.21)
(33)優先権主張国・地域又は機関
米国(US)

(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 MACHINES 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
New Orchard Road, Armonk, New York 10504, United States of America
(74)代理人　100112690
弁理士　太佐 種一

最終頁に続く

(54)【発明の名称】　分散コンピューティング環境における作業負荷の自動対角スケーリング

(57)【特許請求の範囲】

【請求項１】

分散コンピューティング環境における、少なくとも１つのプロセッサによる、作業負荷の自動対角スケーリングの方法であって、

複数のアプリケーション・インスタンスの各々に対するオペレーションの要件を計算するステップであって、前記計算された要件が垂直増加オペレーションおよび垂直低減オペレーション、ならびに水平分割オペレーションおよび水平折り畳みオペレーションを含む、前記計算するステップと、

前記垂直低減オペレーションおよび水平折り畳みオペレーションを処理するステップと、

前記処理するステップに引き続き、前記複数のアプリケーション・インスタンスを有するアプリケーションの優先度に基づいて、前記垂直増加オペレーションおよび水平分割オペレーションを順序付けるステップと、

前記順序付けられた順序に基づいて前記垂直増加オペレーションおよび水平分割オペレーションを処理し、これにより、前記分散コンピューティング環境におけるアプリケーションの効率および前記複数のリソースの利用度を最適化するステップと

を含む、前記方法。

【請求項２】

複数のアプリケーション・インスタンスの各々の複数のリソースの各々に対し、前記複数のリソースの少なくとも１つの割り当ての変更が必要かどうかを判断するステップであって、前記判断するステップが、前記複数のリソースの各々の実際の消費と、前記複数の

リソースの各々の現在の割り当てとを比較することによって行われる、前記判断するステップ
をさらに含み、該判断に応じて、前記計算するステップが実行される、請求項１に記載の方法。
【請求項３】
前記垂直増加オペレーションおよび垂直低減オペレーションが、それぞれ、アプリケーション・インスタンスに対するリソースの割り当てを増加するステップと低減するステップとを含み、
前記水平分割オペレーションおよび水平折り畳みオペレーションが、それぞれ、アプリケーションのインスタンスを生成するステップと除去するステップとを含む、
請求項１に記載の方法。
【請求項４】
スケーリング・オペレーション・コンポーネントを、
入力として、前記アプリケーションの仕様および優先度、ならびにそれらのアプリケーション・インスタンスのモニタリング情報を受信するステップと、
入力情報を組み合せることによってスケーリング・オペレーションを計算するステップと、
前記計算されたオペレーションをタスクとしてオペレーション実行キューに加えるステップであって、前記スケーリング・オペレーションの計算が前記複数のアプリケーション・インスタンスの各々に対し同時に行われる、前記加えるステップと、
のうちの少なくとも１つを行うことによって設定する、請求項３に記載の方法。
【請求項５】
リソース放出オペレーションのための、前記オペレーション実行キューの一つめを設定するステップであって、前記リソース放出オペレーションが並行して行われる、前記設定するステップと、
リソース割り当てオペレーションのための、前記オペレーション実行キューの二つめを設定するステップであって、前記割り当てオペレーションが前記優先度の前記順序付けに従って行われる、前記設定するステップと、
のうちの少なくとも１つを行うステップをさらに含む、請求項４に記載の方法。
【請求項６】
前記スケーリング・オペレーション・コンポーネントを、
入力として、前記オペレーション実行キューから複数の前記タスクを受信するステップと、
前記複数のタスクの１つ１つを実行するステップであって、前記実行するステップが、
システム・スケジューラまたはリソース・マネージャを使ってリソース割り当てオペレーションおよびリソース放出オペレーションを計算し適用するステップ、
特定ホスト上で実行されている複数のアプリケーション・インスタンスのアプリケーション・インスタンスに対するリソース消費限度を設定または修正するステップ、
前記特定のホスト上で実行されている前記アプリケーション・インスタンスを更新された利用可能なリソースに合わせ調整するステップ、および
前記特定のホスト上で前記アプリケーション・インスタンスを生成および除去するステップ、
のうちの少なくとも１つを含む、前記実行するステップと、
によって設定するステップをさらに含む、請求項４又は５に記載の方法。
【請求項７】
前記リソース割り当てを増加させることから成る各タスクに対して、前記スケーリング・オペレーション・コンポーネントによって行われる前記オペレーションの前記順序付けが、最初に前記リソース割り当てオペレーションを計算し適用するステップと、それに続いて前記リソース消費限度を設定および修正するステップと、それに続いてそれぞれのアプリケーション・インスタンスを、更新された利用可能なリソースに合わせ調整するステ

ップとを含む、請求項６に記載の方法。
【請求項８】
前記リソース割り当てを低減させることから成る各タスクに対して、前記スケーリング・オペレーション・コンポーネントによって行われる前記オペレーションの前記順序付けが、最初に前記それぞれのアプリケーション・インスタンスを前記更新された利用可能なリソースに合わせ調整するステップと、それに続いて前記リソース消費限度を設定および修正するステップと、それに続いてリソース放出オペレーションを計算し適用するステップとを含む、請求項７に記載の方法。
【請求項９】
分散コンピューティング環境における作業負荷の自動対角スケーリングのためのシステムであって、前記システムは、
メモリ中に格納された命令を実行する少なくとも１つのプロセッサを備えており、
前記プロセッサは、前記命令の実行に応じて、請求項１～７のいずれか１項に記載の方法を実行する、前記システム。
【請求項１０】
分散コンピューティング環境における、少なくとも１つのプロセッサによる作業負荷の自動対角スケーリングのためのコンピュータ・プログラムであって、前記プロセッサに、請求項１～７のいずれか１項に記載の方法を実行させるための前記コンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般にコンピューティング・システムに関し、さらに具体的には、分散コンピューティング・コンポーネント群の群内もしくは群の間またはその両方のリソース利用を最適化するための様々な実施形態に関する。
【背景技術】
【０００２】
現在の社会で、コンピュータ・システムはありふれたものである。コンピュータ・システムは、職場、家庭、または学校で見ることができる。コンピュータ・システムへの依拠、その便利さおよび携帯性が益々高まるにつれ、インターネットおよびそれへの依存も指数関数的に上昇してきている。現在、かってないほど、個人および企業は、作業負荷をこなし情報およびデータを格納するために分散型コンピューティング・システム（一般に「クラウド」と言われている）に依拠している。作業負荷の処理、リソース割り当て、およびデータ・アクセスに関する技術進展の大幅な前進が達成されてくるのに伴い、これらの機能を提供するバックエンド支援システムにおける成長および発展に対する益々増加する要求がある。
【０００３】
略してクラスタとも言われるコンピュータ・クラスタは、一緒に連結された複数の協働するコンピュータ（ソフトウェアもしくはハードウェア・リソースまたはその両方などのコンピューティング・リソースとしても知られる）によって、コンピューティング・ジョブを遂行するタイプのコンピュータ・システムである。これらの同じマネジメント領域内にあるコンピューティング・リソースは、統一されたマネジメント・ポリシーを有し、全体としてユーザにサービスを提供する。クラスタ・システム中の単一のコンピュータは、多くの場合、通常、ノードまたはコンピューティング・ノードと呼ばれる。
【０００４】
コンピューティング・クラスタは、多くの場合、ユーザによって提供され該クラスタ中で行われる作業負荷を実行するのに使われるリソースを最適化するための様々な技法を実装している。一般に用いられる１つのよくある技法は、リソースもしくはアプリケーションまたはその両方のスケーリングの使用である。対角スケーリング(DIAGONAL SCALING)は、各アプリケーションのリソース要求に適合するための、垂直スケーリングと水平ス

ケーリングとの組み合せを適用する最近のアプローチである。垂直スケーリングとは、或る特定の実行中のアプリケーション・インスタンスに対するリソースの追加または低減を言う。水平スケーリングとは、アプリケーション・インスタンスの生成または除去を言う。換言すれば、垂直スケーリングは、或るアプリケーションに対し、またはどちらかと言えばそのアプリケーションの多くの実行中のインスタンスの１つに対し、コンピューティング・クラスタの特定のリソース（例えば、メモリ、中央処理ユニット（ＣＰＵ：ｃｅｎｔｒａｌ　ｐｒｏｃｅｓｓｉｎｇ　ｕｎｉｔ）、ストレージ容量など）を割り当てまたは割り当て解除するために用いることができ、水平スケーリングは、同じアプリケーションのアプリケーション・インスタンスの１つ以上を追加または除去するために用いることができる。対角スケーリングは、諸アプリケーションのリソース要求の動的な変化に自動的に対応してコンピューティング・クラスタのリソースが充当され、効率的に利用されることを確実にするために、これら技能の両方を組み合せる。

【発明の概要】

【発明が解決しようとする課題】

【０００５】

既存の技術は、水平スケーリング、すなわちアプリケーション・インスタンスの追加および除去に重点を置く傾向がある。本明細書で説明する諸実施形態は、垂直スケーリング（アプリケーション・インスタンスに対するリソースを追加または除去する）および水平スケーリング（アプリケーション・インスタンスを追加または除去する）の両方に対するメカニズムを特徴とする。さらに、本発明は、アプリケーションおよびプラットフォームの効率を最適化するために、垂直および水平スケーリングの両方を自動的に適用し、垂直スケーリングと水平スケーリングとの間のシナジーを生成する（すなわち、対角スケーリング）統合アルゴリズムのメカニズムを提供する。

【０００６】

多くの既存の技術は、仮想マシンまたはコンテナを対象とし、アプリケーションまたはリソース・スケーリングを実装するため、（例えば、使用量を計測し、リソース要件を定義するために）それら固有の特性および能力を必要とし、一部の既存の技術は、スケーリング・オペレーションを容易にするために、ユーザにリソース（例えばＣＰＵ）の利用度に対する目標を事前設定することを要求する。かかる目標の恣意性、およびこれらの目標が、アプリケーションのサービス面のより高い品質よりもむしろインフラストラクチャのリソースを対象にしている事実は、かかるスケーリング・メカニズムの挙動の、アプリケーションのサービスの品質に対する実際の要求との整合を低下させかねない。本明細書で提供するメカニズムは、かかる事前設定された目標を使わず、代わりに、諸アプリケーションの優先度およびサービス面での品質を考慮することによって、提供されるリソースをアプリケーションの実際の負荷に整合させる。さらに、本発明のメカニズムは、アプリケーションが実行される形態に依存することなく、対角スケーリングを実装するための汎用的方法を提供する。

【０００７】

伝統的に、既存の技術は、スケーリング・オペレーションを決める際に、他のアプリケーションから分離して、各リソースを単独におよび各アプリケーションを単独に考慮する。本明細書で説明するメカニズムの機能を用いて、スケーリング・オペレーションを決めるにあたり、全ての関係するリソースの測量および集合情報ならびにアプリケーションの優先度が検討される。

【０００８】

最後に、既存の技術の一部は、スケーリング・オペレーションを行うために他の技術（例えば、利用状況指標を収集するための技術）の利用可能性を必要としそれに依拠し、また他の既存技術は、スケーリング目的のため、特定の指標に分けることなく集合指標を使って作業する。本発明は、他の技術を必要とせず、またはそれらに依存しない。さらに、本発明では、全ての関連指標の測量は、個別に適用され、次いで、スケーリング・オペレーションが、指標とアプリケーションとの統一された見地に基づいて決められる。

【課題を解決するための手段】
【０００９】
知られた既存の方法、およびその技術に改良を加える見地から、本開示で考察される新規のアルゴリズムおよび方法は、垂直スケーリングと水平スケーリングとを統一された自動メカニズムに統合し、アプリケーションおよびリソース・プラットフォームの効率を最適化することによって、自動対角スケーリングのための包括的且つ効率的な機能を提供する。
【００１０】
上記に応じて、後に説明するように、本自動対角スケーリング方法を実装するための様々な実施形態が本明細書中で開示される。単なる例として、一実施形態において、複数のアプリケーション・インスタンスの各々の複数のリソースの各々に対し、複数のリソースの少なくとも１つの割り当ての変更が必要かどうかの判断が必要とされる。複数のアプリケーション・インスタンスの各々に対しオペレーションの要件が計算され、この計算された要件は、垂直増加および低減オペレーション、ならびに水平分割および折り畳みオペレーションを含む。垂直低減および水平折り畳みオペレーションが最初に処理され、垂直増加および水平分割が順序付けられ、続いてその順序に基づいて垂直増加および水平分割が処理され、これにより、分散コンピューティング環境におけるアプリケーションの効率および複数のリソースの利用度が最適化される。
【００１１】
前述の例示的な実施形態に加え、システムおよびコンピュータ・プログラムおよびコンピュータ・プログラム製品の様々な他の実施形態が提供され、関連する利点が提示される。前述の要約は、簡単な形で概念の抜粋を紹介するために提示されたものであり、これらは後述の発明を実施するための形態においてさらに説明される。この要約は、請求される主題の重要な特徴または本質的特徴を識別することは意図されておらず、また、請求される主題の範囲を決める上での補助として使われるようにも意図されていない。
【００１２】
本発明の利点が容易に理解されるように、上記で略述された本発明のさらに具体的な説明が、添付の図面中に示された具体的な諸実施形態を参照しながら提示される。これらの図面が本発明の典型的な実施形態だけを表すもので、したがって本発明の範囲を限定するものと見なされるべきでないことを理解した上で、これら添付の図面の使用を介して、さらなる具体性と詳細さをもって本発明を記述し説明することとする。
【図面の簡単な説明】
【００１３】
【図１】本発明の或る実施形態による、例示的なコンピューティング・ノードを表すブロック図である。
【図２】本発明の或る実施形態による、例示的なクラウド・コンピューティング環境を表すさらなるブロック図である。
【図３】本発明の或る実施形態による、抽象化モデル層を表すさらなるブロック図である。
【図４】本発明の諸態様による、分散コンピューティング環境中の作業負荷の自動対角スケーリングの方法を表すフローチャート図を示す。
【図５】本発明の諸態様による、対角スケーリング・モデルを表す組み合せブロック／フローチャート図を示す。
【図６】本発明の諸態様による、分散コンピューティング環境中の作業負荷の自動対角スケーリングの方法を表すさらなるフローチャート図を示す。
【図７】本発明の諸態様による、複数のリソース種類の各々に対する例示的な計算メカニズムを表すブロック図を示す。
【図８】本発明の諸態様による、複数のリソース種類の各々に対する例示的な計算メカニズムを表すさらなるブロック図を示す。
【図９】本発明の諸態様による、複数のリソース種類の各々に対する例示的な計算メカニズムを表すまださらなるブロック図を示す。

【図１０】本発明の諸態様による、過去のデータに基づく、リソースの自動的且つ適応的増加および低減の方法を表す組み合せブロック／フローチャート図を示す。
【図１１】本発明の諸態様による、例示的な対角スケーリング増加オペレーションを表すブロック図を示す。
【図１２】本発明の諸態様による、アプリケーション・インスタンスのリソース割り当てに対する、計算された増加オペレーションを処理する方法を表すフローチャート図を示す。
【図１３】本発明の諸態様による、例示的な対角スケーリング低減オペレーションを表すブロック図を示す。
【図１４】本発明の諸態様による、対角スケーリング・アルゴリズムを表すフローチャート図を示す。
【図１５】本発明の諸態様による、アプリケーションの優先度を計算するための、例示的なアプリケーションのトポロジを表すブロック図を示す。
【図１６】本発明の諸態様による、分散コンピューティング環境中の作業負荷の自動対角スケーリングの方法を表す組み合せブロック／フローチャート図を示す。
【発明を実施するための形態】
【００１４】
前述したように、コンピューティング・クラスタは、多くの場合、ユーザによって提供され該クラスタ中で行われる作業負荷を実行するのに使われるリソースを最適化するための様々な技法を実装している。一般に用いられる１つのよくある技法は、リソースもしくはアプリケーションまたはその両方のスケーリングの使用である。対角スケーリングは、各アプリケーションのリソース要求に適合するための、垂直スケーリングと水平スケーリングとの組み合せを適用する最近のアプローチである。垂直スケーリングとは、或る特定の実行中のアプリケーション・インスタンスに対するリソースの追加または低減を言う。水平スケーリングとは、アプリケーション・インスタンスの生成または除去を言う。換言すれば、垂直スケーリングは、或るアプリケーションに対し、またはどちらかと言えばそのアプリケーションの多くの実行中のインスタンスの１つに対し、コンピューティング・クラスタの特定のリソース（例えば、メモリ、中央処理ユニット（ＣＰＵ）、ストレージ容量など）を割り当てまたは割り当て解除するために用いることができ、水平スケーリングは、同じアプリケーションのアプリケーション・インスタンスの１つ以上を追加または除去するために用いることができる。対角スケーリングは、諸アプリケーションのリソース要求の動的な変化に自動的に対応してコンピューティング・クラスタのリソースが充当され、効率的に利用されることを確実にするために、これら技能の両方を組み合せるが、但し、この種のスケーリングを自動的に、効率的に、且つ最適に統合する現行のソリューションは存在しない。
【００１５】
上記に応じ、本発明は、自動対角スケーリングを使って、分散コンピューティング環境においてリソースをもっと効率的に最適化して利用する機能を用いる。すなわち、本開示の諸実施形態は、自動対角スケーリングのための効率的なメカニズムを用い、本メカニズムは以下の代表的な仕様を有する。第一に、アプリケーション・インスタンスの実際のリソース消費が自動的に追跡され、そのアプリケーション・インスタンスに割り当てられたリソースと比較される。第二に、割り当てリソースとそのリソース限度が、比較された消費量に従ってこれらリソースの割り当てに自動的に適合（増加／低減）される。さらに具体的には、アプリケーションの作業負荷が増大すると、本明細書で説明するメカニズムが、そのアプリケーションに対し追加のリソースを利用可能にする。同様に、作業負荷が減ると、リソースは低減される。第三に、アプリケーション・インスタンスもしくはホストまたはその両方の状態およびポリシーに従って、垂直および水平スケーリングが自動的に用いられる。第四にそして最後に、本明細書の機能は、その設定がカスタム化可能で効率的で容易である。例えば、ユーザが設定可能な項目には、（例えば、過去の統計、およびコスト制約に基づく）各種リソースの消費の最大限度、各種リソースの利用可能性に対する最小限度、スケーリング・オペレーションが必要なことを判断するためのトリガ、およ

び垂直および水平スケーリングの統合に対するポリシーが含まれてよい。
【００１６】
提案されたメカニズムは、これらの仕様によって以下の利益を提供する。第一に、アプリケーションのスループットが、アプリケーションへの入力として供給される実際の作業負荷、アプリケーションの優先度、および利用可能なリソースに従って最適化される。第二に、アプリケーションに割り当てられたリソースのコストが最小化され、さらにクラスタのリソースの利用度が向上する。しかして、クラスタ・リソースのより効率的な利用によって、クラスタは、追加の作業負荷に、かかる作業負荷の実行コストを低減しながら対応することが可能になる。第三に、クラスタを利用する顧客は、彼らそれぞれの作業負荷を履行するため実際に必要となりまたは使用されたリソースに対してだけ支払いをし、しかして未使用のリソースに対する過剰支払いが回避される。さらに、クラウドのフレキシビリティおよびその金額の課金メカニズムが改良される。第四に、作業負荷に従ってリソース割り当ておよびスケーラビリティを見積もり、それらをマニュアルで調整する必要性を除去することにより、アプリケーションのおよび加えてその設定の効率が向上する。第五にそして最後に、本明細書のメカニズムは、様々な作業負荷マネジメント・システムに追加しまたは実装することが可能な汎用的自動対角スケーリング機能を使用する。
【００１７】
なお、本開示では、簡潔さのため、「リソース」と言う言葉を多用する。本発明の実際の実装では、本明細書中のリソースという用語は、ＣＰＵ、グラフィカル処理ユニット（ＧＰＵ：ｇｒａｐｈｉｃａｌ　ｐｒｏｃｅｓｓｉｎｇ　ｕｎｉｔ）、メモリ、ストレージ・デバイス、ネットワーク・デバイス、アクセラレータ・デバイスから、さらにはコンピューティング・ノード全体から成り得る。当業者には当然であろうが、実際上は、当該技術分野で周知の一切のハードウェアもしくはソフトウェアまたはその両方が、本明細書に記載の「リソース」または「リソースの種類」と互換的に解釈されることになる。さらに、本開示は、アプリケーションの「アプリケーション・インスタンス」を説明する。当業者なら認識しているであろうが、アプリケーション・インスタンスは、アプリケーションの、特定の履行または実行されているアプリケーションの個々の生起を言及することが意味されているが、但し、アプリケーションまたはそのアプリケーション・インスタンスの性質は、本明細書で開示される機能の個々の実装によって広範に変化し得る。
【００１８】
さらに、前もって当然のことながら、本開示はクラウド・コンピューティングの詳細な記述を含むが、本明細書で述べる教示の実装は、クラウド・コンピューティング環境に限定されない。それどころか、本発明の諸実施形態は、現在既知のまたは今後開発される任意の種類のコンピューティング環境に連結して実装することが可能である。
【００１９】
クラウド・コンピューティングは、最小の管理作業またはサービスのプロバイダとのやり取りで、迅速に立ち上げてリリースすることができる、構成可能なコンピューティング・リソース（例えば、ネットワーク、ネットワーク帯域、サーバ、処理、メモリ、ストレージ、アプリケーション、仮想マシン、およびサービス）の共用のプールへの、便利でオンデマンドのネットワーク・アクセスを可能にするためのサービス・デリバリのーモデルである。このクラウド・モデルは、少なくとも５つの特徴と、少なくとも３つのサービス・モデルと、少なくとも４つの展開モデルとを含むことができる。
【００２０】
特徴は次の通りである。
オンデマンド・セルフサービス：クラウド・コンシューマは、サービスのプロバイダとの人間のやり取りなしに、必要な場合は自動的に、サーバ時間およびネットワーク・ストレージなどのコンピューティング能力を一方的にセットアップすることができる。
広範なネットワーク・アクセス：諸能力はネットワークを介して利用可能で、異種から成るシンまたはシック・クライアント・プラットフォーム（例えば、携帯電話、ラップトップ、およびＰＤＡ）による使用を推進する標準的なメカニズムを通してアクセスされる。

リソースのプール化：プロバイダのコンピューティング・リソースは、マルチテナント・モデルを用いる複数のコンシューマにサービスするために、デマンドに従って動的に割り当ておよび再割り当てされる各種の物理的および仮想のリソースとしてプール化される。一般にコンシューマは提供されたリソースの正確な場所の制御または知識を持たない、という点で、場所無異存性の感覚があるが、抽象化のより高位レベルでは場所（例えば、国、州、またはデータセンタ）を特定することを可能にできる。

敏速な伸縮性：諸能力は、迅速にスケール・アウトするため、場合によっては自動的に、敏速且つ伸縮自在にセットアップでき、迅速にスケール・インするために敏速にリリースされる。コンシューマには、セットアップのため利用可能な諸能力が多くの場合無制限に見え、いつでもどのような量でも購入が可能である。

計量されるサービス：クラウド・システムは、サービスの種類（例えば、ストレージ、処理、帯域幅、および有効ユーザ・アカウント）に適した抽象化のいずれかのレベルで、計量機能を利用することによって、リソース使用を自動的に管理し、最適化する。リソース利用は、プロバイダと、利用されるサービスのコンシューマとの双方に透明性を提供しながら、モニタし、管理し、報告することができる。

【００２１】

サービス・モデルは次のとおりである。

サービスとしてのソフトウェア（ＳａａＳ：Ｓｏｆｔｗａｒｅ ａｓ ａ Ｓｅｒｖｉｃｅ）：コンシューマに提供される能力は、クラウド・インフラストラクチャ上で実行されているプロバイダのアプリケーションを使うことである。これらのアプリケーションは、様々なクライアント・デバイスから、ウェブ・ブラウザ（例えば、ウェブベースのｅメール）などのシン・クライアント・インターフェースを介してアクセス可能である。コンシューマは、限られたユーザ固有のアプリケーション構成設定のあり得る例外を除いて、ネットワーク、サーバ、オペレーティング・システム、ストレージ、または個別のアプリケーション機能でさえも含め、根底にあるクラウド・インフラストラクチャを管理または制御はしない。

サービスとしてのプラットフォーム（ＰａａＳ：Ｐｌａｔｆｏｒｍ ａｓ ａ Ｓｅｒｖｉｃｅ）：コンシューマに提供される能力は、クラウド・インフラストラクチャ上に、プロバイダによってサポートされるプログラミング言語およびツールを使って生成された、コンシューマ生成の、またはコンシューマ取得のアプリケーションを展開することである。コンシューマは、ネットワーク、サーバ、オペレーティング・システム、またはストレージを含め、根底にあるクラウド・インフラストラクチャを管理または制御することはないが、これら展開されるアプリケーション、およびおそらくはアプリケーションのホスティング環境設定に対する制御を有する。

サービスとしてのインフラストラクチャ（ＩａａＳ：Ｉｎｆｒａｓｔｒｕｃｔｕｒｅ ａｓ ａ Ｓｅｒｖｉｃｅ）：コンシューマに提供される能力は、コンシューマが、処理、ストレージ、ネットワーク、およびオペレーティング・システムおよびアプリケーションを含み得る、任意のソフトウェアを展開し実行することが可能な他の基本的コンピューティング・リソースをセットアップすることである。コンシューマは、根底にあるクラウド・インフラストラクチャを管理または制御することはないが、オペレーティング・システム、ストレージ、展開されるアプリケーションに対する制御、およびおそらくは選択ネットワーク・コンポーネント（例えば、ホストのファイアウォール）の限定された制御を有する。

【００２２】

展開モデルは次のとおりである。

プライベート・クラウド：このクラウド・インフラストラクチャは、一組織のためだけに運営される。これは、その組織または第三者によって管理されてよく、自組織構内に所在しても自組織構外に所在してもよい。

コミュニティ・クラウド：このクラウド・インフラストラクチャは、いくつかの組織によって共有され、共有の利害関係（例えば、任務、安全要件、指針、およびコンプライア

ンス配慮事項）を有する特定のコミュニティをサポートする。これは、これらの組織または第三者によって管理されてよく、これらの組織構内に所在してもこれらの組織構外に所在してもよい。

パブリック・クラウド：このクラウド・インフラストラクチャは、一般公衆または大きな産業グループに利用可能にされており、クラウド・サービスを販売する組織によって所有されている。

ハイブリッド・クラウド：このクラウド・インフラストラクチャは、独自のエンティティに留まりながら、データおよびアプリケーションの可搬性（例えば、クラウド間の負荷バランスのためのクラウド・バースティング）を可能にする標準化されたまたは独自の技術によって一緒に結ばれた２つ以上のクラウド（プライベート、コミュニティ、またはパブリック）の合成体である。

【００２３】

クラウド・コンピューティング環境は、ステートレスネス、弱連結、モジュール性、および意味相互運用性に焦点を合わせて方向付けられたサービスである。クラウド・コンピューティングの中心には、相互接続されたノードのネットワークを含むインフラストラクチャが在る。

【００２４】

ここで図１を参照すると、クラウド・コンピューティング・ノードの一例の概略図が示されている。クラウド・コンピューティング・ノード１０は、適したクラウド・コンピューティング・ノードの単なる一例であって、本明細書に記載の本発明の諸実施形態の用途または機能の範囲に関し、いかなる限定をも示唆することは意図されていない。上記に関係なく、クラウド・コンピューティング・ノード１０は、前述した機能のセットのいずれをも実装するもしくは実行するまたはその両方を行うことが可能である。

【００２５】

クラウド・コンピューティング・ノード１０中には、コンピュータ・システム／サーバ１２が在り、これは、数多くの他の汎用または特殊用途コンピューティング・システム環境または構成と共に作動可能である。コンピュータ・システム／サーバ１２と共に使用するのに適し得る周知のコンピューティング・システム、環境、もしくは構成、またはこれらの組み合せの例には、以下に限らないが、パーソナル・コンピュータ・システム、サーバ・コンピュータ・システム、シン・クライアント、シック・クライアント、ハンドヘルドまたはラップトップ・デバイス、マルチプロセッサ・システム、マイクロプロセッサベースのシステム、セット・トップ・ボックス、プログラマブル・コンシューマ電子機器、ネットワークＰＣ、ミニコンピュータ・システム、メインフレーム・コンピュータ・システム、および上記のシステムまたはデバイスのいずれかを含む分散クラウド・コンピューティング環境などが含まれる。

【００２６】

コンピュータ・システム／サーバ１２は、コンピュータ・システムによって実行される、プログラム・モジュールなどのコンピュータ・システム実行可能命令の一般的コンテキストで表すことができる。一般に、プログラム・モジュールは、特定のタスクを行う、または特定の抽象データ型を実装する、ルーティン、プログラム、オブジェクト、コンポーネント、ロジック、データ構造体などを含んでよい。コンピュータ・システム／サーバ１２は、通信ネットワークを介して連結された遠隔の諸処理デバイスによってタスクが行われる分散クラウド・コンピューティング環境中で実行することができる。分散クラウド・コンピューティング環境中で、プログラム・モジュールは、ローカルおよび遠隔のコンピュータ・システム両方の、メモリ・ストレージ・デバイスを含むストレージ媒体に配置することが可能である。

【００２７】

図１に示されるように、クラウド・コンピューティング・ノード１０中のコンピュータ・システム／サーバ１２は、汎用コンピューティング・デバイスの形で示されている。コンピュータ・システム／サーバ１２のコンポーネントには、以下に限らないが、１つ以上

のプロセッサまたは処理ユニット１６、システム・メモリ２８、およびシステム・メモリ２８とプロセッサ１６との間を含め、様々なシステム・コンポーネントを連結しているバス１８を含めることができる。

【００２８】

バス１８は、メモリ・バスまたはメモリ・コントローラ、周辺バス、アクセラレーテッド・グラフィックス・ポート、および様々なバス・アーキテクチャのいずれかを使った、プロセッサまたはローカル・バスを含む、いくつかの型のバス構造体のいずれか１つ以上を表す。限定でなく例として、かかるアーキテクチャは、業界標準アーキテクチャ（ＩＳＡ：Ｉｎｄｕｓｔｒｙ Ｓｔａｎｄａｒｄ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、マイクロ・チャネル・アーキテクチャ（ＭＣＡ：Ｍｉｃｒｏ Ｃｈａｎｎｅｌ Ａｒｃｈｉｔｅｃｔｕｒｅ）バス、拡張型ＩＳＡ（ＥＩＳＡ：Ｅｎｈａｎｃｅｄ ＩＳＡ）バス、ビデオ・エレクトロニクス・スタンダーズ・アソシエーション（ＶＥＳＡ：Ｖｉｄｅｏ Ｅｌｅｃｔｒｏｎｉｃｓ Ｓｔａｎｄａｒｄｓ Ａｓｓｏｃｉａｔｉｏｎ）ローカル・バス、および周辺コンポーネント相互接続（ＰＣＩ：Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ）バスを含む。

【００２９】

コンピュータ・システム／サーバ１２は、通常、様々なコンピュータ・システム可読媒体を含む。かかる媒体は、コンピュータ・システム／サーバ１２によってアクセス可能な任意の利用可能な媒体であってよく、これには、揮発性および不揮発性媒体、ならびにリムーバブルおよび固定型媒体の両方が含まれる。

【００３０】

システム・メモリ２８は、ランダム・アクセス・メモリ（ＲＡＭ：ｒａｎｄｏｍ ａｃｃｅｓｓ ｍｅｍｏｒｙ）３０もしくはキャッシュ・メモリ３２またはその両方など、揮発性メモリの形のコンピュータ・システム可読媒体を含んでよい。コンピュータ・システム／サーバ１２は、他のリムーバブル／固定型、揮発性／不揮発性コンピュータ・システム・ストレージ媒体をさらに含むことが可能である。単なる例として、ストレージ・システム３４は、固定型、不揮発性磁気媒体（図示せず、通常、「ハード・ドライブ」と呼ばれる）からの読み取りおよびそれに書き込むように設けることができる。図示はされていないが、リムーバブルな不揮発性磁気ディスク（例えば、「フレキシブル・ディスク」）から読み取り、これに書き込むための磁気ディスク・ドライブ、およびＣＤ－ＲＯＭ、ＤＶＤ－ＲＯＭ、または他の光媒体など、リムーバブルな不揮発性光ディスクから読み取り、これに書き込むための光ディスク・ドライブを設けることが可能である。かかる場合、それぞれを、１つ以上のデータ媒体インターフェースによってバス１８に接続することができる。以下に、さらに示し説明するように、システム・メモリ２８は、本発明の実施形態の諸機能を遂行するよう構成されたプログラム・モジュールの（例えば、少なくとも１つの）セットを有する、少なくとも１つのプログラム製品を含むことができる。

【００３１】

限定でなく例として、プログラム・モジュール４２の（少なくとも１つの）セットを有するプログラム／ユーティリティ４０、ならびにオペレーティング・システム、１つ以上アプリケーション・プログラム、他のプログラム・モジュール、およびプログラム・データをシステム・メモリ２８中に格納することが可能である。各オペレーティング・システム、１つ以上アプリケーション・プログラム、他のプログラム・モジュール、およびプログラム・データ、またはこれらのいくつかの組み合せは、ネットワーキング環境での実装を含むことが可能である。プログラム・モジュール４２は、一般に、本明細書で説明した本発明の実施形態の機能もしくは方法またはその両方を遂行する。

【００３２】

また、コンピュータ・システム／サーバ１２は、キーボード、ポインティング・デバイス、ディスプレイ２４など１つ以上の外部デバイス１４；ユーザがコンピュータ・システム／サーバ１２とやり取りできるようにする１つ以上のデバイス；もしくはコンピュータ・システム／サーバ１２が１つ以上の他のコンピューティング・デバイスと通信すること

を可能にする任意のデバイス（例えば、ネットワーク・カード、モデムなど）；またはこれらの組み合せと通信することができる。かかる通信は、入力／出力（Ｉ／Ｏ：Ｉｎｐｕｔ／Ｏｕｔｐｕｔ）インターフェース２２を介して行うことが可能である。さらにまた、コンピュータ・システム／サーバ１２は、ローカル・エリア・ネットワーク（ＬＡＮ：ｌｏｃａｌ　ａｒｅａ　ｎｅｔｗｏｒｋ）、一般広域ネットワーク（ＷＡＮ：ｗｉｄｅ　ａｒｅａ　ｎｅｔｗｏｒｋ）もしくは公衆ネットワーク（例えばインターネット）またはこれらの組み合せなど、１つ以上のネットワークとネットワーク・アダプタ２０を介して通信することができる。図示のように、ネットワーク・アダプタ２０は、バス１８を介してコンピュータ・システム／サーバ１２の他のコンポーネントと通信する。図示はされていないが、当然のことながら、コンピュータ・システム／サーバ１２に関連させて他のハードウェアもしくはソフトウェア・コンポーネントまたはその両方を用いることができよう。諸例には、以下に限らないが、マイクロコード、デバイス・ドライブ、冗長処理ユニット、外部ディスク・ドライブ・アレイ、ＲＡＩＤシステム、テープ・ドライブ、データ・アーカイバル・ストレージ・システムなどが含まれる。

【００３３】

本発明のコンテキストにおいて、また当業者ならよく理解しているように、図１に表された様々なコンポーネントは、本明細書で教示する対角スケーリング機能を用いて分散作業負荷を処理するのに使用することができる。例えば、例示の実施形態のメカニズムに関連する処理およびデータ・ストレージ能力の一部は、ローカルの処理コンポーネントを介してローカルに行うことが可能であり、同時に、それらコンポーネントは、本発明の様々な目的を達成するために、ネットワークを介して遠隔に配置された分散されたコンピューティング・データ処理およびストレージ・コンポーネントに連結される。これもまた、当業者には当然のことながら、上記の例示は、様々な本発明の態様を達成する分散コンピューティング・コンポーネントの可能な全体的連結ネットワークの単なる一サブセットを示すことが意図されている。

【００３４】

ここで図２を参照すると、例示的なクラウド・コンピューティング環境５０が描かれている。図示のように、クラウド・コンピューティング環境５０は、例えば、携帯情報端末（ＰＤＡ：ｐｅｒｓｏｎａｌ　ｄｉｇｉｔａｌ　ａｓｓｉｓｔａｎｔ）または携帯電話５４Ａ、デスクトップ・コンピュータ５４Ｂ、ラップトップ・コンピュータ５４Ｃ、もしくは自動車コンピュータ・システム５４Ｎ、またはこれらの組み合せなど、クラウドのコンシューマによって使われるローカル・コンピューティング・デバイスが通信可能な１つ以上のクラウド・コンピューティング・ノード１０を含む。諸ノード１０は、相互に通信が可能である。これらは、前述のように、プライベート、コミュニティ、パブリック、またはハイブリッド・クラウド、またはこれらの組み合せなど、１つ以上のネットワーク中に、物理的にまたは仮想的にグループ化する（図示せず）ことができる。これは、クラウド・コンピューティング環境５０が、インフラストラクチャ、プラットフォーム、もしくはサービスとしてのソフトウェア、またはこれらの組み合せを提供することを可能にし、それらに対し、クラウド・コンシューマは、ローカルのコンピューティング・デバイスにリソースを維持する必要はない。当然のことながら、図２に示されたコンピューティング・デバイス５４Ａ～Ｎの種類は例示だけが意図されており、また、コンピューティング・ノード１０およびクラウド・コンピューティング環境５０は、任意の種類のネットワーク、もしくは（例えば、ウェブ・ブラウザを使って）ネットワーク・アドレス可能な接続またはその両方を介して、任意の種類のコンピュータ化デバイスと通信することが可能である。

【００３５】

ここで、図３を参照すると、クラウド・コンピューティング環境５０（図２）によって設けられた機能抽象化層のセットが示されている。前もって当然のことながら、図３中に示されたコンポーネント、層、および機能は、例示だけを意図されており、本発明の諸実施形態はこれに限定されない。図示のように、以下の層と対応する機能とが提示されている。

【００３６】
デバイス層５５は、クラウド・コンピューティング環境５０で様々なタスクを実行するため、センサ、アクチュエータ、および他のオブジェクトを組み込んだ、もしくはスタンドアローンの電子機器またはその両方の、物理もしくは仮想デバイスまたはその両方を含む。デバイス層５５中のデバイスの各々は、他の機能抽象化層とのネットワーキング機能を組み込み、しかして、これらデバイスから得られた情報をそれらの層に提供することができ、もしくは他の抽象化層からの情報をこれらデバイスに提供することができ、またはその両方が可能である。一実施形態において、デバイス層５５を含めこれら様々のデバイスには、「モノのインターネット」（ＩｏＴ：ｉｎｔｅｒｎｅｔ ｏｆ ｔｈｉｎｇｓ）と総称されるエンティティのネットワークを組み込むことが可能である。当業者には当然のことながら、かかるエンティティのネットワークは、種々様々な目的を達成するためにデータの相互通信、収集、および配布を可能にする。
【００３７】
図示のデバイス層５５は、示されるように、センサ５２、アクチュエータ５３、統合処理を備えた「学習」サーモスタット５６、センサ、ネットワーキング電子機器、カメラ５７、制御可能家庭用コンセント／ソケット５８、および制御可能電気スイッチ５９を含む。他の可能なデバイスには、以下に限らないが、様々な追加のセンサ・デバイス、ネットワーク・デバイス、（遠隔制御デバイスなどの）電子デバイス、追加のアクチュエータ・デバイス、冷蔵庫または洗濯機／乾燥機などのいわゆる「スマート」電気製品、および多様な他の可能な相互連結されたオブジェクトを含めることができる。
【００３８】
ハードウェアおよびソフトウェア層６０は、ハードウェアおよびソフトウェア・コンポーネントを含む。ハードウェア・コンポーネントの例は、メインフレーム６１、ＲＩＳＣ（Ｒｅｄｕｃｅｄ Ｉｎｓｔｒｕｃｔｉｏｎ Ｓｅｔ Ｃｏｍｐｕｔｅｒ（縮小命令セット・コンピュータ））アーキテクチャ・ベースのサーバ６２、サーバ６３、ブレード・サーバ６４、ストレージ・デバイス６５、ならびにネットワークおよびネットワーク形成コンポーネント６６、を含む。いくつかの実施形態において、ソフトウェア・コンポーネントは、ネットワーク・アプリケーション・サーバ・ソフトウェア６７、およびデータベース・ソフトウェア６８を含む。
【００３９】
仮想化層７０は、当該層から仮想エンティティの以下の例を設けることが可能な抽象化層を提供する：仮想サーバ７１、仮想ストレージ７２、仮想プライベート・ネットワークを含む仮想ネットワーク７３、仮想アプリケーションおよびオペレーティング・システム７４、ならびに仮想クライアント７５。
【００４０】
一例において、管理層８０は、以下に記載の機能を提供することができる。リソース・プロビジョニング８１は、クラウド・コンピューティング環境内のタスクを実施するのに用いられる、コンピューティング・リソースおよび他のリソースの動的な調達を提供する。計量および課金８２は、リソースがクラウド・コンピューティング環境内で用いられる際のコストの追跡、およびそれらリソースの消費に対する請求書または使用明細書作成を提供する。一例においては、これらのリソースは、アプリケーション・ソフトウェアのライセンスを含み得る。セキュリティは、クラウド・コンシューマおよびタスクに対する本人確認、並びにデータおよび他のリソースに対する保護を提供する。ユーザ・ポータル８３は、コンシューマおよびシステム管理者に対し、クラウド・コンピューティング環境へのアクセスを提供する。サービス品質管理８４は、必要なサービス・レベルが満たされるように、クラウド・コンピューティング・リソースの割り当ておよび管理を提供する。サービス品質保証契約（ＳＬＡ：Ｓｅｒｖｉｃｅ Ｌｅｖｅｌ Ａｇｒｅｅｍｅｎｔ）計画および達成８５は、ＳＬＡによって予期される今後の要求のためのクラウド・コンピューティング・リソースの事前準備および調達を提供する。
【００４１】

作業負荷層９０は、クラウド・コンピューティング環境を使用することが可能な機能の事例を提供する。この層から提供できる作業負荷および機能の例には、マッピングおよびナビゲーション９１、ソフトウェアの開発およびライフサイクル管理９２、仮想クラスルーム教育配信９３、データ分析処理９４；トランザクション処理９５、ならびに本発明の例示の実施形態のコンテキストにおいて、様々なリソースおよびアプリケーションのスケーリング機能９６が含まれる。さらに、リソースおよびアプリケーション・スケーリング機能９６は、特定の作業負荷にリソースを割り当てまたは割り当て解除するため、もしくは特定の作業負荷のアプリケーション・インスタンスを生成または除去するため、またはその両方のために特定のデータを解析するなどのオペレーションを含んでよく、これについてはさらに説明することとする。また、当業者には当然のことながら、リソースおよびアプリケーション・スケーリング機能９６は、本発明の例示の諸実施形態の様々な目的を達成するために、ハードウェアおよびソフトウェア６０、仮想化７０、管理８０、および（例えば、データ解析処理９４などの）他の作業負荷９０中の部分など、様々な抽象化層の他の部分と共に機能することができる。

【００４２】

続いて、図４は、本発明の一実施形態による、分散コンピューティング環境中の作業負荷の自動対角スケーリングの方法４００を示す。方法４００（および後記で説明する追加の方法）は、様々な実施形態において、とりわけ、図１～３に表された環境のいずれでも、本発明に従って実行することが可能である。当然のことながら、当業者が本明細書を読めば理解できるように、方法４００には、図４中に具体的に記載されたものよりも多いまたは少ないオペレーションが含まれてよい。

【００４３】

方法４００（および後記で説明する追加の方法）の各ステップは、上記のオペレーティング環境のどの適切なコンポーネントによっても実行することが可能である。例えば、様々な実施形態において、方法４００は、部分的にまたは全体的に、一プロセッサによって、または１つ以上のプロセッサを有する何らかの他のデバイスによって実行することができる。方法４００の１つ以上のステップを実行するための任意のデバイス中には、例えば、処理回路（１つまたは複数）、チップ（１つまたは複数）、もしくは、ハードウェアもしくはソフトウェアまたはこれらの組み合せに実装され、望ましくは少なくとも１つのハードウェア・コンポーネントを有するモジュール（１つまたは複数）、または前述の組み合せ、などの処理装置を用いることができる。処理装置の例には、以下に限らないが、ＣＰＵ、特定用途向け集積回路（ＡＳＩＣ：ａｐｐｌｉｃａｔｉｏｎ ｓｐｅｃｉｆｉｃ ｉｎｔｅｇｒａｔｅｄ ｃｉｒｃｕｉｔ）、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ：Ｆｉｅｌｄ Ｐｒｏｇｒａｍｍａｂｌｅ Ｇａｔｅ Ａｒｒａｙ）など、これらの組み合せ、または、当該技術分野で既知の任意の他の適切なコンピューティング・デバイスが含まれる。

【００４４】

方法４００は、複数のアプリケーション・インスタンスの各々の複数のリソースの各々に対し、該複数リソースの少なくとも１つへの割り当てを変更する必要があるかどうかを判断する（ステップ４０４）ことで開始される（ステップ４０２）。複数のアプリケーション・インスタンスの各々に対するオペレーションの要件が計算され、該計算された要件は垂直増加および低減オペレーション、ならびに水平分割および折り畳みオペレーションを含む（ステップ４０６）。垂直低減および水平折り畳みオペレーションが最初に処理され（ステップ４０８）、垂直増加および水平分割が順序付けられ（ステップ４１０）、続いてその順序に基づいて垂直増加および水平分割が処理され（ステップ４１２）、これにより、分散コンピューティング環境における、アプリケーション効率および複数のリソースの利用度が最適化される。方法４００が終了する（ステップ４１４）

【００４５】

対角スケーリングの一般的アプローチは、アプリケーション・インスタンスを垂直にスケールし（リソースを割り当てる、または割り当て解除する）、アプリケーション・イン

スタンスまたはホストが飽和状態のときまたはアプリケーションが遊休状態のときは、水平にスケールする（アプリケーション・インスタンスを生成する、または除去する）ことである。水平スケーリングの後、続いてアプリケーション・インスタンスを垂直にスケールする。この実装が図５に表されており、これは、対角スケーリング・方法５００を表す組み合せブロック／フローチャート図である。

【００４６】

方法５００は、アプリケーション・インスタンスに対し垂直スケーリングを適用する（ステップ５０４）ことで開始される（ステップ５０２）。すなわち、或るアプリケーション・インスタンスが必要とするリソース要件を満たすために、そのアプリケーション・インスタンスにリソースが割り当てられ、もしくはリソースが割り当て解除される。アプリケーション・インスタンスが飽和状態（フル使用状態）のとき、もしくはホストが飽和状態のとき、方法５００はそのアプリケーション・インスタンスに対し垂直スケーリング・オペレーションを適用すること（ステップ５０６）ができる。すなわち、ステップ５０４でリソース割り当てまたは割り当て解除の垂直スケーリングを行い、それでもフル飽和状態または遊休状態であると判断されると、次いで、当該アプリケーション・インスタンスは、そのアプリケーション・インスタンスを追加するかもしくは除去することによって水平にスケールされる。水平スケーリング・オペレーション（ステップ５０６）に続いて、本方法は、必要に応じ垂直スケーリング・オペレーション（ステップ５０４）の適用に戻る（且つその後も適用を続ける）。

【００４７】

図６は、本発明の諸態様による、分散コンピューティング環境中の作業負荷の自動対角スケーリングの方法６００を表すさらなるフローチャート図を示す。方法６００は、１つ以上のアプリケーション・インスタンスの各々のリソース消費を自動的に追跡し、そのアプリケーション・インスタンスの各々の消費を当該アプリケーション・インスタンスのリソース割り当てと比較する（ステップ６０２）ことで開始される。次いで、方法６００は、ステップ６０２での比較の結果に従って、そのアプリケーション・インスタンスへのリソース割り当ての修正オペレーション（増加または低減）の計算（ステップ６０４）へと続く。次に、計算された修正オペレーションは、アプリケーションもしくはアプリケーション・インスタンスまたはその両方の優先度と、利用可能なリソースとに照らして改善される（ステップ６０６）。次いで、計算された修正オペレーションが、それぞれ、諸アプリケーション・インスタンスに動的に適用され（ステップ６０８）、この修正オペレーションはステップ６１０および６１２に示されるように多様な種類がある。

【００４８】

適用が可能な１つの種類のオペレーションは、或る特定のアプリケーション・インスタンスに割り当てられたリソースの割り当て量およびリソースの限度を増加するまたは低減することを含む。このようにして、アプリケーションの負荷が増大したとき、その特定のアプリケーション・インスタンスに対し追加のリソースが利用可能にされる。同様に、特定のアプリケーション・インスタンスの負荷が減少したとき、そのアプリケーション・インスタンスに不必要または使用されない過剰なリソースは、その特定のアプリケーション・インスタンスから割り当て解除される。ステップ６０８に戻って、適用可能な別種の修正オペレーションは、アプリケーション・インスタンスを追加し、除去し、先取りし、または移動すること（（ステップ６１２）を含む。例えば、或るアプリケーション・インスタンスがステートレスな場合、そのステートレスなアプリケーション・インスタンスは容易に除去または先取りすることが可能である。逆に、或るアプリケーション・インスタンスがステートフルな場合、そのアプリケーション・インスタンスは、コンピューティング・クラスタ中の別のリソースに移動される方がより適しているかもしれない。ステートフルなアプリケーションおよびそれへの依存対象の取り扱いについては後記でさらに説明することとする。

【００４９】

リソース割り当ての変更に対する要件の識別

図７は、本発明の諸態様による、複数のリソース種類の各々に対する例示的な計算メカニズム７００を表すブロック図を示す。全体のバーは、或る特定のアプリケーション・インスタンスに対する或る特定のリソースの現在の割り当て７０２（プラス／マイナス全ての累積差分）を表す。計算メカニズム７００に描かれた２つの水位線は、現在の割り当てに対する計算の２つのティアを画定する。高位の水位線７０６は、消費の高位ティア７０４の下限を画定する。或る特定のリソースに対し高位の水位線７０６が画定されていない場合、当該リソースに対しスケールアップ・メカニズム（リソース割り当ての垂直スケーリング）はアクティブ化されていない。

【００５０】

同様に、低位の水位線７０８は、消費の低位ティア７１０に対する上限を画定する。或る特定のリソースに対し低位の水位線７０８が画定されていない場合、当該リソースに対しスケールダウン・メカニズム（リソース割り当て解除の垂直スケーリング）はアクティブ化されていない。高位ティア７０４における持続的な消費は、特定のアプリケーション・インスタンスに対する特定のリソースの割り当ての増加をトリガすることになり、同様に、低位ティア７１０における持続的な消費は、特定のアプリケーション・インスタンスに対する特定のリソースの割り当ての低減をトリガすることになる。

【００５１】

持続的な消費を定義するために、割り当て変更に該当する期間が定められる。この期間とは、ティア（高位ティア７０４または低位ティア７１０）内で持続的な消費状態に留まっている、割り当て変更に該当する十分な数のサンプルを有する時間窓である。この期間は、時間とともにスライドして行く窓であり、デフォルト値を有してよい。範囲外のサンプルのパーセントが以下のようにさらに定義される。持続的な消費は、これらの定義に基づいて、画定された時間窓の持続時間の間に、高位ティア７０４もしくは低位ティア７１０を外れたサンプルが範囲外パーセント以下である消費として定義される。例えば、１時間の時間窓と１０％の範囲外パーセントとを仮定し、サンプルの少なくとも９０％がこの時間の最後まで、これらティア領域の１つ中にあった場合、関連リソースに対する適切な増加／低減処置が計算されることになる。

【００５２】

リソース割り当て増加の取り扱い

図８は、本発明の諸態様による、複数のリソース種類の各々に対する例示的な計算メカニズム７３０を表すさらなるブロック図を示す。具体的に、計算メカニズム７３０は、高位ティア７０４において識別された持続的な消費の例を表す。

【００５３】

これも同様に、全体のバーは、或る特定のアプリケーション・インスタンスに対する或る特定のリソースの現在の割り当て７０２を表す。高位ティア７０４は、検査された時間窓中の、全てのサンプルから範囲外パーセント以下のサンプルを除いたサンプルを含み、これにより、特定のリソースおよび特定のアプリケーション・インスタンスに対する増加オペレーションを生成するように示されている。この増加オペレーション（特定のアプリケーション・インスタンスに対する特定のリソースの追加割り当て）は、次に詳記するように、一定の増分で、または漸増する増分で、または適応増分で行うことが可能である。

【００５４】

増加ステップ７３２は、絶対値または現在の割り当て７０２のパーセントで定義される。増加関数の型がさらに定義され、これは以下の可能な値によって設定されてよい。１）１ステップ（デフォルト）、ここでは、割り当てが１ステップ分（例えば、増加ステップ７３２）増加されることになる。２）漸増する増加（例えば、漸増する増加７３４）、これは、増加オペレーションが相互に続いて行われる場合に適用される。または、３）自動的且つ適応的増加、これは、以下に詳記するように過去のデータに基づく。

【００５５】

増加オペレーションがない期間については、漸増増加７３４関数がリセットされる。漸増増加７３４の様々な関数を設定することが可能で、例えば、前回の増加オペレーション

での増加値＋１のステップを行うことができる（例えば、この値は、１ステップ、２ステップ、３ステップ、４ステップ、５ステップなどのパターンによる）。別の例では、前回の増加オペレーション＋リニアに漸増するステップを行うことも可能である（例えば、この値は、１ステップ、３ステップ、６ステップ、１０ステップ、１５ステップ、２１ステップなどのパターンによる）。

【００５６】

さらに別の例において、前回のオペレーションの増加値×２のステップを行ってもよい（例えば、この値は、１ステップ、２ステップ、４ステップ、８ステップ、１６ステップ、３２ステップなどのパターンによる）。ユーザが最大消費および関連コストを制御できるように、増加限度７３６がさらに定義される。いくつかの実施形態において、異なる期間（例えば、一日）にマップされる複数の増加限度があってよく、これは、リソースが割り当てられるもしくは使用される、またはその両方の時間の如何によってリソースのコストが変わる場合には有用であり得る。或るリソースおよび或るアプリケーション・インスタンスに対する増加オペレーションは、そのリソースおよびアプリケーション・インスタンスに対する増加限度７３６の定義値（または値群）を超えないことになる。リソースとアプリケーションとの各対（つい）は或る指示とさらに関連付けられ、これは、増加に対しクリティカル・リソース指示、と名付けることができよう。この増加に対しクリティカル・リソース指示が該当に設定され、リソースおよびアプリケーション・インスタンスに対する増加限度７３６に既に達しており、さらに高位ティア７０４での消費が行われている場合、適切な処置が取られることになり、この処置は、当該アプリケーション・インスタンス全体に対し定義され、アプリケーション・インスタンスの水平スケーリング（例えば、アプリケーション・インスタンスの追加）を含んでよい。

【００５７】

リソース割り当て低減の取り扱い

図９は、本発明の諸態様による、複数のリソース種類の各々に対する例示的な計算メカニズム７６０を表すさらなるブロック図を示す。具体的に、計算メカニズム７６０は、低位ティア７１０において識別された持続的な消費の例を表している。

【００５８】

さらにこれもまた、全体のバーは、或る特定のアプリケーション・インスタンスに対する或る特定のリソースの現在の割り当て７０２を表す。低位ティア７１０は、検査された時間窓中の、全てのサンプルから範囲外のパーセント以下のサンプルを除いたサンプルを含み、これにより、特定のリソースおよび特定のアプリケーション・インスタンスに対する低減オペレーションを生成するように示されている。この低減オペレーション（特定のアプリケーション・インスタンスからの特定のリソースの割り当て解除）は、次に詳記するように、一定の減分で、または最大減分で、または、漸増する減分で、または適応減分で行うことが可能である。

【００５９】

低減ステップ７６２は、絶対値または現在の割り当て７０２のパーセントで定義される。低減関数の型がさらに定義され、これは以下の可能な値によって設定されてよい。１）１ステップ（デフォルト）、ここでは、割り当てが１ステップ分（例えば、低減ステップ７６２）低減されることになる。２）最大低減、ここでは、割り当ての低減が、低減限度７６６を制限している上部のステップまで下げて適用されることになる。３）漸増する低減（例えば、漸増低減７６４）、これは低減オペレーションが相互に続いて行われる場合に適用される。または、４）自動的且つ適応的低減、これは、以下に詳記するように過去のデータに基づく。

【００６０】

低減オペレーションがない期間については、漸増低減７６４機能がリセットされる。漸増低減７６４の様々な関数を設定することが可能で、例えば、前回の低減オペレーションでの低減値＋１のステップを行うことができる（例えば、この値は、１ステップ、２ステップ、３ステップ、４ステップ、５ステップなどのパターンによる）。別の例では、前回

の低減オペレーション＋リニアに漸増するステップを行うことも可能である（例えば、この値は、１ステップ、３ステップ、６ステップ、１０ステップ、１５ステップ、２１ステップなどのパターンによる）。

【００６１】

さらに別の例において、前回のオペレーションの低減値×２のステップを行ってもよい（例えば、この値は、１ステップ、２ステップ、４ステップ、８ステップ、１６ステップ、３２ステップなどのパターンによる）。ユーザが当該アプリケーション・インスタンスに対する特定のリソースの最小割り当てを制御できるように、低減限度７６６がさらに定義される。いくつかの実施形態において、異なる期間（例えば、一日）にマップされる複数の低減限度があってよく、これは、リソースが割り当てられるもしくは使用される、またはその両方の時間の如何によってリソースのコストが変わる場合には有用であり得る。或るリソースおよび或るアプリケーション・インスタンスに対して計算される低減オペレーションは、そのリソースおよびアプリケーション・インスタンスに対する低減限度７６６の定義値（または値）を下回る割り当ては低減しないことになる。リソースとアプリケーションとの各対は或る指示とさらに関連付けられ、これは、低減に対するクリティカル・リソース指示、と名付けることができよう。この低減に対しクリティカル・リソース指示が該当に設定され、リソースおよびアプリケーション・インスタンスに対する低減限度７６６に既に達しており、さらに低位ティア７１０の消費が行われている場合、適切な処置が取られることになり、この処置は、当該アプリケーション・インスタンス全体に対し定義され、アプリケーション・インスタンスの水平スケーリング（例えば、アプリケーション・インスタンスの除去）を含んでよい。

【００６２】

過去のデータに基づく、自動的且つ適応的な増加および低減

図１０は、本発明の諸態様による、過去のデータに基づく、リソースの自動的且つ適応的増加および低減の方法８００を表す組み合せブロック／フローチャート図を示す。ステップ８０２で開始され、自動的且つ適応的であるように、コンピューティング・クラスタによって保持されている過去のデータ８０６に基づいて、増加または低減関数の型を設定することができる。この機能を実装するために、本明細書のメカニズムは、時間を通しての、各アプリケーションに対する各リソースの消費レベルについての過去の消費データ８０６を維持する（ステップ８０４）。或るアプリケーション・インスタンスに対して増加または低減オペレーションがトリガされると（ステップ８０８）、その特定のリソース、アプリケーション、および時間に対する過去の消費データ８０６（但し、過去の消費データ８０６中に維持され、考慮に入れられるさらなる属性があってよい）に基づいて、予測消費レベルが計算される（ステップ８１０）。次いで、これに基づき、当該アプリケーションに対するリソースの予測された消費レベルに対応するため、増加または低減オペレーションが計算される（ステップ８１２）。方法８００が終了する（ステップ８１４）。

【００６３】

アプリケーション・レベルのメカニズム

<u>増加方法</u>

図１１は、本発明の諸態様による、例示的な対角スケーリング増加オペレーションを表すブロック図９００を示す。これらオペレーションは、アプリケーションのインスタンス（９０２）の基本数で開始され、この基本数は、単一のアプリケーション・インスタンスであってもよい。アプリケーション・インスタンス９０２の各々は、随意的に垂直スケーリングによって定義され、各アプリケーション・インスタンスは、各独立的にスケーリング・オペレーションのため計算される。さらなるオプションは、その下で水平スケーリングだけが行われ、その上で、垂直スケーリングに続いて水平スケーリングが行われることになることになるいくつかのアプリケーション・インスタンスを定義することである。このアプリケーション・インスタンスの定義数がアプリケーション・インスタンスの基本数９０２に等しい場合、最初に垂直スケーリングが、その後に水平スケーリングが適用される。このシナリオは、例えば、（状態データを有する）ステートフルなアプリケーション

に対して有用であり得る。さらに、アプリケーション・インスタンスのこの定義数が無制限である場合、垂直スケーリングは実際上無用であり、水平スケーリング・オペレーションだけが行われることになる。このシナリオは、例えば、（状態データを持たない）ステートレスなアプリケーションに対して有用であり得る。但し、全ての場合において、前述の実施形態の機能には、スケーリング要件の自動追跡および計算が適用される。

【００６４】

これらのオペレーションは、アプリケーション・インスタンス群（９０２）の各アプリケーション・インスタンスのリソース消費を追跡し、各アプリケーション・インスタンスのリソース消費をそれへのリソース割り当てと比較する。アプリケーション・インスタンスの負荷が増大した場合、特定のアプリケーション・インスタンスに与えるリソースの割り当てに対する増加オペレーションを計算することができる。以下は、アプリケーション・インスタンスのリソース割り当てに対し計算された増加オペレーションの処理についてのいくつかのケース例である。

【００６５】

ケース１－リソースが利用可能である：ホスト上で（リソースの利用可能性およびアプリケーションの優先度を考慮に入れながら）垂直増加に対応が可能で、アプリケーション・インスタンスの増加限度７３６に達していない場合、そのアプリケーション・インスタンスには垂直スケーリングが適用され（ブロック９０４）、増加限度７３６の閾値を超えていないアプリケーション・インスタンスには、その後も引き続き垂直増加オペレーションが適用される（ブロック９０６）。（なお、アプリケーション・インスタンス群は、垂直増加オペレーションの過程で増大する円または楕円で表され、これにより各それぞれのアプリケーション・インスタンスに割り当てられたリソースの量を示している。）

【００６６】

ケース２－アプリケーション・インスタンスの増加限度７３６に達している：
アプリケーション・インスタンスのリソースに対する増加限度７３６に達しており、そのリソースが増加に対しクリティカル・リソースに設定されている場合（ブロック９０８）、そのアプリケーション・インスタンスは、当該アプリケーションの生成されたアプリケーション・インスタンス９１２の所定数でさらに水平にスケールされてよい（ブロック９１０）。これら追加のアプリケーション・インスタンスは、飽和状態のインスタンスの特定のリソースの現行の割り当てを継承してもよく、またはそのリソースの新規の基本割り当てを受けてもよい。

【００６７】

ケース３－ホストの増加限度に達した：ホスト上の当該リソースが（アプリケーションの優先度を合わせ考慮した上で）使い尽くされている場合、以下のように、１つ以上の事前定義されたオプションを行うことが可能である。

【００６８】

第一オプションは、ホスト上に（生成されたアプリケーション・インスタンス９１２に示されるような）アプリケーションの追加のインスタンスを生成することによって、前述したような水平スケーリングを行うことを含む。このオプションは、他のまたは代わりのホスト上に追加のアプリケーション・インスタンス９１２を生成するまたは利用可能にするための状態データが必要とされないので、ステートレス・アプリケーションに適している。

【００６９】

第二オプションは、第一オプションに加えてまたは換えて行うことができ、他のアプリケーション・インスタンスからリソースの割り当てを取得するよう試みることを含む。このシナリオでは、最低の優先度、および最小の負荷のアプリケーション・インスタンスが計算され、それらのアプリケーション・インスタンスにそのリソースの割り当てを手放す要求が送信される。それらのアプリケーション・インスタンスがリソースの割り当てを、リソースを必要とするアプリケーション・インスタンスに手放しても、十分なリソースの割り当てを準備することができない場合は、リソースを必要性とする当該アプリケーショ

ン・インスタンスは、割り当てに利用可能なリソースがある別のホストに移動することが可能である。さらに、この移動が可能でない場合、前に詳記したように、他のホスト上に当該アプリケーションの追加のアプリケーション・インスタンスを生成することによって、水平スケーリングを引き続いて行うことができる。この移動オプションは、（状態データを有する）ステートフル・アプリケーションに適している。ステートフル・アプリケーションに対しては、かかるアプリケーションがそれらの状態に関し、分散／同期化をサポートしていないことがあるので、水平スケーリングよりも移動の方が望ましい。また、アプリケーション・データもアプリケーション・インスタンスと共に移動することが可能である。

【００７０】

図１２は、本発明の諸態様による、前のケースのシナリオで説明したような、アプリケーション・インスタンスのリソース割り当てに対する、計算された増加オペレーションを処理する方法１０００を表すフローチャート図を示す。方法１０００は、或るアプリケーション・インスタンスのリソース割り当てに対して計算された増加オペレーションを受信すること（ステップ１００４）によって開始され（ステップ１００２）、このオペレーションは、当該アプリケーション・インスタンスの作業負荷によって必要となった１つ以上の特定のリソース要求のため実行されなければならない。

【００７１】

ステップ１００６で、当該ホスト上で垂直増加（追加リソースの割り当て）に対応可能かどうか、また、そのアプリケーション・インスタンスに対する増加限度７３６に達しているかどうかに関して判断が行われる。ホスト上で垂直増加に対応が可能であり、アプリケーション・インスタンスに対する増加限度７３６に達していない場合、方法１０００は、そのアプリケーション・インスタンスに追加のリソース（１つまたは複数）を割り当てることによって、該アプリケーション・インスタンスに垂直増加オペレーションを適用するステップ（ステップ１０１４）に進み、そこで方法１０００は終了する（ステップ１０２６）。

【００７２】

ステップ１００６に戻って、ホスト上で垂直増加に対応が可能でなく、もしくはアプリケーション・インスタンスに対する増加限度７３６に達している、またはその両方の場合、アプリケーション・インスタンスに対する増加限度７３６に本当に達しているかどうかさらなる判断が行われ、達していれば、そのリソースが、増加に対しクリティカル・リソースとして設定されているかどうかが判断される（ステップ１００８）。当該アプリケーション・インスタンスに対する増加限度７３６に達しており、且つリソースが増加に対しクリティカル・リソースとして設定されている場合、方法１０００は、所定数の追加アプリケーション・インスタンスを追加または生成することによって、アプリケーション・インスタンスを水平にスケールするステップ（ステップ１０１６）に進み、そこで方法１０００は終了する（ステップ１０２６）。

【００７３】

ステップ１００８に戻って、当該アプリケーション・インスタンスに対する増加限度７３６にまだ達していない、またはリソースが増加に対しクリティカル・リソースとして設定されていない場合、諸アプリケーションの優先度を考慮に入れながら、ホスト上のリソースが使い尽くされているかどうかについてさらなる判断が行われる（ステップ１０１０）。ステップ１０１０で、諸アプリケーションの優先度を考慮に入れてもホスト上のリソースが使い尽くされていない場合、方法１０００は終了する（ステップ１０２６）。

【００７４】

ステップ１０１０に戻って、アプリケーションの優先度を考慮に入れて、ホスト上のリソースが使い尽くされている場合、当該アプリケーションがステートレスかどうかに関してさらなる判断が行われる（ステップ１０１２）。ステップ１０１２で、当該アプリケーションがステートレスであると判断された場合、方法１０００は、対象作業負荷を取り扱うため、所定数の追加アプリケーション・インスタンスを生成または追加することによっ

てアプリケーション・インスタンスを水平にスケールすることを継続し（ステップ１０１６）、方法１０００は終了する（ステップ１０２６）。

【００７５】

ステップ１０１２に戻って、アプリケーションがステートフルであると判断された場合、方法１０００は、ホスト上の、最低の優先度および最小の負荷のアプリケーション・インスタンスを計算し、これら（１つ以上の）インスタンスに、リソースを要求しているアプリケーション・インスタンスに必要なリソース割り当てを手放す要求を送信することを継続する（ステップ１０１８）。このステップが完了すると、アプリケーション・インスタンスのリソース要求が、リソースを必要とするアプリケーション・インスタンスへのリソースの再割り当てによって満たされたかどうかについての判断が行われる（ステップ１０２０）。ステップ１０２０で、アプリケーション・インスタンスのリソース要求が満たされた場合、方法１０００は終了する（ステップ１０２６）。

【００７６】

ステップ１０２０に戻って、アプリケーション・インスタンスのリソース要求が、リソースを必要とするアプリケーション・インスタンスへのリソースの再割り当てによって満たされなかった場合、もしくは他の（代わりの）アプリケーション・インスタンス（１つまたは複数）によって、リソースを必要としているアプリケーション・インスタンスへのリソースの再割り当てが全くできなかった場合、またはその両方の場合、リソースを必要としているアプリケーション・インスタンスを、割り当てのためのリソースが利用可能な別のホストへ移動する試みが行われる（ステップ１０２２）。最後に、ステップ１０２４で、この移動が可能であったかどうかの判断が行われ、そうであれば、成功だったかどうか判断される。移動が可能でない、もしくは成功しなかった、またはその両方の場合、方法１０００は、所定数の生成アプリケーション・インスタンスによるアプリケーション・インスタンスの水平スケーリングに戻り（これは当該基本アプリケーション・インスタンスと同じホスト上でも他のホスト上でもよい）、そこで方法１０００は終了する（ステップ１０２６）。上記以外に、ステップ１０２４で、移動が可能であり実際に成功した場合も本方法は終了する（ステップ１０２６）。

【００７７】

上記で説明したシナリオをモデルし、さらにそれをユーザ設定のポリシーに組み込むため、垂直増加限度に達した、またはリソースが使い尽くされた場合の処置が定義される。この処置は、増加に対しクリティカルな少なくとも１つのリソースが当該アプリケーション・インスタンスに対する増加限度７３６に達したか、またはそのアプリケーション・インスタンスを実行しているホスト上でそのリソースが使い尽くされたとき行われ、該特定のアプリケーション・インスタンスに対し高位ティア７０４の消費の検出は継続される。この処置は、以下のオプションによって、ユーザにより設定されてよい。１）ユーザに通知する、２）他のアプリケーション・インスタンスから、必要なリソースのリソース割り当てを取得することを試みる、３）当該アプリケーション・インスタンスを（他のまたは代わりのホスト（１つまたは複数）に）移動する、または、４）水平増加をおこなう。ユーザがオプション４（水平増加オペレーションを行う）を指定する場合、そのユーザは、生成対象のいくつかの追加のアプリケーション・インスタンス、および、それら追加のアプリケーション・インスタンスにどのリソース割り当て（どのリソースのどの量）を適用するかをさらに設定することができる。この指定されたリソース割り当ては、現在のアプリケーション・インスタンスの現在の割り当てを含んでもよく、または各指定されたリソースの新規の基本割り当てを受けるために、新しく生成されたアプリケーション・インスタンスが構成されてもよい。

【００７８】

<u>低減方法</u>

この低減方法の目的は、過剰なリソースを、それらリソースが（例えば、リソース（１つまたは複数）を必要としている他のアプリケーション・インスタンスのために）再利用されるように、アプリケーション・インスタンスから解放することである。この低減への

一般的アプローチは、最初に垂直低減オペレーションを適用し、その後に水平低減オペレーションを適用することとして定義することができ、この垂直低減は、前述したリソース消費の自動追跡技法に従ってリソース毎に適用される。

【００７９】

水平低減オペレーションを行うために、遊休状態のアプリケーション・インスタンスは、以下の２つの可能な定義によって定義される。

１．システム定義の遊休状態インスタンス、ここでは、低減に対しクリティカルな全てのリソースがそれらの低減限度７６６に達しており、それらの当該消費レベルが、それぞれの低位ティア７１０において、水平低減オペレーションに該当する期間に亘って検出されている。

２．ユーザ定義の遊休状態インスタンス、ここでは、或る特定のアプリケーション・インスタンスが遊休状態かどうかを判断するためのユーザ規定の手順／実行可能プログラムを実行するためのインターフェースが定義される。この種のユーザ・ロジックは、一般に、アプリケーション・インスタンスの負荷をチェックするのに効果的である。

【００８０】

前述のこれら２つの技法は、相互に合わって機能することが可能である。例えば、システム定義に基づいて、或る遊休状態アプリケーション・インスタンスが識別されたとき、そのアプリケーション・インスタンスの状態を検証するために、ユーザ規定の手順を加えて作動することが可能である。しかして、或る遊休状態アプリケーション・インスタンスが識別されたとき、前述の技法に従って、その遊休状態インスタンスを終了させることができ、その割り当てリソースを他のアプリケーション・インスタンスの再利用のために解放することが可能である。さらに、ユーザは、アプリケーションに対し保持されるべき最小数のインスタンスを加えて指定することができ、アプリケーション・インスタンスのかかる最小数が設定されている場合、遊休状態アプリケーション・インスタンスは、当該アプリケーションのインスタンスの現在数がそのアプリケーションに対し設定されたインスタンスの最小数よりも高い場合にだけ、終了されることになる。

【００８１】

図１３は、本発明の諸態様による、例示的な対角スケーリング低減オペレーションを表すブロック図１１００を示す。これらオペレーションは、アプリケーションのインスタンスの現在数および形成（アプリケーション・インスタンスの現在数１１０２）で開始される。すなわち、本オペレーションは、アプリケーション・インスタンスの現在数１１０２および各それぞれのインスタンスに割り当てられたリソースの量を算定することによって開始される（図１１００においても同様に、楕円形がどの位大きいかで各アプリケーション・インスタンスの大きさを表して示されている）。現在のアプリケーション・インスタンス１１０２の各々のリソース消費がモニタおよび追跡され、最後にアプリケーション・インスタンスへのリソース割り当てと比較される（これも同様に、各それぞれのアプリケーション・インスタンスに対し、各リソースが個別にモニタおよび追跡されることに留意する）。或る特定のアプリケーション・インスタンスの負荷が軽減されている場合、そのアプリケーション・インスタンスに割り当て配属されているリソースの量を低減するため、垂直低減オペレーションが計算されてよい（ブロック１１０４）。さらに、前述した方法を用いて遊休状態アプリケーション・インスタンスを識別することができ（遊休状態インスタンス１１０６）、同時に、残りのアプリケーション・インスタンスは活動状態として識別することが可能である（活動状態インスタンス１１０８）。遊休状態アプリケーション・インスタンス１１０６のこの判定に基づいて、それら識別された遊休状態アプリケーション・インスタンス１１０６の一部または全部を終了することができる。すなわち、遊休状態アプリケーション・インスタンス１１０６を低減または終了するために、水平低減オペレーションが行われる（ブロック１１１０）。

【００８２】

ユーザが、アプリケーション・インスタンスを遊休状態として判定するための上記の技法を設定することを可能にするために、水平低減処置の指示が定義される。この指示が或

るアプリケーション・インスタンスに該当として設定された場合、且つ、その特定のアプリケーション・インスタンスに対して、低減に対しクリティカルな全てのリソースがそれらの低減限度７６６に達しており、該特定のアプリケーション・インスタンスのための消費レベルが、低位ティア７１０において水平低減に該当する期間の間持続していることが検出された場合、水平低減処置が取られることになる。

【００８３】

水平低減処置は、１）ユーザに通知する、または、２）当該アプリケーション・インスタンスを（静々とまたは強制的に）終了する、オプションのうちからユーザによって設定することが可能である。水平低減に対する期間がユーザによってさらに定義され、これは、或るアプリケーション・インスタンスが遊休状態として判定されるためにその遊休状態が満たすべき最小の持続時間を含む。

【００８４】

クラスタ・レベルのメカニズム

対角スケーリング・アルゴリズム

図１４は、本発明の諸態様による、ハイレベルでの対角スケーリング・アルゴリズムを表すフローチャート図１２００を示す。図１２００を検討すると、本対角スケーリング・アルゴリズムの達成目的は、諸アプリケーションに入力として供給される実際の作業負荷、アプリケーションの優先度、および利用可能なリソースに従って、コンピューティング・クラスタのアプリケーションのスループットを最大化することと、これらのリソースのコストを最小化することとの両方である。これらの目的を達成するために、本対角スケーリング・アルゴリズムは、アプリケーション・インスタンス毎に諸特定リソースの実際の消費に基づいてリソース割り当ての必要な修正を計算し、引き続いて、アプリケーションの優先度および利用可能なリソースに従って、実際のオペレーションを計算する。

【００８５】

本対角スケーリング・アルゴリズムは、リソース要件を計算する第一フェーズで開始される（フェーズ１２０２）。このフェーズは、２つのステップを含み、その第一（ステップ１２０４）は、個別のアプリケーション・インスタンスの個別のリソースのレベルに焦点を合わせる。ステップ１２０４で、本アルゴリズムは、アプリケーション・インスタンスの各個別のリソースに対し、前述した方法を用いてリソースの割り当ての変更が必要かどうかを判断する。フェーズ１２０２の第二ステップ（ステップ１２０６）は、アプリケーション・インスタンス・レベルに焦点を合わせる。ステップ１２０６で、本アルゴリズムは、前述した方法を用いて各アプリケーション・インスタンスに対するオペレーションの要件を計算する。これらの計算されたオペレーションの要件は、垂直および水平オペレーションの両方を含み、ここでも同様に、垂直スケーリング・オペレーションは、アプリケーション・インスタンス毎の個別のリソースに対する割り当てを増加および低減し、水平スケーリング・オペレーションは、アプリケーションのインスタンスを分割または折り畳む。アプリケーション・インスタンスの分割とは、追加のアプリケーション・インスタンスを生成することを言い、アプリケーション・インスタンスの折り畳みとはアプリケーション・インスタンスを除去することを言う。

【００８６】

本対角スケーリング・アルゴリズムは、次いで、前のステップで計算された要件を処理する第二フェーズ（フェーズ１２０８）に進む。フェーズ１２０８は、２つのステップを含む。フェーズ１２０８中の第一ステップ（ステップ１２１０）は、アプリケーション・レベルに焦点を合わせる。すなわち、ステップ１２１０で、本アルゴリズムは、前述した方法を用いてステップ１２０６で計算された垂直低減および水平折り畳みオペレーションを処理する。フェーズ１２０８中の第二ステップ（ステップ１２１２）は、クラスタ・レベルに焦点を合わせる。しかして、ステップ１２１２で、本アルゴリズムは、諸アプリケーションに対する優先度を計算するかまたは取得し、それらアプリケーションの優先度に基づいて（例えば、優先度キューを使って）垂直増加および水平分割オペレーションを順序付ける。優先度付けおよび順序付けについては以下でさらに説明することとする。さら

に、ステップ１２１２で、本アルゴリズムは、次いでこの順序付けに基づいて、図１１～１３の説明と一致する垂直増加および水平分割オペレーションを処理する。

【００８７】

スケーリング・オペレーションに対するアプリケーションの優先度の計算

図１５は、本発明の諸態様による、アプリケーションの優先度を計算するための、例示的なアプリケーションのトポロジを表すブロック図１３００を示す。図１３００に示されたアプリケーションのトポロジを考察すると、４つのアプリケーション、すなわち、アプリケーション（１）１３０２、アプリケーション（２）１３０４、アプリケーション（３）１３０６、およびアプリケーション（４）１３０８が示されている。アプリケーション１３０２～１３０８の各々に対し、その中に、アプリケーション１３０２～１３０８のそれぞれの重要さ（もしくは重要性）またはそこで行われる機能として、Ｓ［アプリケーション］が定義されている。すなわち、アプリケーション（１）１３０２の重要さＳ［アプリケーション１］は、アプリケーション（２）１３０４の重要さＳ［アプリケーション２］等々よりも高い（アプリケーション自体もしくはアプリケーション（１）１３０２によって行われる機能の）重要さを有し得る。このアプリケーションの重要さに対する尺度は、５つの典型的なレベルを含むが、但し、これらのレベルは実装の如何による必要に応じて修改されてよい。図１３００において、アプリケーション１３０２～１３０８の各々は、そのそれぞれのＳ［アプリケーション］の値に関連付けられる。

【００８８】

図１３００中のアプリケーション（１）１３０２を考察してみよう。アプリケーション（１）１３０２に対し、図１３００は、アプリケーションの依存性の一例を示し、この例では、アプリケーション（２）１３０４、アプリケーション（３）１３０６、およびアプリケーション（４）１３０８は、各々、アプリケーション（１）１３０２への依存性を保持し、さらに、アプリケーション（１）１３０２も他のアプリケーション（図示せず）への依存性を保持する。一般性を失うことなく、この例では、アプリケーション（１）１３０２への全ての依存性が示されていると仮定すれば、各々の依存性、Ｓ［依存性］（またはＳ［ｄｐｎ］）が、先行のアプリケーションへの依存アプリケーションの依存性の重要さ（または重要性）として定義される。

【００８９】

図１３００にさらに示されているように、アプリケーション（２）１３０４、アプリケーション（３）１３０６、アプリケーション（４）１３０８およびアプリケーション（１）１３０２との間の依存性の各々は、それぞれのＳ［ｄｐｎ］値に関連付けられる。重要さの２つの型の間の差異を説明するために、以下の２つの例を考えてみよう。（１）一方のアプリケーションが他方の使用（または他方への依存性）を有するが、その利用は低い重要さの機能に対するものである、２つの重要なアプリケーション、および（２）一方のアプリケーションが他方を使用し、この使用は依存アプリケーションの主要機能を促進し、したがってこの依存性に対し高い重要さの値を有する、２つの低い重要さのアプリケーション。

【００９０】

ユーザが、各アプリケーションおよびその依存性に対する重要さの値を規定する。本対角スケーリング・アルゴリズムは、これらの重要さの値を標準化し、アプリケーションに対する優先度を計算し、また、依存関係のトポロジが変更されたとき、またはアプリケーションがクラスタに追加されるかまたはクラスタから除去されたときに優先度を再計算する。以下は、本アルゴリズムが、或るアプリケーションＸの優先度を計算するための算式である。

【００９１】

【数１】

$$Priority[application\ X]=$$

A)

$$W\times S[applicaiton\ X]+$$

B)

$$(1-W)\times\frac{\sum_{[application\ Y\in applications\ dependent\ on\ application\ X]}[S[dependency\ Y\rightarrow X]\times S[application\ Y]]}{[Divisor]}$$

【００９２】

この算式において、第一要素（Ａ）はアプリケーションの重要さをモデルし、第二要素（Ｂ）はアプリケーションへの依存性の重要さをモデルしている。Ｗは、アプリケーションの重要さ対そのアプリケーションへの依存性の重要さの相対ウェイトである。Ｗの値に対する範囲の一例は０～１であってよく、Ｗはデフォルト値を有してよい。また、重要さの値、Ｓ［アプリケーションまたは依存性］も０～１の範囲の値に標準化することが可能である。

【００９３】

より明瞭にするため、また上記の算式（要素ＡおよびＢ）中に示されるように、アプリケーションの優先度は、１）アプリケーションの重要さと第一ウェイトとの積として第一要素を計算し、２）当該アプリケーションへの依存性の重要さと依存アプリケーションの重要さとの積を集約し、その集約を或る除数で除して、その結果に第二ウェイトを乗じて第二要素を計算し、さらに３）その第一要素と第二要素とを加算することによって計算することができる。

【００９４】

第二要素中の除数はいくつかの仕方で定義することができる。以下は、この除数を計算するための例示の実施形態である。

【００９５】

【数２】

$$Divisor=Total\ number\ of\ applications-1$$

$$Divisor=\underset{[application\ I\in all\ applications]}{MAX}\left[\sum_{[application\ J\in applications\ dependent\ on\ application\ I]}1\right]$$

$$Divisor=\underset{[application\ I\in all\ applications]}{MAX}\left[\sum_{[application\ J\in applications\ dependent\ on\ application\ I]}S[application\ J]\right]$$

$$Divisor=\underset{[application\ I\in all\ applications]}{MAX}\left[\sum_{[application\ J\in applications\ dependent\ on\ application\ I]}S[dependency\ J\rightarrow I]\times S[application\ J]\right]$$

【００９６】

すなわち、この除数は、１）第一例のように、アプリケーションの合計数引く１、２）全てのアプリケーションの間からの所与のアプリケーションへの依存性の最大数、３）全てのアプリケーションの間での、或るアプリケーションに依存する諸アプリケーションの重要さの値の最大総和、もしくは、４）全てのアプリケーションの間での、依存性の重要さの値を備えた或るアプリケーションに依存する諸アプリケーションの重要さの値の積の最大総和、またはこれらの組み合せとして計算することが可能である。

【００９７】

なお、重要さの値およびＷが両方とも０～１の範囲内の場合、アプリケーションの優先度を計算するため上記に規定された算式は、所与のアプリケーションに対し０～１の範囲内の値を生成する。

【００９８】

この所与のアルゴリズムは、以下の弁別的な特徴を有する。（１）本アルゴリズムは、

各依存性の重要さを考慮する（すなわち入力として使用する）が、他方、既存の方法は、一般に、依存性それ自体にはいかなる重要さの値も関連付けない。（２）本アルゴリズムは、アプリケーションの重要さと、そのそれぞれの依存性の重要さとを組み合せるが、他方、既存の方法は、通常、ランク対象の要素を、固有のまたはユーザの知覚する重要さと関連付けない。および（３）本アルゴリズムは、アプリケーションの優先度の非反復的な計算を可能にし、しかして、再計算が必要なのは、アプリケーションとそれらの依存性のトポロジ変更、または重要さの値の変更があったときにだけである。既存の方法は、入力が異なり、トポロジの型とスケールとが異なるので、通常、反復的な計算を使用する。しかして、本提供のアルゴリズムは、効率的な非反復の計算を行うことを可能にするトポロジとスケールとを扱う。

【００９９】

例示のシステム実施形態

図１６は、本発明の諸態様による、分散コンピューティング環境における作業負荷の自動対角スケーリングに対するシステム実施形態を表す組み合せブロック／フローチャート図１４００を示す。

【０１００】

本システムへの入力は、アプリケーション明細（ブロック１４０２）およびアプリケーション・モニタリング情報（ブロック１４０４）である。スケーリング計算コンポーネント（ブロック１４０６）は、ブロック１４０４のモニタリング情報と組み合せて、アプリケーション明細および優先度を使って、前述のようにスケーリング・オペレーションを計算し、これらは同時的に行われてよい。次いで、スケーリング計算コンポーネント１４０６は、計算されたスケーリング・オペレーションをタスクとしてオペレーション実行キューに加える。

【０１０１】

２つのキューが定義され、１つのキューは並列オペレーション（ブロック１４０８）（すなわち、リソースの放出オペレーション）のためのものであり、第二のキューは優先オペレーション（ブロック１４１０）（すなわち、リソースの割り当てオペレーション）である。スケーリング・オペレーション・コンポーネント（ブロック１４１２）は、オペレーション実行キュー１４０８および１４１０からタスクを取得し、それらタスクを実行する。また、スケーリング・オペレーション・コンポーネント１４１２は、以下の任意の組み合せを実行することができる。（１）システム・スケジューラまたはリソース・マネージャを用いて、適切なリソース割り当ておよび放出オペレーションを計算し、適用する（ブロック１４１４）、（２）特定のホスト上で実行されている諸アプリケーション・インスタンスに対するリソースの消費限度を設定または修改する（ブロック１４１６）、（３）アプリケーション・インスタンスの設定を、それらアプリケーション・インスタンスに対し利用可能な更新されたリソースに合わせ調整する（例えば、アプリケーション・インスタンス内のスレッドの数を増加／低減する）（ブロック１４１８）、もしくは、（４）アプリケーション・インスタンスを生成および除去する（ブロック１４２０）、またはこれらの組み合せを実行する。なお、このスケーリング・オペレーション・コンポーネントは、分散されたコンポーネント、または分散されたメカニズムであってよく、独立したオペレーションを同時に適用してもよい。

【０１０２】

本発明は、システム、方法、もしくはコンピュータ・プログラム製品またはこれらの組み合せとすることができる。このコンピュータ・プログラム製品は、プロセッサに本発明の態様を実行させるためのコンピュータ可読プログラムを有するコンピュータ可読ストレージ媒体（または媒体群）を含むことが可能である。

【０１０３】

このコンピュータ可読ストレージ媒体は、命令実行デバイスが使用するための命令を保持し格納できる有形のデバイスとすることができる。このコンピュータ可読ストレージ媒体は、例えば、以下に限らないが、電子ストレージ・デバイス、磁気ストレージ・デバイ

ス、光ストレージ・デバイス、電磁気ストレージ・デバイス、半導体ストレージ・デバイス、または前述のデバイスの任意の適切な組み合せであってよい。コンピュータ可読ストレージ媒体のさらに具体的な例の非包括的リストには、携帯型コンピュータ・ディスケット、ハード・ディスク、ランダム・アクセス・メモリ（ＲＡＭ）、読み取り専用メモリ（ＲＯＭ）、消去およびプログラム可能読み取り専用メモリ（ＥＰＲＯＭ：ｅｒａｓａｂｌｅ ｐｒｏｇｒａｍｍａｂｌｅ ｒｅａｄ－ｏｎｌｙ ｍｅｍｏｒｙまたはフラッシュ・メモリ）、静的ランダム・アクセス・メモリ（ＳＲＡＭ：ｓｔａｔｉｃ ｒａｎｄｏｍ ａｃｃｅｓｓ ｍｅｍｏｒｙ）、携帯型コンパクト・ディスク読み取り専用メモリ（ＣＤ－ＲＯＭ：ｃｏｍｐａｃｔ ｄｉｓｃ ｒｅａｄ－ｏｎｌｙ ｍｅｍｏｒｙ）、デジタル多用途ディスク（ＤＶＤ：ｄｉｇｉｔａｌ ｖｅｒｓａｔｉｌｅ ｄｉｓｋ）、メモリ・スティック、フレキシブル・ディスク、パンチカードまたは記録された命令を有する溝中の嵩上げ構造体などの機械的符号化デバイス、および前述の任意の適切な組み合せが含まれる。本明細書で用いられるコンピュータ可読ストレージ媒体は、無線波または他の自由に伝播する電磁波、ウェーブガイドまたは他の送信媒体（例えば、光ファイバを通過する光パルス）、またはワイヤを通って送信される電気信号など、本質的に一時的な信号であると解釈されるものではない。

【０１０４】

本明細書に述べられたコンピュータ可読プログラム命令は、コンピュータ可読ストレージ媒体から、それぞれのコンピューティング／処理デバイスに、または、例えばインターネット、ローカル・エリア・ネットワーク、広域ネットワークもしくはワイヤレス・ネットワークまたはこれらの組み合せなどのネットワークを介して、外部のコンピュータもしくは外部のストレージ・デバイスにダウンロードすることが可能である。このネットワークには、銅送信ケーブル、光送信ファイバ、ワイヤレス通信、ルータ、ファイアウォール、スイッチ、ゲートウェイ・コンピュータ、もしくはエッジ・サーバまたはこれらの組み合せが含まれてよい。それぞれのコンピューティング／処理デバイス中のネットワーク・アダプタ・カードまたはネットワーク・インターフェースは、ネットワークからコンピュータ可読プログラム命令を受信し、そのコンピュータ可読プログラム命令を、ストレージのため、それぞれのコンピューティング／処理デバイス内のコンピュータ可読ストレージ媒体の中に転送する。

【０１０５】

本発明のオペレーションを実行するためのコンピュータ可読プログラム命令は、アセンブラ命令、命令集合アーキテクチャ（ＩＳＡ：ｉｎｓｔｒｕｃｔｉｏｎ－ｓｅｔ－ａｒｃｈｉｔｅｃｔｕｒｅ）命令、マシン命令、マシン依存命令、マイクロコード、ファームウェア命令、状態設定データ、または、Ｓｍａｌｌｔａｌｋ、Ｃ＋＋などのオブジェクト指向プログラミング言語、および「Ｃ」プログラミング言語もしくは類似のプログラミング言語などの従来式の手続き型プログラミング言語を含む、１つ以上のプログラミング言語の任意の組み合せで記述されたソース・コードもしくはオブジェクト・コードであってよい。このコンピュータ可読プログラム命令は、スタンドアロン・ソフトウェア・パッケージとしてユーザのコンピュータで専ら実行することも、ユーザのコンピュータで部分的に実行することもでき、一部をユーザのコンピュータで一部を遠隔コンピュータで実行することもでき、あるいは遠隔のコンピュータまたはサーバで専ら実行することもできる。後者の場合は、ローカル・エリア・ネットワーク（ＬＡＮ：ｌｏｃａｌ ａｒｅａ ｎｅｔｗｏｒｋ）または広域ネットワーク（ＷＡＮ：ｗｉｄｅ ａｒｅａ ｎｅｔｗｏｒｋ）を含む任意の種類のネットワークを介して、遠隔コンピュータをユーザのコンピュータに接続することもでき、あるいは（例えばインターネット・サービス・プロバイダを使いインターネットを介し）外部のコンピュータへの接続を行うことも可能である。いくつかの実施形態において、例えば、プログラム可能論理回路、フィールドプログラム可能ゲート・アレイ（ＦＰＧＡ：ｆｉｅｌｄ－ｐｒｏｇｒａｍｍａｂｌｅ ｇａｔｅ ａｒｒａｙ）、またはプログラム可能論理アレイ（ＰＬＡ：ｐｒｏｇｒａｍｍａｂｌｅ ｌｏｇｉｃ ａｒｒａｙ）を含む電子回路は、本発明の諸態様を実行すべく、該電子回路をカスタマイズするため

コンピュータ可読プログラム命令の状態情報を利用することによって、該コンピュータ可読プログラム命令を実行することができる。

【０１０６】

本発明の諸態様は、本発明の諸実施形態による方法、装置（システム）、およびコンピュータ・プログラム製品のフローチャート図もしくはブロック図またはその両方を参照しながら本明細書で説明されている。当然のことながら、フローチャート図もしくはブロック図またはその両方の各ブロック、およびフローチャート図もしくはブロック図またはその両方のブロックの組み合せは、コンピュータ可読プログラム命令によって実装することが可能である。

【０１０７】

これらのコンピュータ可読プログラム命令を、汎用コンピュータ、特殊用途コンピュータ、またはマシンを形成する他のプログラム可能データ処理装置のプロセッサに提供し、そのコンピュータまたは他のプログラム可能データ処理装置のプロセッサを介して実行されるこれらの命令が、フローチャートもしくはブロック図またはその両方のブロックもしくはブロック群中に特定されている機能群／動作群を実装するための手段を生成するようにすることができる。また、コンピュータ、プログラム可能データ処理装置、もしくは他のデバイスまたはこれらの組み合せに対し特定の仕方で機能するよう命令することが可能なこれらのコンピュータ可読プログラム命令を、コンピュータ可読ストレージ媒体に格納し、格納された命令を有するコンピュータ可読ストレージ媒体が、フローチャートもしくはブロック図またはその両方のブロックまたはブロック群中に特定されている機能／動作の諸態様を実装する命令群を包含する製造品を構成するようにすることができる。

【０１０８】

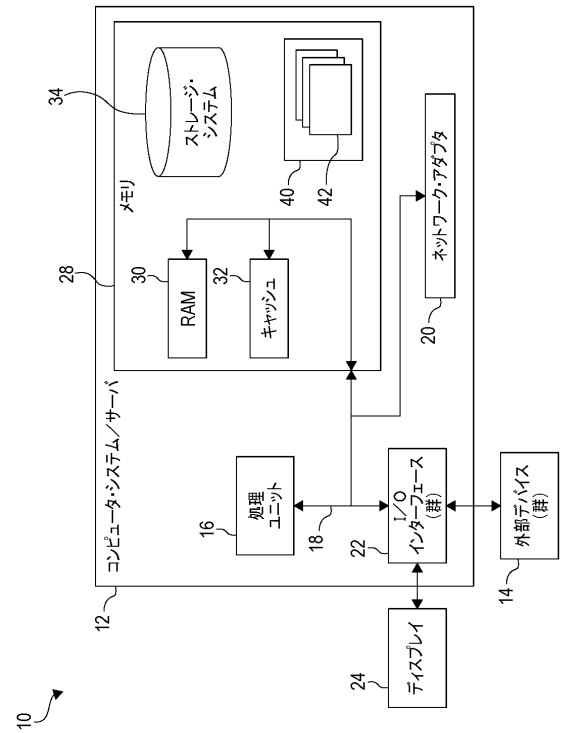
さらに、これらコンピュータ可読プログラム命令を、コンピュータ、他のプログラム可能データ処理装置、または他のデバイスにロードし、そのコンピュータ上で、他のプログラム可能装置上で、または他のデバイス上で一連のオペレーション・ステップを実施させて、コンピュータ実装のプロセスを作り出し、当該コンピュータ上で、他のプログラム可能装置上でもしくは他のデバイス上で実行される命令が、フローチャートもしくはブロック図またはその両方のブロックもしくはブロック群中に特定されている機能群／動作群を実装するようにすることも可能である。

【０１０９】

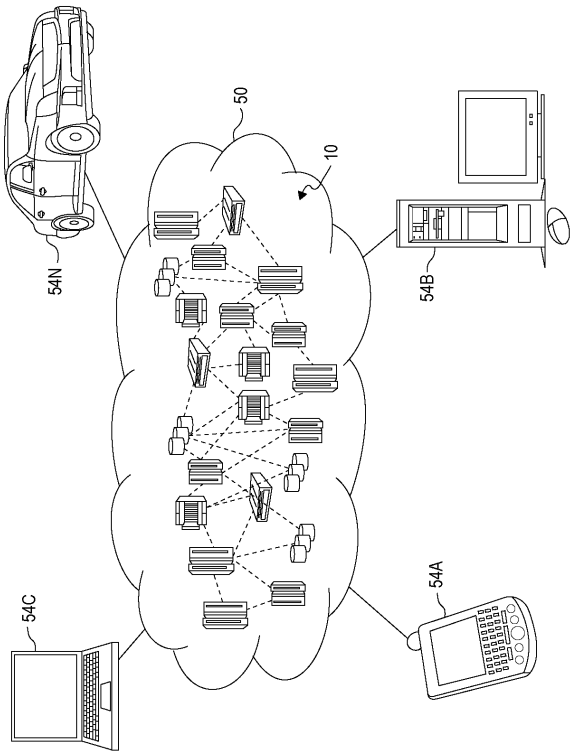
諸図面中のフローチャートおよびブロック図は、本発明の様々な実施形態による、システム、方法、およびコンピュータ・プログラム製品から可能となる実装のアーキテクチャ、機能、およびオペレーションを表している。この点に関し、フローチャートまたはブロック図中の各ブロックは、特定の論理機能（１つまたは複数）を実装するための一つ以上の実行可能命令を含む、モジュール、セグメント、または命令の部分を表し得る。いくつかの別の実装においては、ブロック中に記載された機能が、図面に記載された順序から外れて行われてよい。例えば、連続して示された２つのブロックが、関与する機能に応じ、実際にはほぼ同時に実行されることがあり、時にはこれらのブロックが逆の順序で実行されることもあり得る。さらに、ブロック図もしくはフローチャート図またはその両方の各ブロック、およびブロック図もしくはフローチャート図またはその両方中のブロック群の組み合せは、特定の機能または動作を実施する特殊用途ハードウェア・ベースのシステムによって実装でき、または特殊用途ハードウェアとコンピュータ命令との組み合せによって実行できることにも留意すべきである。

【図面】

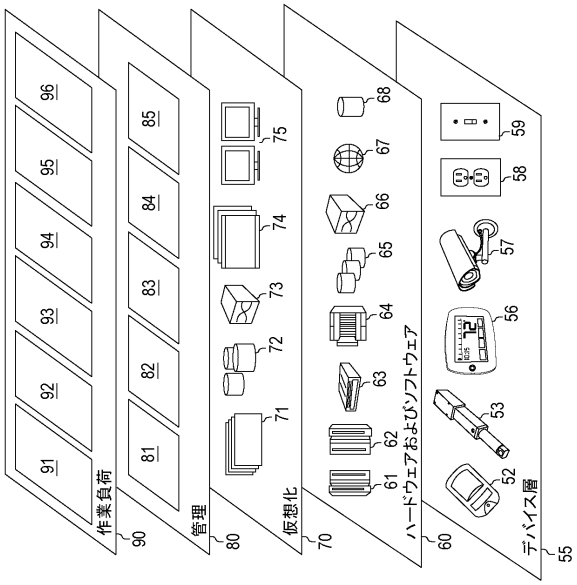
【図１】


ストレージ・システム
メモリ
RAM
キャッシュ
ネットワーク・アダプタ
コンピュータ・システム／サーバ
処理ユニット
I/Oインターフェース（群）
外部デバイス（群）
ディスプレイ
10
12
14
16
18
20
22
24
28
30
32
34
40
42

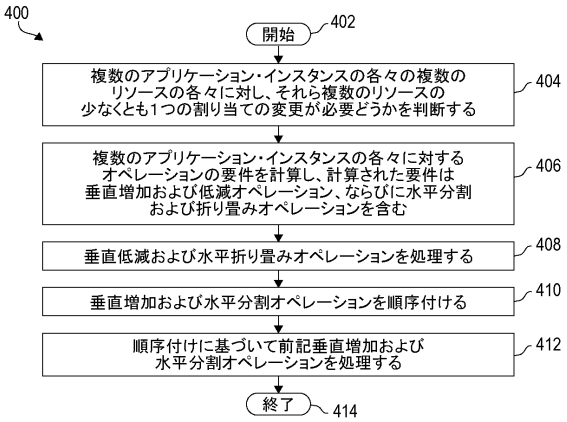
【図２】


50
10
54N
54B
54A
54C

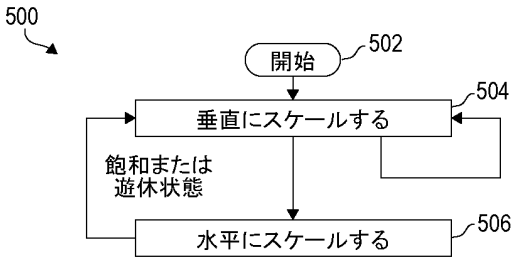
【図３】


作業負荷
管理
仮想化
ハードウェアおよびソフトウェア
デバイス層
90
80
70
60
55
91
92
93
94
95
96
81
82
83
84
85
71
72
73
74
75
61
62
63
64
65
66
67
68
52
53
56
57
58
59

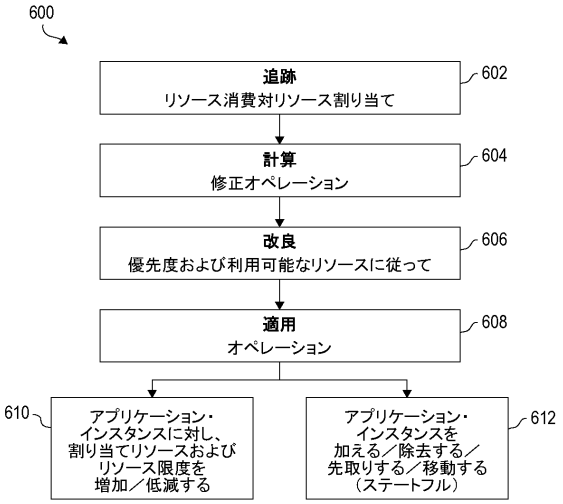
【図４】


400
開始
402
複数のアプリケーション・インスタンスの各々の複数のリソースの各々に対し、それら複数のリソースの少なくとも1つの割り当ての変更が必要どうかを判断する
404
複数のアプリケーション・インスタンスの各々に対するオペレーションの要件を計算し、計算された要件は垂直増加および低減オペレーション、ならびに水平分割および折り畳みオペレーションを含む
406
垂直低減および水平折り畳みオペレーションを処理する
408
垂直増加および水平分割オペレーションを順序付ける
410
順序付けに基づいて前記垂直増加および水平分割オペレーションを処理する
412
終了
414

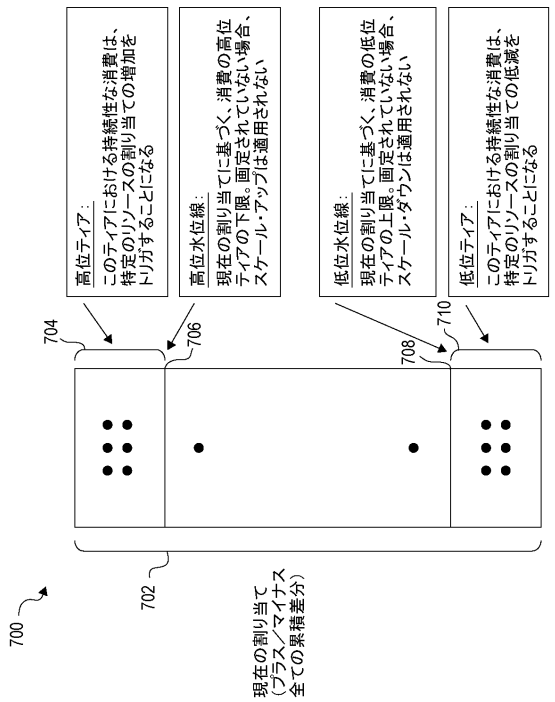
【図５】


500
開始
502
垂直にスケールする
504
飽和または
遊休状態
水平にスケールする
506

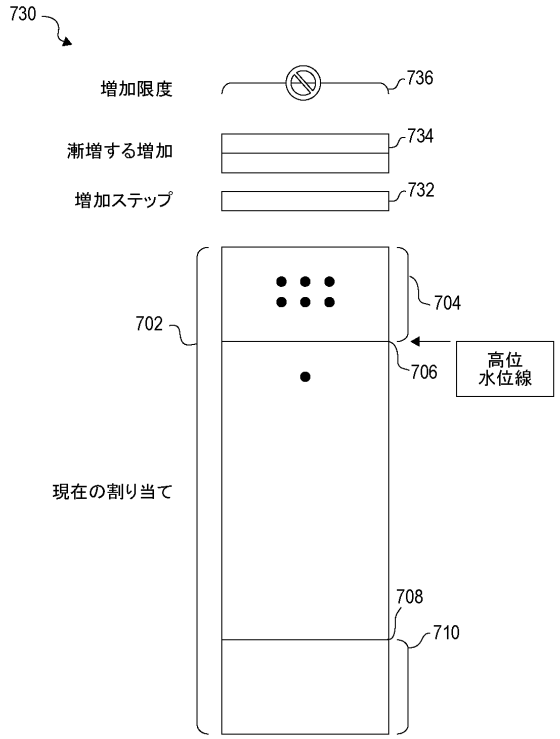
【図６】


600
追跡
リソース消費対リソース割り当て
602
計算
修正オペレーション
604
改良
優先度および利用可能なリソースに従って
606
適用
オペレーション
608
610
アプリケーション・
インスタンスに対し、
割り当てリソースおよび
リソース限度を
増加／低減する
アプリケーション・
インスタンスを
加える／除去する／
先取りする／移動する
（ステートフル）
612

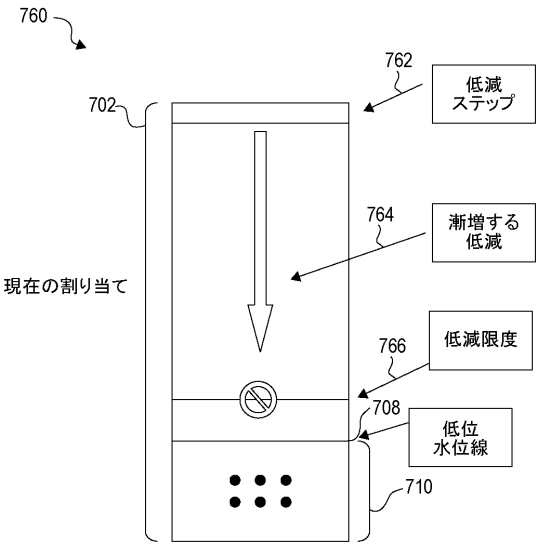
【図７】


高位ティア：
このティアにおける持続性な消費は、
特定のリソースの割り当ての増加を
トリガすることになる
高位水位線：
現在の割り当てに基づく、消費の高位
ティアの下限。画定されていない場合、
スケール・アップは適用されない
低位水位線：
現在の割り当てに基づく、消費の低位
ティアの上限。画定されていない場合、
スケール・ダウンは適用されない
低位ティア：
このティアにおける持続性な消費は、
特定のリソースの割り当ての低減を
トリガすることになる
704
706
708
710
700
702
現在の割り当て
（プラス／マイナス
全ての累積差分）

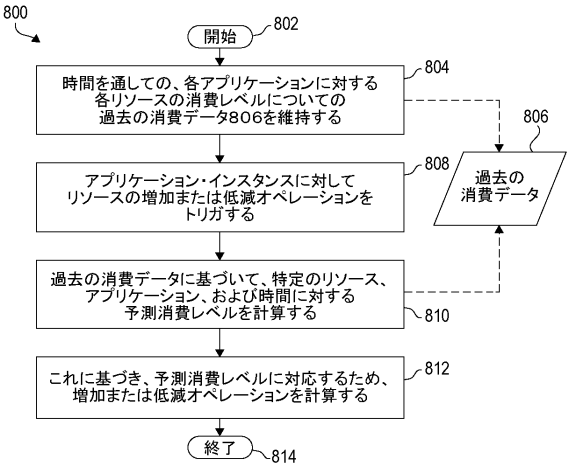
【図８】


730
増加限度
736
漸増する増加
734
増加ステップ
732
704
702
706
高位
水位線
現在の割り当て
708
710

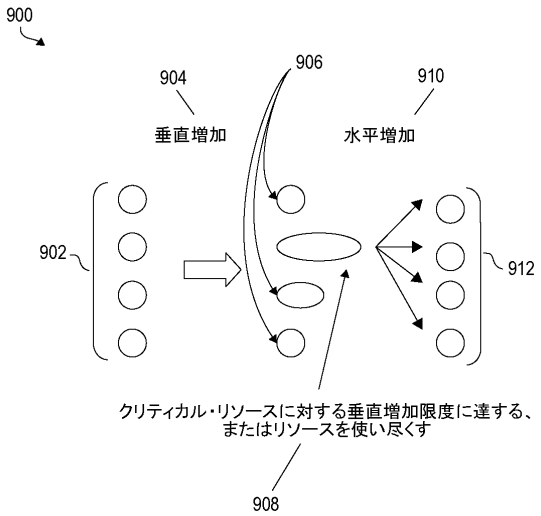
【図９】



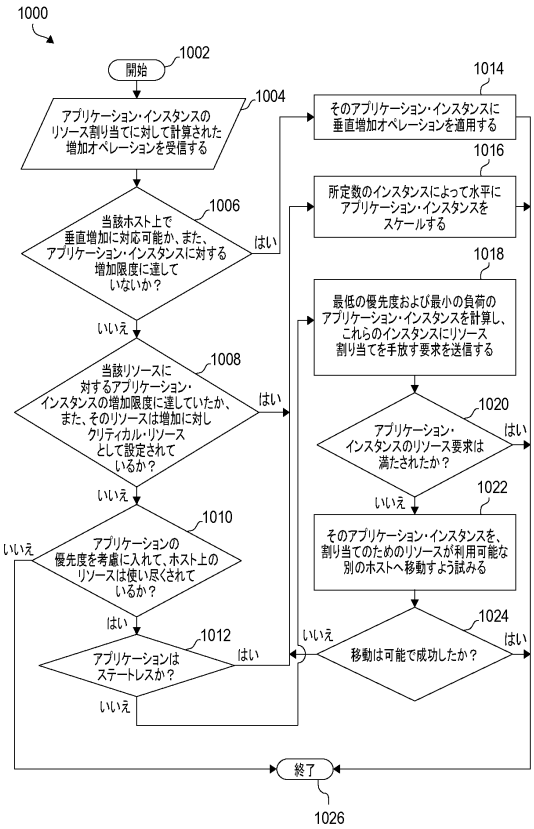
【図１０】



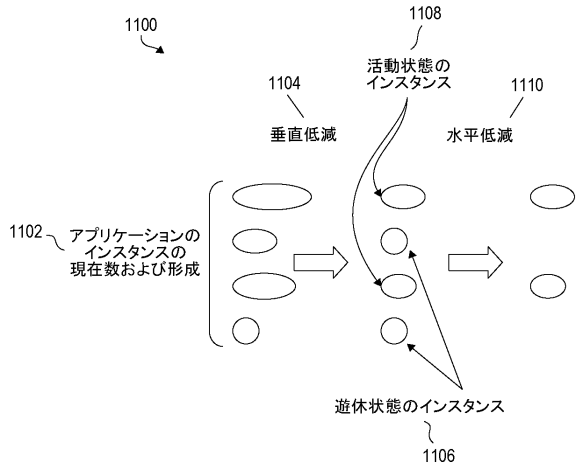
【図１１】



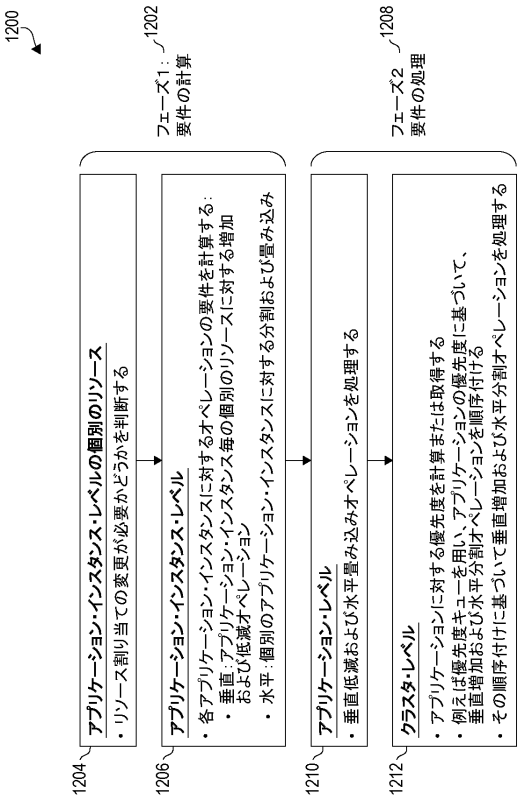
【図１２】



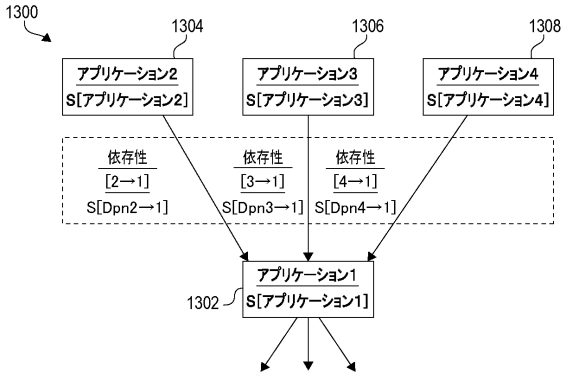
【図１３】



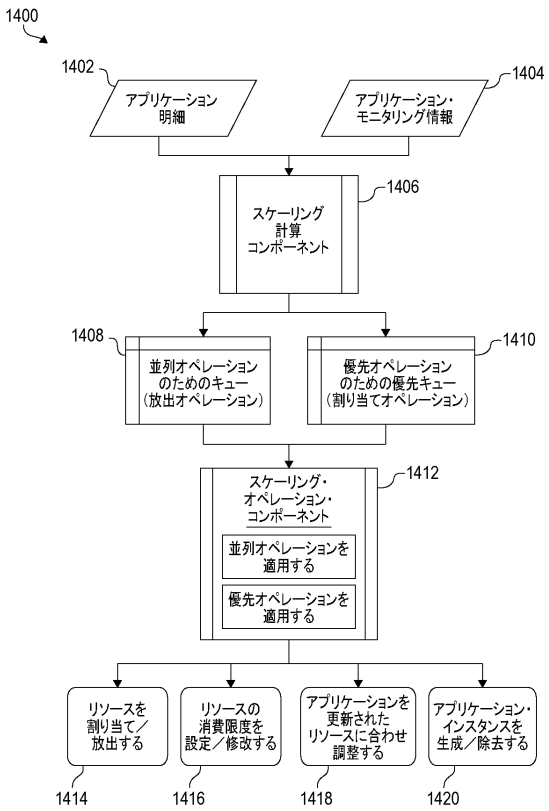
【図１４】



【図１５】



【図１６】



フロントページの続き

(72)発明者　アロノヴィッチ、リオル
カナダ　Ｌ３Ｒ　９Ｚ７　オンタリオ州　マーカム　スティールス・アベニュー・イースト　３６００
(72)発明者　アハメド、カリード
カナダ　Ｌ３Ｒ　９Ｚ７　オンタリオ州　マーカム　スティールス・アベニュー・イースト　３６００
(72)発明者　パスクアントニオ、ヴィンチェンツォ
カナダ　Ｌ６Ｇ　１Ｃ７　オンタリオ州　マーカム　ワーデン・アベニュー　８２００
(72)発明者　フェイマン、マイケル
カナダ　Ｌ３Ｒ　９Ｚ７　オンタリオ州　マーカム　スティールス・アベニュー・イースト　３６００

審査官　田中　幸雄
(56)参考文献　米国特許出願公開第２０１７／０３３１７０５（ＵＳ，Ａ１）
特開２０１５－１１５０５９（ＪＰ，Ａ）
アレスポー，ジョン，キャパシティプランニング，第1版，日本，株式会社オライリー・ジャパン，2009年03月18日，１１～２２ページ
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／５０
Ｇ０６Ｆ　９／４８