

(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2017-16693
(P2017-16693A)

(43) 公開日 平成29年1月19日(2017.1.19)

(51) Int.Cl.

G06F 9/50 (2006.01)

F I

G06F 9/46 465E

テーマコード (参考)

審査請求 有 請求項の数 12 O L (全 41 頁)

(21) 出願番号 特願2016-185469 (P2016-185469)
 (22) 出願日 平成28年9月23日(2016.9.23)
 (62) 分割の表示 特願2014-181494 (P2014-181494) の分割
 原出願日 平成19年6月7日(2007.6.7)
 (31) 優先権主張番号 60/813,738
 (32) 優先日 平成18年6月13日(2006.6.13)
 (33) 優先権主張国 米国 (US)
 (31) 優先権主張番号 60/850,908
 (32) 優先日 平成18年10月11日(2006.10.11)
 (33) 優先権主張国 米国 (US)
 (31) 優先権主張番号 11/744,461
 (32) 優先日 平成19年5月4日(2007.5.4)
 (33) 優先権主張国 米国 (US)

(71) 出願人 508368138
 タンネンバウム、ズヴィ
 アメリカ合衆国 カリフォルニア州 92
 657、ニューポート ビーチ、86 シ
 ドニー ベイ ドライブ
 (71) 出願人 508368127
 ドーガー、ディーン、イー。
 アメリカ合衆国 カリフォルニア州 92
 649、ハンティングトン ビーチ、35
 82 ペンチャー ドライブ
 (74) 代理人 110000729
 特許業務法人 ユニアス国際特許事務所

(特許庁注：以下のものは登録商標)

最終頁に続く

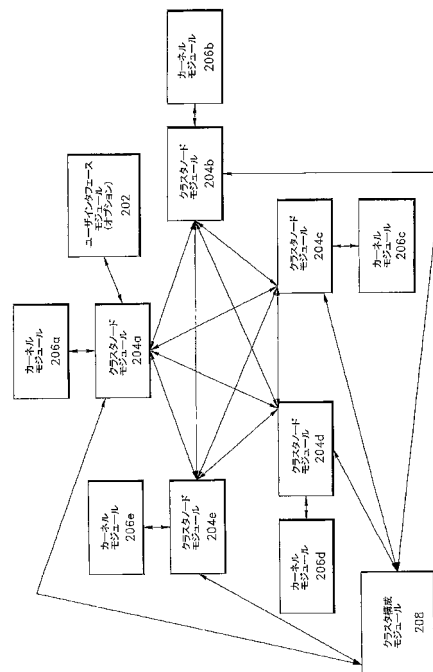
(54) 【発明の名称】 アプリケーションプログラムのためのクラスタコンピューティングのサポート

(57) 【要約】 (修正有)

【課題】複数のノードと、ユーザインタフェースおよびプログラムコード命令を翻訳するカーネルを含むソフトウェアパッケージと、を含むコンピュータクラスタシステムを提供する。

【解決手段】クラスタノードモジュールが、カーネルおよび他のクラスタノードモジュールと通信するように構成される。クラスタノードモジュールは、互いに通信し、かつ、カーネルと通信しているいくつかのクラスタノードモジュールがコンピュータクラスタとして動作することが可能であるように、ユーザインタフェースからの命令を受け付け、それらの命令のうちの少なくともいくつかを翻訳する。

【選択図】図2



【特許請求の範囲】**【請求項 1】**

1つまたは1つ以上の特別目的のマイクロプロセッサ上でコマンドを実行することによって、フロントエンドから受けた命令を実行するためのシステムであって、

複数のノードであって、各ノードが、シングルノードカーネルモジュールのプログラムコードを含むコンピュータ可読のメモリ装置にアクセスするように構成された複数のノードを備え、各シングルノードカーネルモジュールは、シングルノードカーネルモジュールによって受けた命令を、前記特別目的のマイクロプロセッサによって実行可能なコマンドに解釈し、

複数のクラスタノードモジュールであって、前記複数のクラスタノードモジュールが前記コンピュータ可読のメモリ装置に記憶され、かつ互いに通信して、1つまたは1つ以上のハードウェアプロセッサを用いてコマンドを実行するクラスタとして動作するように、命令を受けるために、シングルノードカーネルモジュールおよび1つまたは複数の他のクラスタノードモジュールと通信すること、前記命令の少なくともいくつかを解釈すること、を行うように構成された、複数のクラスタノードモジュールと、

前記複数のノード同士を接続するように構成された通信システムと、を備え、

前記複数のクラスタノードモジュールは、複数のシングルノードカーネルモジュールによる実行のための命令を、必要に応じて解釈することと翻訳することとを協働して行い、前記複数のクラスタノードモジュールの少なくとも1つが、結果を前記フロントエンドへ返す、システム。

【請求項 2】

前記特別目的のマイクロプロセッサは、デジタル信号プロセッサを備える、請求項 1 に記載のシステム。

【請求項 3】

前記複数のノードは、2つまたは2つ以上のグループのノードのサブセットとして編成される、請求項 1 に記載のシステム。

【請求項 4】

前記2つまたは2つ以上のグループのノードのサブセットは、前記特別目的のマイクロプロセッサとデータのやり取りをする、請求項 3 に記載のシステム。

【請求項 5】

前記特別目的のマイクロプロセッサは、マルチプルプロセッサコアを備える、請求項 1 に記載のシステム。

【請求項 6】

前記複数のクラスタノードモジュールの少なくとも1つは、プロセッサキャッシュメモリに記憶されている、請求項 1 に記載のシステム。

【請求項 7】

前記各シングルノードカーネルモジュールは、プロセッサキャッシュメモリに記憶されている、請求項 1 に記載のシステム。

【請求項 8】

1つまたは1つ以上のハードウェアプロセッサ上でコマンドを実行することによって、フロントエンドから受けた命令を実行するためのシステムであって、

複数のノードであって、各ノードが、シングルノードカーネルモジュールのプログラムコードを含むコンピュータ可読のメモリ装置にアクセスするように構成された複数のノードを備え、各シングルノードカーネルモジュールは、シングルノードカーネルモジュールによって受けた命令を、前記ハードウェアプロセッサによって実行可能なコマンドに解釈し、前記コマンドが、要素のリストから1つまたは1つ以上の要素上で計算が行われるように構成され、

複数のクラスタノードモジュールであって、前記複数のクラスタノードモジュールが前記コンピュータ可読のメモリ装置に記憶され、かつ互いに通信して、要素のリスト上で計算が行われるクラスタとして動作するように、命令を受けるために、シングルノードカー

10

20

30

40

50

ネルモジュールおよび1つまたは複数の他のクラスタノードモジュールと通信すること、前記命令の少なくともいくつかを解釈すること、を行うように構成された、複数のクラスタノードモジュールと、

前記複数のノード同士を接続するように構成された通信システムと、を備え、

前記複数のクラスタノードモジュールは、複数のシングルノードカーネルモジュールによる実行のための命令を、必要に応じて解釈することと翻訳することとを協働して行い、

前記要素のリストが、複数のノードの実行のために分割され、前記複数のクラスタノードモジュールの少なくとも1つが、結果を前記フロントエンドへ返す、システム。

【請求項9】

前記要素のリストの1つまたは1つ以上の要素が、異なるノードに移行する、請求項8に記載のシステム。

10

【請求項10】

コンピュータクラスタにおいてコマンドを評価する方法であって、

ユーザインタフェースまたはスクリプトの少なくとも一方からのコマンドを、前記コンピュータクラスタ内の1つまたは複数のクラスタノードモジュールに伝達するステップと、

前記1つまたは複数のクラスタノードモジュールのそれぞれが、前記コマンドに基づくメッセージを、前記クラスタノードモジュールに関連付けられた各カーネルモジュールに伝達するステップと、

前記1つまたは複数のクラスタノードモジュールのそれぞれが、前記クラスタノードモジュールに関連付けられた前記各カーネルモジュールから結果を受け取るステップと、

20

前記1つまたは複数のクラスタノードモジュールのうちの少なくとも1つが、他のクラスタノードモジュールからのメッセージに応答するステップと、を含む方法。

【請求項11】

前記コマンドに基づくメッセージを、前記クラスタノードモジュールに関連付けられた各カーネルモジュールに伝達する前記ステップは、特別に識別されたメッセージを前記各カーネルモジュールに伝達するステップを含む、請求項10に記載の方法。

【請求項12】

前記クラスタノードモジュールのうちの少なくとも1つが、前記クラスタノードモジュールと同じコンピュータシステムで実行されているユーザインタフェースまたはスクリプトのうちの少なくとも一方へ前記結果を転送するステップをさらに含む、請求項10に記載の方法。

30

【請求項13】

前記クラスタノードモジュールのうちの少なくとも1つが、前記結果を、メッセージとして、1つまたは複数の他のクラスタノードモジュールに伝達するステップをさらに含む、請求項10に記載の方法。

【請求項14】

他のクラスタノードモジュールからのメッセージに応答する前記ステップは、

前記他のクラスタノードモジュールからのメッセージを、受信済みメッセージキューに転送するステップと、

40

メッセージ受信キュー内の各エントリからのデータを、前記受信済みメッセージキュー内のエントリと照合するステップと、

前記メッセージ受信キューからのデータを、前記受信済みメッセージキュー内の合致するデータと結合するステップと、

前記合致データを完了としてマーキングするステップと、を含む、請求項10に記載の方法。

【請求項15】

ユーザインタフェースまたはスクリプトの少なくとも一方からのコマンドを、前記コンピュータクラスタ内の1つまたは複数のクラスタノードモジュールに伝達する前記ステップは、Mathematicaフロントエンドからの命令を第1のクラスタノードモジュ

50

ールに伝達するステップを含み、前記第1のクラスタノードモジュールは、前記命令を、前記コンピュータクラスタで実行されている他のクラスタノードモジュールに転送する、請求項10に記載の方法。

【請求項16】

前記第1のクラスタノードモジュールは、メッセージバッシングインタフェースからのコマンドを用いて、前記命令を、前記コンピュータクラスタで実行されている他のクラスタノードモジュールに転送する、請求項15に記載の方法。

【請求項17】

前記1つまたは複数のクラスタノードモジュールのそれぞれは、MathLinkを用いて、それぞれのカーネルモジュールと通信している、請求項10に記載の方法。

10

【請求項18】

複数のノードにおいてMathematicaコードを実行するコンピューティングシステムであって、

第1のノードで実行されている第1のMathematicaカーネルと通信している、第1のノードモジュールと、

第2のノードで実行されている第2のMathematicaカーネルと通信している、第2のノードモジュールと、

第3のノードで実行されている第3のMathematicaカーネルと通信している、第3のノードモジュールと、を備え、

前記第1のノードモジュール、前記第2のノードモジュール、および前記第3のノードモジュールは、ピアツーピアアーキテクチャを用いて互いに通信するように構成された、コンピューティングシステム。

20

【請求項19】

前記第1のノードモジュール、前記第2のノードモジュール、および前記第3のノードモジュールのそれぞれは、他のノードモジュールから発せられたメッセージを保持するデータ構造と、メッセージの受信先として期待されるロケーションと、前記メッセージの送信元として期待されるノードの識別子と、を指定するデータを保持するデータ構造と、を備える、請求項18に記載のコンピューティングシステム。

【請求項20】

前記複数のクラスタノードモジュールは、少なくとも3つのクラスタノードモジュールを備える、請求項1に記載のシステム。

30

【発明の詳細な説明】

【技術分野】

【0001】

(優先権情報)

本出願は、2006年6月13日に出願された米国特許仮出願第60/813738号、ならびに2006年10月11日に出願された米国特許仮出願第60/850908号の優先権を主張するものである。上記参照された各出願は、その全体が、参照によって本明細書に組み込まれ、本明細書の一部になっている。

【0002】

40

本開示は、主としてクラスタコンピューティングの分野に関し、特に、コンピュータプログラムにクラスタコンピューティング機能性を追加するシステムおよび方法に関する。

【背景技術】

【0003】

コンピュータクラスタは、相互通信することにより、あたかも単一のコンピュータであるかのようにタスクを達成することが可能である、2つ以上のコンピュータ、マイクロプロセッサ、および/またはプロセッサコア(「ノード」)からなるグループを含む。現在、多くのコンピュータアプリケーションプログラムは、たとえ、それらが、クラスタとして動作することが可能なノードのグループにおいて実行されているとしても、コンピュータクラスタが提供しうる利点の恩恵を受けるようには設計されていない。コンピュータ

50

プログラムの中には、単一ノードでしか実行できないものがあるが、これは、たとえば、それらがタスクを直列に実行するようにコーディングされているためであったり、単一ノードのみを認識するように、あるいは単一ノードにのみ命令を送信するように設計されているためであったりする。

【発明の概要】

【発明が解決しようとする課題】

【0004】

アプリケーションプログラムの中には、ユーザ、スクリプト、または別のソースによってそのプログラムに与えられる命令を実行するインタプリタを含むものがある。そのようなインタプリタは、「カーネル」と呼ばれることがあり、それは、たとえば、インタプリタが、コンピュータシステムの少なくともいくつかのハードウェアリソースを管理することが可能であるため、かつ/または、それらのリソースとソフトウェア（たとえば、高級プログラミング言語を含むことが可能な与えられた命令）との間の通信を管理することが可能であるためである。ソフトウェアプログラムの中には、単一ノードと通信するように設計されたカーネルを含むものがある。単一ノードと通信するように設計されたカーネルを含むソフトウェアパッケージの一例として、Wolfram Research, Inc. 製の Mathematica (登録商標) (「Mathematica」) がある。他のベンダ製の数学ソフトウェアパッケージや他の種類のソフトウェアも、そのようなカーネルを含むことが可能である。

【0005】

同じく Wolfram Research, Inc. 製である、grid Mathematica として知られる製品が、「分散コンピューティング」として知られる形式のグリッドコンピューティングを実行する機能を、Mathematica に与える。グリッドコンピュータは、一般にピアとして互いに通信することがない複数のノードを含む。分散コンピューティングは、計算処理時にジョブ間でデータを共有する必要がない、多数の独立ジョブ、作業パケットからなる作業負荷に対して最適化されることが可能である。グリッドコンピュータは、複数のスレーブノードまたは計算ノードを管理する、マスタノードとして知られる、少なくとも1つのノードを含む。grid Mathematica では、複数のカーネルのそれぞれが、単一ノードで実行される。1つのカーネルが、他のカーネル（計算カーネルまたはスレーブカーネル）のすべての入力、出力、およびスケジューリングを取り扱うマスタカーネルに指定される。計算カーネルは、マスタカーネルを実行しているノードからのみコマンドおよびデータを受け取る。各計算カーネルは、各自の作業を、他の計算カーネルとは独立に実行し、あるジョブの中間結果は、他のノードで進行中の他のジョブに影響を及ぼさない。

【課題を解決するための手段】

【0006】

本明細書に記載の実施形態は、いくつかの特徴を有するが、それらの1つ1つは、それぞれの望ましい属性を単独で担うものではない。以下では、特許請求の範囲で表される本発明の範囲を限定することなく、有利な特徴のいくつかについて簡単に説明する。

【0007】

本明細書に記載のいくつかの実施形態は、コンピュータアプリケーションにクラスタコンピューティング機能性を便利に追加する手法を提供する。一実施形態では、ソフトウェアパッケージのユーザが、そのソフトウェアがクラスタ内の複数のノードから恩恵を受けることを可能にすることにより、そのソフトウェアパッケージから、より高い性能、および/または、より高い可用性を達成できるであろう。一実施形態は、スーパーコンピュータレベルの性能を有するコンピュータクラスタで実行されることが可能なアプリケーションを、ユーザが、Mathematica のような高級言語を用いて作成することを可能にする。一実施形態は、Mathematica フロントエンド、コマンドラインインタフェース、1つまたは複数の高級コマンド、またはCやFORTRAN のようなプログラミング言語を用いる、そのような高性能コンピューティングへのアクセスを提供する。

【0008】

一実施形態は、シングルノードで実行されるように設計された、たとえば、*Mathematica*カーネルなどのソフトウェアモジュールを、たとえそのソフトウェアモジュールがサポートを提供するように設計されていない場合であっても、クラスタコンピューティングをサポートするよう適応させる。一実施形態は、そのプログラムのソースコードへのアクセスが可能でない場合でも、アプリケーションプログラムの並列化を提供する。一実施形態は、メッセージパッシングインタフェース（「MPI」）呼び出しを、たとえば、*Mathematica*プログラミング環境などのユーザインタフェースの中から直接追加およびサポートする。一実施形態では、MPI呼び出しは、*Mathematica*フロントエンドなどのインタラクティブプログラミング環境に追加され、そこから使用

10

【0009】

一実施形態は、第1のプロセッサ、第2のプロセッサ、および第3のプロセッサを含むコンピュータクラスタを提供する。クラスタは、第1のプロセッサ、第2のプロセッサ、または第3のプロセッサのうち少なくとも1つと通信している少なくとも1つのコンピュータ可読媒体を含む。第1のカーネルが、少なくとも1つのコンピュータ可読媒体に常駐し、コマンドを、第1のプロセッサで実行されるコードに変換するように構成される。第1のクラスタノードモジュールが、少なくとも1つのコンピュータ可読媒体に常駐する。第1のクラスタノードモジュールは、第1のカーネルへコマンドを送信し、ユーザインタフェースからコマンドを受信するように構成される。第2のカーネルが、少なくとも1つのコンピュータ可読媒体に常駐する。第2のカーネルは、コマンドを、第2のプロセッサで実行されるコードに変換するように構成される。第2のクラスタノードモジュールが、少なくとも1つのコンピュータ可読媒体に常駐する。第2のクラスタノードモジュールは、第2のカーネルへコマンドを送信するように構成され、第1のクラスタノードモジュールと通信する。第3のカーネルが、少なくとも1つのコンピュータ可読媒体に常駐する。第3のカーネルは、コマンドを、第3のプロセッサで実行されるコードに変換するように構成される。第3のクラスタノードモジュールが、少なくとも1つのコンピュータ可読媒体に常駐する。第3のクラスタノードモジュールは、第3のカーネルへコマンドを送信するように構成され、第1のクラスタノードモジュールおよび第2のクラスタノードモジュールと通信するように構成される。第1のクラスタノードモジュールは、第2および第3のクラスタノードモジュールから発せられたメッセージが記憶されるデータ構造を備える。

20

30

【0010】

別の実施形態は、複数のノードと、ユーザインタフェースおよびプログラムコード命令を翻訳するシングルノードカーネルを含むソフトウェアパッケージと、を含むコンピュータクラスタを提供する。クラスタノードモジュールが、シングルノードカーネルおよび他のクラスタノードモジュールと通信するように構成される。クラスタノードモジュールは、互いに通信しているいくつかのクラスタノードモジュールがクラスタとして動作するように、ユーザインタフェースからの命令を受け付け、それらの命令のうち少なくともいくつかを翻訳する。クラスタノードモジュールは、ユーザインタフェースからは、シングルノードカーネルのように見える。一実施形態では、この、シングルノードカーネルは、*Mathematica*カーネルを含む。実施形態によっては、ユーザインタフェースは、*Mathematica*フロントエンドまたはコマンドラインのうち少なくとも一方を含むことが可能である。実施形態によっては、クラスタノードモジュールは、MPI呼び出しの少なくとも一部を実施するライブラリ呼び出しを含むツールキットを含む。実施形態によっては、クラスタノードモジュールは、高級クラスタコンピューティングコマンドを含むツールキットを含む。一実施形態では、クラスタシステムは、複数の*Macintosh*（登録商標）コンピュータ（「Mac」）、*Windows*（登録商標）ベースのパーソナルコンピュータ（「PC」）、および/または*Umx/Linux*ベースのワークステーションを含むことが可能である。

40

50

【 0 0 1 1 】

さらなる実施形態は、複数のノードを含むコンピュータクラスタを提供する。各ノードは、ユーザインタフェースのプログラムコードと、ユーザ命令を翻訳するように構成されたシングルノードカーネルモジュールのプログラムコードと、を備えるコンピュータ可読媒体にアクセスするように構成される。クラスタは、複数のクラスタノードモジュールを含む。この複数のクラスタノードモジュールが互いに通信してクラスタとして動作するように、各クラスタノードモジュールは、シングルノードカーネルおよび1つまたは複数の他のクラスタノードモジュールと通信することと、ユーザインタフェースから命令を受け付けることと、ユーザ命令の少なくともいくつかを翻訳することと、を行うように構成される。通信ネットワークが、これらのノードを接続する。複数のクラスタノードモジュールの1つが、結果をユーザインタフェースへ返す。

10

【 0 0 1 2 】

別の実施形態は、コンピュータクラスタにおいてコマンドを評価する方法を提供する。ユーザインタフェースまたはスクリプトの少なくとも一方からのコマンドが、コンピュータクラスタ内の1つまたは複数のクラスタノードモジュールに伝達される。その1つまたは複数のクラスタノードモジュールのそれぞれが、そのコマンドに基づくメッセージを、そのクラスタノードモジュールに関連付けられた各カーネルモジュールに伝達する。その1つまたは複数のクラスタノードモジュールのそれぞれが、そのクラスタノードモジュールに関連付けられた各カーネルモジュールから結果を受け取る。その1つまたは複数のクラスタノードモジュールのうちの少なくとも1つが、他のクラスタノードモジュールからのメッセージに応答する。

20

【 0 0 1 3 】

別の実施形態は、複数のノードにおいて Mathematica コードを実行するコンピューティングシステムを提供する。このコンピューティングシステムは、第1のノードで実行されている第1の Mathematica カーネルと通信する第1のノードモジュールと、第2のノードで実行されている第2の Mathematica カーネルと通信する第2のノードモジュールと、第3のノードで実行されている第3の Mathematica カーネルと通信する第3のノードモジュールと、を含む。第1のノードモジュール、第2のノードモジュール、および第3のノードモジュールは、ピアツーピアアーキテクチャを用いて互いに通信するように構成される。実施形態によっては、第1のノードモジュール、第2のノードモジュール、および第3のノードモジュールのそれぞれは、他のノードモジュールから発せられたメッセージを保持するデータ構造と、メッセージの受信先として期待されるロケーション、およびそのメッセージの送信元として期待されるノードの識別子を指定するデータを保持するデータ構造と、を含む。

30

【 0 0 1 4 】

図面を参照しながら、種々の特徴を実施する全体のアーキテクチャを説明する。各図面および関連する説明は、実施形態を例示するために提供されており、本開示の範囲を限定するものではない。図面全体を通じて、参照される要素の間の対応を示すために、参照符号を繰り返し使用している。

40

【 図面の簡単な説明 】

【 0 0 1 5 】

【 図 1 】 コンピュータクラスタの一実施形態のブロック図である。

【 図 2 】 コンピュータクラスタの一実施形態で実行されているソフトウェアモジュール同士の関係を示すブロック図である。

【 図 3 】 クラスタノードモジュールの一実施形態のブロック図である。

【 図 4 】 クラスタ初期化プロセスの一実施形態を示すフローチャートである。

【 図 5 】 クラスタノードモジュールの動作の一実施形態を示すフローチャートである。

【 発明を実施するための形態 】

【 0 0 1 6 】

本明細書では、例示を目的として、いくつかの実施形態を、 Mathematica ソ

50

ソフトウェアを用いるクラスタコンピューティングのコンテキストで説明する。本開示は、単一のソフトウェアプログラムに限定されず、本システムおよび方法は、他のアプリケーションソフトウェアとともに使用されることが可能であり、そのようなソフトウェアとして、たとえば、Maple（登録商標）、MATLAB（登録商標）、MathCAD（登録商標）、Apple Shake（登録商標）、Apple（登録商標） Compressor、IDL（登録商標）、他の、インタプリタまたはカーネルを用いるアプリケーション、Microsoft Excel（登録商標）、Adobe After Effects（登録商標）、Adobe Premiere（登録商標）、Adobe Photoshop（登録商標）、Apple Final Cut Pro（登録商標）、Apple iMovie（登録商標）などがある。しかしながら、いくつかの図面および/または説明は、Mathematicaを実行するコンピュータクラスタの実施形態に関連している。本システムは、多様な用途を含むことが可能であり、そのような用途には、これらに限定されないが、学生、教育者、科学者、技術者、数学者、研究者、技能者などが含まれる。また、他の実施形態では、本システムおよび方法は、単一モジュールとして実装されること、および/または、他の様々なモジュールとともに実装されることが可能であることも理解されよう。さらに、本明細書に記載の個々の実施態様は、本開示を限定するためではなく、本開示の例を示すために説明されている。

10

I. 概要

【0017】

本明細書に記載のクラスタコンピューティングシステムは、一般に、1つまたは複数の通信ネットワークを介して互いに接続された1つまたは複数のコンピュータシステムを含む。この通信ネットワークは、ローカルエリアネットワーク（「LAN」）、ワイドエリアネットワーク（「WAN」）、イントラネット、インターネットなどのうちの1つまたは複数を含むことが可能である。一実施形態では、コンピュータシステムは、たとえば、1つまたは複数のプロセッサコア（「ノード」）を含むことが可能なマイクロプロセッサのような、1つまたは複数のプロセッサを備える。「ノード」という用語は、コードのシングルスレッド実行が可能なプロセッサユニットまたはプロセッササブユニットを意味する。プロセッサは、たとえば、ランダムアクセスメモリ（「RAM」）のような、1つまたは複数のメモリ装置、および/または、たとえば、ハードディスクのような、1つまたは複数の光ストレージ装置に接続されることが可能である。プロセッサとそのような他の装置との間の通信は、たとえば、コンピュータシステムの1つまたは複数のローカルバスを介して、あるいは、LAN、WAN、ストレージエリアネットワーク（「SAN」）、および/または他の任意の、コンピュータシステムコンポーネント間で信号を搬送することが可能な通信ネットワークを介して、行われることが可能である。一実施形態では、カーネルのような、1つまたは複数のソフトウェアモジュールが、相互接続された複数のコンピュータシステムの中のノードで実行される。一実施形態では、カーネルは、単一ノードでのみ実行されるように設計される。一実施形態では、クラスタコンピューティング機能性を実装するために、クラスタノードモジュールが、カーネルと通信し、クラスタノードモジュール同士で通信する。

20

30

【0018】

図1は、コンピュータクラスタ100の一実施形態のブロック図であり、ここでは、コンピュータシステム110、120、130が、通信ネットワーク102を介して互いに通信している。ネットワーク102は、LAN、WAN、無線ネットワーク、イントラネット、またはインターネットのうちの1つまたは複数を含む。本コンピュータクラスタの一実施形態では、コンピュータシステム110は、プロセッサ112a、112b、メモリ114、およびオプションのストレージ116を含む。他のコンピュータシステム120、130も同様の装置を含むことが可能であり、それらの装置は、一般に、コンピュータシステム内で、ローカルバス（図示せず）のようなローカル通信アーキテクチャを介して互いに通信している。コンピュータシステムは、1つまたは複数のプロセッサを含むことが可能であり、各プロセッサは、シングルスレッド実行が可能な、1つまたは複数のプ

40

50

ロセッサコアを含むことが可能である。プロセッサコアは、一般には、独立したマイクロプロセッサであるが、複数のプロセッサコアが単一チップパッケージに含まれることも可能である。シングルスレッド実行を行うように設計されたソフトウェアコードは、一般に、一度に1つのプロセッサコアで実行されることが可能である。たとえば、シングルスレッドソフトウェアコードは、典型的には、コンピュータシステム内に複数のプロセッサコアがあることの恩恵を受けない。

【0019】

図2は、コンピュータクラスタ100の一実施形態で実行されているソフトウェアモジュール同士の関係を示すブロック図である。図2に示された実施形態では、カーネルモジュール206a~eは、シングルスレッド実行を行うように設計されている。たとえば、図1に示されたプロセッサ112a、112b、122a、122b、132のそれぞれが、プロセッサコアを1つだけ含む場合、コンピュータシステム110のメモリ114にロードされた2つのカーネルモジュール(たとえば、カーネルモジュール206a、206b)は、2つのプロセッサ112a、112bの処理用帯域幅の少なくとも一部を活用することが可能である。同様に、コンピュータシステム120のメモリ124にロードされた2つのカーネルモジュール206c、206dは、2つのプロセッサ122a、122bの処理用帯域幅の少なくとも一部を活用することが可能である。同様に、コンピュータシステム130のプロセッサ132の帯域幅は、コンピュータシステムのメモリ134にロードされた、クラスタノードモジュール204eの単一インスタンスによって利用されることが可能である。

10

20

【0020】

図2に示された実施形態では、カーネルモジュール206a~eのそれぞれが、単一クラスタノードモジュール204a~eとそれぞれ通信している。たとえば、カーネルモジュール206aは、クラスタノードモジュール204aと通信しており、カーネルモジュール206bは、クラスタノードモジュール206bと通信しており、他も同様である。一実施形態では、クラスタノードモジュール204a~eの1つのインスタンスが、コンピュータシステムで実行されているカーネルモジュール206a~eのインスタンスごとに、コンピュータシステムのメモリ114、124、134にロードされている。図2に示されるように、クラスタノードモジュール204a~eのそれぞれが、他のクラスタノードモジュール204a~eのそれぞれと通信している。たとえば、1つのクラスタノードモジュール204aは、他のすべてのクラスタノードモジュール204b~eと通信している。クラスタノードモジュール204aは、たとえば、両方のクラスタノードモジュール204a~bが同じコンピュータシステム110にあるプロセッサ112a、112bで実行されている場合に、ローカルバス(図示せず)を介して別のクラスタノードモジュール204bと通信することが可能である。クラスタノードモジュール204aはまた、たとえば、クラスタノードモジュール204a、cが、異なるコンピュータシステム110、120にあるプロセッサ112a、122aで実行されている場合に、通信ネットワーク102を介して別のクラスタノードモジュール204cと通信することも可能である。

30

【0021】

図2に示されるように、たとえば、Mathematicaフロントエンドおよび/またはコマンドラインインタフェースなどのオプションのユーザインタフェースモジュール202が、クラスタノードモジュール204aに接続可能である。ユーザインタフェースモジュールは、クラスタノードモジュール204aが実行されている、同じコンピュータシステム110および/または同じマイクロプロセッサ112aで実行可能である。クラスタノードモジュール204a~eは、シングルスレッドカーネルモジュールにクラスタコンピューティング機能を実装するMPI呼び出しおよび/または高度クラスタ機能を提供する。クラスタノードモジュール204a~eは、ユーザインタフェースモジュール202から見れば、カーネルモジュール206aのように見えてカーネルモジュール206aのようにふるまうように構成されている。同様に、クラスタノードモジュール202a

40

50

～ e は、カーネルモジュール 206 a から見れば、ユーザインタフェースモジュール 202 のように見えてユーザインタフェースモジュール 202 のようにふるまうように構成されている。第 1 のクラスタノードモジュール 204 a は、他の 1 つまたは複数のクラスタノードモジュール 204 b、204 c などと通信しており、これらのそれぞれが、MPI 呼び出しおよび/または高度クラスタコマンドのセットを提供する。一実施形態では、MPI は、コンピュータクラスタにおけるノード間メッセージ送信に用いられることが可能である。

【0022】

「隣接する」カーネル同士の間に限らない、任意の 2 つ以上のクラスタノードモジュールの間（たとえば、クラスタノードモジュール 204 a と別のクラスタノードモジュール 204 c との間）で通信が行われることが可能である。クラスタノードモジュール 204 a～e のそれぞれは、それぞれのカーネルモジュール 206 a～e と通信している。したがって、クラスタノードモジュール 204 a は、カーネルモジュール 206 a と通信している。MPI 呼び出しおよび高度クラスタコマンドは、オプションのユーザインタフェースモジュール 208 から受け取られたプログラムコードを並列化し、タスクをカーネルモジュール 206 a～e に分配するために用いられる。クラスタノードモジュール 204 a～e は、それらのタスクが実行されている間のカーネルモジュール 206 a～e 間の通信を提供する。カーネルモジュール 206 a～e によって実行された評価の結果は、クラスタノードモジュール 204 a～e を介して第 1 のクラスタノードモジュール 204 a へ返され、第 1 のクラスタノードモジュール 204 a は、それらの結果をユーザインタフェースモジュール 208 へ伝達する。

10

20

【0023】

スレッド実行の間のカーネルモジュール 206 a～e 間の相互通信は、クラスタノードモジュール 204 a～e によって可能にされ、たとえば、様々な種類の数学的問題および科学的問題に取り組むことに関して有利に働く。クラスタコンピューティングによって提供される相互通信は、並列計算の過程におけるノード間の情報交換を可能にする。本開示の諸実施形態は、そのような相互通信を、Mathematica のようなソフトウェアプログラムに提供する一方、グリッドコンピューティングソリューションは、1 つのマスタノードと多数のスレーブノードとの間でのみ通信を実施することが可能である。グリッドコンピューティングでは、スレッド実行の間は、スレーブノード間で通信を行うことができない。

30

【0024】

本明細書では、いくつかの実施形態の概要を与える目的で、本発明の特定の態様、利点、利益、および新規な特徴について説明する。本発明の任意の特定の実施形態によれば、そのような利点または利益のすべてが、必ずしも実現可能ではないことを理解されたい。したがって、たとえば、当業者であれば理解されるように、本発明は、本明細書で教示されるように 1 つまたは複数の利点を達成する様式で、本明細書で教示または提案されうるような他の利点または利益を必ずしも達成することなく、実施または実行されることが可能である。

II. コンピュータクラスタ 100

40

【0025】

図 1 に示されるように、クラスタシステム 100 の一実施形態は、通信ネットワーク 102 を介して互いに通信しているコンピュータシステム 110、120、130 を含む。第 1 のコンピュータシステム 110 は、1 つまたは複数のプロセッサ 112 a～b、メモリ装置 114、およびオプションのストレージ装置 116 を含むことが可能である。同様に、第 2 のコンピュータシステム 120 は、1 つまたは複数のプロセッサ 122 a～b、メモリ装置 124、およびオプションのストレージ装置 126 を含むことが可能である。同様に、第 3 のコンピュータシステム 130 は、1 つまたは複数のプロセッサ 132、メモリ装置 134、およびオプションのストレージ装置 136 を含むことが可能である。コンピュータシステム 110、120、130 のそれぞれは、通信ネットワーク 102 に接

50

続されるためのネットワークインタフェース（図示せず）を含み、通信ネットワーク 102 は、LAN、WAN、イントラネット、無線ネットワーク、および/またはインターネットのうちの 1 つまたは複数を含むことが可能である。

A. コンピュータシステム 110

【0026】

一実施形態では、第 1 のコンピュータシステム 110 は、コンピュータクラスタ 100 の一部として、ネットワーク 102 を介して、他のコンピュータシステム 120、130 と通信している。一実施形態では、コンピュータシステム 110 は、1 つまたは複数のプロセッサ 112 a ~ b、メモリ装置 114、オプションのストレージ装置 116、ならびに、ネットワーク 102 との通信のためのネットワークインタフェースモジュール（図示せず）を含むパーソナルコンピュータ、ワークステーション、サーバ、またはブレードである。

10

1. プロセッサ 112 a ~ b

【0027】

一実施形態では、コンピュータシステム 110 は、1 つまたは複数のプロセッサ 112 a ~ b を含む。プロセッサ 112 a ~ b は、1 つまたは複数の汎用シングルコアマイクロプロセッサまたはマルチコアマイクロプロセッサであることが可能であり、そのようなプロセッサとして、たとえば、Pentium（登録商標）プロセッサ、Pentium（登録商標）II プロセッサ、Pentium（登録商標）Pro プロセッサ、Pentium（登録商標）III プロセッサ、Pentium（登録商標）4 プロセッサ、Core Duo（登録商標）プロセッサ、Core 2 Duo（登録商標）プロセッサ、Xeon（登録商標）プロセッサ、Itanium（登録商標）プロセッサ、Pentium（登録商標）M プロセッサ、x86 プロセッサ、Athlon（登録商標）プロセッサ、8051 プロセッサ、MIPS（登録商標）プロセッサ、PowerPC（登録商標）プロセッサ、ALPHA（登録商標）プロセッサなどがある。さらに、プロセッサ 112 a ~ b のうちの 1 つまたは複数が、デジタル信号プロセッサのような専用マイクロプロセッサであることが可能である。コンピュータシステム 110 内にあるすべてのプロセッサ 112 a ~ b の中のプロセッサコア（たとえば、シングルスレッド実行が可能なプロセッサユニット）の総数は、コンピュータシステム 110 内で使用可能なノードの数に対応する。たとえば、プロセッサ 112 a ~ b のそれぞれが、2 つのプロセッサコアを有する Core 2 Duo（登録商標）プロセッサであった場合、コンピュータシステム 110 は、全部で 4 つのノードを有することになる。各ノードは、シングルスレッドカーネルモジュールのようなプログラムモジュールの 1 つまたは複数のインスタンスを実行することが可能である。

20

30

2. ネットワークインタフェースモジュール

【0028】

コンピュータシステム 110 はさらに、コンピュータシステム 110 と他のコンピュータシステム 120、130 との間の、通信ネットワーク 102 を介する通信を容易にするネットワークインタフェースモジュール（図示せず）を含むことが可能である。

【0029】

ネットワークインタフェースモジュールは、様々なネットワークプロトコルを使用することが可能である。一実施形態では、ネットワークインタフェースモジュールは、TCP/IP を含む。しかしながら、他の種類のネットワーク通信プロトコル、たとえば、ポイントツーポイントプロトコル（「PPP」）、サーバメッセージブロック（「SMB」）、シリアルラインインターネットプロトコル（「SLIP」）、トンネリング PPP、AppleTalk などを使用可能であることを理解されたい。

40

3. メモリ 114 およびストレージ 116

【0030】

コンピュータシステム 110 は、メモリ 114 を含むことが可能である。メモリ 114 は、たとえば、プロセッサキャッシュメモリ（プロセッサコア別のキャッシュメモリや、

50

複数のプロセッサコアによって共有されるキャッシュメモリなど)、動的ランダムアクセスメモリ(「DRAM」)、静的ランダムアクセスメモリ(「SRAM」)、または、他の任意の種類、コンピュータデータ、命令、またはプログラムコードを記憶できるメモリ装置を含むことが可能である。コンピュータシステム110はさらに、オプションのストレージ116を含むことが可能である。ストレージ116は、たとえば、1つまたは複数のハードディスクドライブ、フロッピーディスク、フラッシュメモリ、磁気ストレージメディア、CD-ROM、DVD、光ストレージメディア、または他の任意の種類、コンピュータデータ、命令、およびプログラムコードを記憶できるストレージ装置を含むことが可能である。

4. コンピュータシステム110情報

10

【0031】

コンピュータシステム110は、様々なオペレーティングシステムとともに使用されることが可能であり、たとえば、Microsoft(登録商標) Windows(登録商標) 3.X、Windows 95(登録商標)、Windows 98(登録商標)、Windows NT(登録商標)、Windows 2000(登録商標)、Windows XP(登録商標)、Windows CE(登録商標)、Palm Pilot OS、OS/2、Apple(登録商標) MacOS(登録商標)、Mac OS X(登録商標)、Mac OS X Server(登録商標)、ディスクオペレーティングシステム(DOS)、UNIX、Linux(登録商標)、VxWorks(IBM(登録商標) OS/2(登録商標))、Sun OS、Solaris OS、IRIX OSなどのオペレーティングシステムとともに使用されることが可能である。

20

【0032】

一実施形態では、コンピュータシステム110は、パーソナルコンピュータ、ラップトップコンピュータ、Blackberry(登録商標)装置、ポータブルコンピューティング装置、サーバ、コンピュータワークステーション、個々のコンピュータからなるローカルエリアネットワーク、インタラクティブキオスク、携帯情報端末、インタラクティブ無線通信装置、ハンドヘルドコンピュータ、埋め込みコンピューティング装置などである。

【0033】

当業者であれば理解されるように、コンピュータシステム110は、様々なサブルーチン、プロシージャ、定義文、およびマクロを含むことが可能である。上記各モジュールは、典型的には、別々にコンパイルされて、1つの実行可能プログラムにリンクされる。しかしながら、当業者であれば理解されるように、それらのモジュールのうちの選択されたモジュールによって実行されたプロセスは、その他のモジュールのいずれかに任意に再分配されるか、1つのモジュールにまとめられるか、共有可能なダイナミックリンクライブラリのかたちで使用可能にされるか、他の任意の論理様式で分割されることが可能である。

30

B. コンピュータシステム120

【0034】

一実施形態では、第2のコンピュータシステム120は、コンピュータクラスタ100の一部として、ネットワーク102を介して、他のコンピュータシステム110、130と通信している。一実施形態では、コンピュータシステム120は、1つまたは複数のプロセッサ122a~b、メモリ装置124、オプションのストレージ装置126、ならびに、ネットワーク102との通信のためのネットワークインタフェースモジュール(図示せず)を含むパーソナルコンピュータ、ワークステーション、サーバ、またはブレードである。

40

1. プロセッサ112a~b

【0035】

一実施形態では、コンピュータシステム120は、1つまたは複数のプロセッサ122a~bを含む。プロセッサ122a~bは、1つまたは複数の汎用シングルコアマイクロ

50

プロセッサまたはマルチコアマイクロプロセッサであることが可能であり、そのようなプロセッサとして、たとえば、Pentium（登録商標）プロセッサ、Pentium（登録商標）IIプロセッサ、Pentium（登録商標）Proプロセッサ、Pentium（登録商標）IIIプロセッサ、Pentium（登録商標）4プロセッサ、Core Duo（登録商標）プロセッサ、Core 2 Duo（登録商標）プロセッサ、Xeon（登録商標）プロセッサ、Itanium（登録商標）プロセッサ、Pentium（登録商標）Mプロセッサ、x86プロセッサ、Athlon（登録商標）プロセッサ、8051プロセッサ、MIPS（登録商標）プロセッサ、PowerPC（登録商標）プロセッサ、ALPHA（登録商標）プロセッサなどがある。さらに、プロセッサ122a～bは、デジタル信号プロセッサのような任意の専用マイクロプロセッサであることが可能である。コンピュータシステム120内にあるすべてのプロセッサ122a～bの中のプロセッサコア（たとえば、シングルスレッド実行が可能なプロセッサユニット）の総数は、コンピュータシステム120内で使用可能なノードの数に対応する。たとえば、プロセッサ122a～bのそれぞれが、2つのプロセッサコアを有するCore 2 Duo（登録商標）プロセッサであった場合、コンピュータシステム120は、全部で4つのノードを有することになる。各ノードは、シングルスレッドカーネルモジュールのようなプログラムモジュールの1つまたは複数のインスタンスを実行することが可能である。

2. ネットワークインタフェースモジュール

【0036】

コンピュータシステム120はさらに、コンピュータシステム120と他のコンピュータシステム110、130との間の、通信ネットワーク102を介する通信を容易にするネットワークインタフェースモジュール（図示せず）を含むことが可能である。

【0037】

ネットワークインタフェースモジュールは、様々なネットワークプロトコルを使用することが可能である。一実施形態では、ネットワークインタフェースモジュールは、TCP/IPを含む。しかしながら、他の種類のネットワーク通信プロトコル、たとえば、ポイントツーポイントプロトコル（「PPP」）、サーバメッセージブロック（「SMB」）、シリアルラインインターネットプロトコル（「SLIP」）、トンネリングPPP、AppleTalkなども使用可能であることを理解されたい。

3. メモリ124およびストレージ126

【0038】

コンピュータシステム120は、メモリ124を含むことが可能である。メモリ124は、たとえば、プロセッサキャッシュメモリ（プロセッサコア別のキャッシュメモリや、複数のプロセッサコアによって共有されるキャッシュメモリなど）、動的ランダムアクセスメモリ（「DRAM」）、静的ランダムアクセスメモリ（「SRAM」）、または、他の任意の種類、コンピュータデータ、命令、またはプログラムコードを記憶できるメモリ装置を含むことが可能である。コンピュータシステム120はさらに、オプションのストレージ126を含むことが可能である。ストレージ126は、たとえば、1つまたは複数のハードディスクドライブ、フロッピーディスク、フラッシュメモリ、磁気ストレージメディア、CD-ROM、DVD、光ストレージメディア、または他の任意の種類、コンピュータデータ、命令、およびプログラムコードを記憶できるストレージ装置を含むことが可能である。

4. コンピュータシステム120情報

【0039】

コンピュータシステム120は、様々なオペレーティングシステムとともに使用されることが可能であり、たとえば、Microsoft（登録商標）Windows（登録商標）3.X、Windows 95（登録商標）、Windows 98（登録商標）、Windows NT（登録商標）、Windows 2000（登録商標）、Windows XP（登録商標）、Windows CE（登録商標）、Palm Pil

o t OS、OS / 2、Apple (登録商標) MacOS (登録商標)、Mac OS X (登録商標)、Mac OS X Server (登録商標)、ディスクオペレーティングシステム (DOS)、UNIX、Linux (登録商標)、VxWorks (IBM (登録商標) OS / 2 (登録商標))、Sun OS、Solaris OS、IRIX OSなどのオペレーティングシステムとともに使用されることが可能である。

【0040】

一実施形態では、コンピュータシステム120は、パーソナルコンピュータ、ラップトップコンピュータ、Blackberry (登録商標) 装置、ポータブルコンピューティング装置、サーバ、コンピュータワークステーション、個々のコンピュータからなるローカルエリアネットワーク、インタラクティブキオスク、携帯情報端末、インタラクティブ無線通信装置、ハンドヘルドコンピュータ、埋め込みコンピューティング装置などである。

10

【0041】

当業者であれば理解されるように、コンピュータシステム120は、様々なサブルーチン、プロシージャ、定義文、およびマクロを含むことが可能である。上記各モジュールは、典型的には、別々にコンパイルされて、1つの実行可能プログラムにリンクされる。しかしながら、当業者であれば理解されるように、それらのモジュールのうちの選択されたモジュールによって実行されたプロセスは、その他のモジュールのいずれかに任意に再分配されるか、1つのモジュールにまとめられるか、共有可能なダイナミックリンクライブラリのかたちで使用可能にされるか、他の任意の論理様式で分割されることが可能である。

20

C. コンピュータシステム130

【0042】

一実施形態では、第3のコンピュータシステム130は、コンピュータクラスタ100の一部として、ネットワーク102を介して、他のコンピュータシステム110、120と通信している。一実施形態では、コンピュータシステム130は、1つまたは複数のプロセッサ132、メモリ装置134、オプションのストレージ装置136、ならびに、ネットワーク102との通信のためのネットワークインタフェースモジュール(図示せず)を含むパーソナルコンピュータ、ワークステーション、サーバ、またはブレードである。

1. プロセッサ112a~b

30

【0043】

一実施形態では、コンピュータシステム130は、プロセッサ132を含む。プロセッサ132は、汎用シングルコアマイクロプロセッサまたはマルチコアマイクロプロセッサであることが可能であり、そのようなプロセッサとして、たとえば、Pentium (登録商標) プロセッサ、Pentium (登録商標) II プロセッサ、Pentium (登録商標) Pro プロセッサ、Pentium (登録商標) III プロセッサ、Pentium (登録商標) 4 プロセッサ、Core Duo (登録商標) プロセッサ、Core 2 Duo (登録商標) プロセッサ、Xeon (登録商標) プロセッサ、Itanium (登録商標) プロセッサ、Pentium (登録商標) M プロセッサ、x86 プロセッサ、Athlon (登録商標) プロセッサ、8051 プロセッサ、MIPS (登録商標) プロセッサ、PowerPC (登録商標) プロセッサ、ALPHA (登録商標) プロセッサなどがある。さらに、プロセッサ132は、デジタル信号プロセッサのような任意の専用マイクロプロセッサであることが可能である。コンピュータシステム130内にあるプロセッサ132の中のプロセッサコア(たとえば、シングルスレッド実行が可能なプロセッサユニット)の総数は、コンピュータシステム130内で使用可能なノードの数に対応する。たとえば、プロセッサ132が、2つのプロセッサコアを有するCore 2 Duo (登録商標) プロセッサであった場合、コンピュータシステム130は、2つのノードを有することになる。各ノードは、シングルスレッドカーネルモジュールのようなプログラムモジュールの1つまたは複数のインスタンスを実行することが可能である。

40

50

2. ネットワークインタフェースモジュール

【0044】

コンピュータシステム130はさらに、コンピュータシステム130と他のコンピュータシステム110、120との間の、通信ネットワーク102を介する通信を容易にするネットワークインタフェースモジュール(図示せず)を含むことが可能である。

【0045】

ネットワークインタフェースモジュールは、様々なネットワークプロトコルを使用することが可能である。一実施形態では、ネットワークインタフェースモジュールは、TCP/IPを含む。しかしながら、他の種類のネットワーク通信プロトコル、たとえば、ポイントツーポイントプロトコル(「PPP」)、サーバメッセージブロック(「SMB」)、シリアルラインインターネットプロトコル(「SLIP」)、トンネリングPPP、AppleTalkなども使用可能であることを理解されたい。

10

3. メモリ134およびストレージ136

【0046】

コンピュータシステム130は、メモリ134を含むことが可能である。メモリ134は、たとえば、プロセッサキャッシュメモリ(プロセッサコア別のキャッシュメモリや、複数のプロセッサコアによって共有されるキャッシュメモリなど)、動的ランダムアクセスメモリ(「DRAM」)、静的ランダムアクセスメモリ(「SRAM」)、または、他の任意の種類の、コンピュータデータ、命令、またはプログラムコードを記憶できるメモリ装置を含むことが可能である。コンピュータシステム130はさらに、オプションのストレージ136を含むことが可能である。ストレージ136は、たとえば、1つまたは複数のハードディスクドライブ、フロッピーディスク、フラッシュメモリ、磁気ストレージメディア、CD-ROM、DVD、光ストレージメディア、または他の任意の種類の、コンピュータデータ、命令、およびプログラムコードを記憶できるストレージ装置を含むことが可能である。

20

4. コンピュータシステム130情報

【0047】

コンピュータシステム130は、様々なオペレーティングシステムとともに使用されることが可能であり、たとえば、Microsoft(登録商標) Windows(登録商標) 3.X、Windows 95(登録商標)、Windows 98(登録商標)、Windows NT(登録商標)、Windows 2000(登録商標)、Windows XP(登録商標)、Windows CE(登録商標)、Palm Pilot OS、OS/2、Apple(登録商標) MacOS(登録商標)、Mac OS X(登録商標)、Mac OS X Server(登録商標)、ディスクオペレーティングシステム(DOS)、UNIX、Linux(登録商標)、VxWorks(IBM(登録商標) OS/2(登録商標))、Sun OS、Solaris OS、IRIX OSなどのオペレーティングシステムとともに使用されることが可能である。

30

【0048】

一実施形態では、コンピュータシステム130は、パーソナルコンピュータ、ラップトップコンピュータ、Blackberry(登録商標)装置、ポータブルコンピューティング装置、サーバ、コンピュータワークステーション、個々のコンピュータからなるローカルエリアネットワーク、インタラクティブキオスク、携帯情報端末、インタラクティブ無線通信装置、ハンドヘルドコンピュータ、埋め込みコンピューティング装置などである。

40

【0049】

当業者であれば理解されるように、コンピュータシステム130は、様々なサブルーチン、プロシージャ、定義文、およびマクロを含むことが可能である。上記各モジュールは、典型的には、別々にコンパイルされて、1つの実行可能プログラムにリンクされる。しかしながら、当業者であれば理解されるように、それらのモジュールのうちの選択されたモジュールによって実行されたプロセスは、その他のモジュールのいずれかに任意に再分

50

配されるか、1つのモジュールにまとめられるか、共有可能なダイナミックリンクライブラリのかたちで使用可能にされるか、他の任意の論理様式で分割されることが可能である。

E. 通信ネットワーク 102

【0050】

一実施形態では、コンピュータシステム 110、120、130は、通信ネットワーク 102を介して、互いに通信している。

【0051】

通信ネットワーク 102は、1つまたは複数の、任意の種類、電子的に接続されたコンピュータのグループを含むことが可能であり、そのようなグループとして、たとえば、仮想プライベートネットワーク、パブリックインターネット、プライベートインターネット、セキュアインターネット、プライベートネットワーク、パブリックネットワーク、付加価値ネットワーク、有線ネットワーク、無線ネットワーク、イントラネットなどのネットワークがある。さらに、ネットワークとの接続性は、たとえば、モデム、イーサネット（登録商標）（IEEE 802.3）、ギガビットイーサネット（登録商標）、10ギガビットイーサネット（登録商標）、トークンリング（IEEE 802.5）、ファイバ分散データリンクインタフェース（FDDI: Fiber Distributed Data Link Interface）、フレームリレー、インフィニバンド（InfiniBand）、ミリネット（Myrinet）、非同期転送モード（ATM）、または別のインタフェースであることが可能である。通信ネットワーク 102は、コンピュータシステム 110、120、130と、たとえば、モデムによって、またはそれらのシステムのそれぞれにあるネットワークインタフェースカードによって、接続可能である。

【0052】

さらに、同じ、あるいは別々の通信ネットワーク 102を使用して、第1のコンピュータシステム 110と第2のコンピュータシステム 120との間の通信、第1のコンピュータシステム 110と第3のコンピュータシステム 130との間の通信、および第2のコンピュータシステム 120と第3のコンピュータシステム 130との間の通信を容易にすることが可能である。

III. ソフトウェアモジュール

【0053】

図1および図2に示されるように、クラスタシステム 100の一実施形態は、第1のクラスタノードモジュール 204aと通信することによって複数のカーネルモジュール 206a~eにアクセスすることが可能なユーザインタフェースモジュール 202を含む。ユーザインタフェースモジュールは、（たとえば、実行中は）メモリ 114、124、134に記憶されることが可能であり、かつ/または、ストレージ装置 116、126、136に記憶されることが可能である。第1のクラスタノードモジュール 204aは、他のクラスタノードモジュール 204b~eのそれぞれと通信している。カーネルモジュール 206a~eは、それらが実行される1つまたは複数のコンピュータシステムのメモリに常駐することが可能である。たとえば、第1のコンピュータシステム 110のメモリ 114は、カーネルモジュール 206a~bのインスタンスを記憶することが可能であり、第2のコンピュータシステム 120のメモリ 124は、カーネルモジュール 206c~dのインスタンスを記憶することが可能であり、第3のコンピュータシステム 130のメモリ 134は、カーネルモジュール 206eのインスタンスを記憶することが可能である。カーネルモジュール 206a~eは、シングルスレッドプログラムコードを含み、それぞれが、プロセッサ 112a、112b、122a、122b、132のうちの1つに関連付けられている。コンピュータシステム 110、120、130のうちの1つまたは複数、またはリモートコンピュータシステムに記憶されたクラスタ構成モジュールが、たとえば、クラスタノードモジュール 204a~eとの通信を確立することが可能である。一実施形態では、クラスタ構成モジュール 208とクラスタノードモジュール 204a~eとの間の通信によって、クラスタノードモジュール 204a~eが初期化されて、コンピュ

ータクラスタ100のクラスタコンピューティングサポートが提供される。

A. クラスタノードモジュール204

【0054】

一実施形態では、クラスタノードモジュール204a~eは、コンピュータクラスタ100で実行されている様々なカーネルモジュール206a~e、たとえば、Mathematicaカーネルなど、が互いに通信するための手段を提供する。クラスタノードモジュール204は、スーパーコンピュータおよびクラスタの、いくつかのインストレーションで用いられている、メッセージパッシングインタフェース(「MPI」)として知られるアプリケーションプログラミングインタフェース(「API」)の少なくとも一部を含むことが可能である。クラスタノードモジュール204a~e間の接続(たとえば、図2に示された矢印)からなるネットワークは、たとえば、イーサネット(登録商標)上のTCP/IPのような通信ネットワーク102を用いて実装されることが可能であるが、これらの接続は、他の任意の種類ネットワーク上またはローカルコンピュータバス上でも行われることが可能である。

10

【0055】

クラスタノードモジュール204は、アプリケーション別のツールキットか、たとえば、MathematicaのMathLink、Add-Ons、またはパケットなどのインタフェースを用いてアプリケーションと対話することが可能である。Mathematicaカーネルを、Mathematicaフロントエンドとして知られるユーザインタフェースまたは他のMathematicaカーネルに接続するために通常使用されるMathLinkは、これらのエンティティのうち任意のエンティティ間でメッセージ、コマンド、またはデータを含む「パケット」を送信する双方向プロトコルである。MathLinkは、コマンドまたはスレッドの実行中は、直接クラスタコンピューティングのような、Mathematicaカーネル間の同時通信を許可しない。MathLinkはさらに、複数の同時ネットワーク接続を実行するには設計されていない。実施形態によっては、クラスタノードモジュール204は、同じコンピュータにあるエンティティ同士の接続に、アプリケーション別のツールキット、たとえば、MathLinkなどを用いることが可能である。

20

【0056】

クラスタまたは他の並列コンピュータに対するプロシージャまたはアクションに関して言えば、すべてのアクションが順番に実行されるとは限らず、順番に実行されなければならないわけでもない。たとえば、古典的な「チューリングマシン」モデルのシングルプロセッサコードとは対照的に、並列コードは、その複数のコピーがクラスタ全体で実行され、典型的には、各プロセッサ(または「プロセッサエレメント」または「コア」)に対して1つのコピーが実行される。そのような並列コードは、同じコードの別々のインスタンスが互いに通信し、共同作業を行い、作業内容を調整し合うことが可能であるように書かれる。これらのコードの複数のインスタンスは、同時に並列に実行されることが可能である。

30

【0057】

コードインスタンスの数が整数Nであれば、コード実行の各インスタンスは、0からN-1までのラベルが付けられることが可能である。たとえば、コンピュータクラスタは、それぞれがプロセッサを含む、N個の接続されたコンピュータを含むことが可能である。第1のコンピュータは、プロセッサ0で実行されるカーネルモジュール0に、クラスタノードモジュール0が接続されている。次のコンピュータは、プロセッサ1で実行されるカーネルモジュール1に、クラスタノードモジュール1が接続されており、接続されたN個のコンピュータのそれぞれについて同様である。それらのプロシージャのいくつかのステップは共同作業であり、いくつかのステップは単独作業である。これらのエンティティは、必ずしもロックステップにはなっていないが、必ず、初期化、主ループ動作(たとえば、クラスタノードモジュール動作)、およびシャットダウンのパターンに従う。

40

【0058】

50

これに対し、grid Mathematicaソフトウェアパッケージの一部として与えられる並列コンピューティングツールキット（PCT：parallel computing toolkit）は、別々のノードで実行されている、同じコードのインスタンスがインスタンス同士で通信し、共同作業を行い、作業内容を調整し合うための手段を提供しない。PCTは、Mathematicaカーネルを、本明細書で開示されているいくつかの実施形態で実現されるピアツーピア関係ではなく、マスタスレーブ関係で接続するコマンドを提供する。ピアツーピアノードアーキテクチャを有するコンピュータクラスタで実行される計算は、マスタスレーブノードアーキテクチャを有するグリッドコンピュータで実行される同等の計算より、効率が良く、設計しやすく、かつ/または、信頼性が高いことが可能である。さらに、マスタスレーブノードアーキテクチャを用いるシステムでは、計算の性質によっては、プログラマが、マルチノード処理能力を利用できない場合がある。

10

【0059】

図3は、MPI呼び出しおよび高度MPI関数を実装するクラスタノードモジュール204の一実施形態を示す。図3に示された実施形態では、クラスタノードモジュール204は、MPIモジュール302、高度機能モジュール304、受信済みメッセージキュー306、およびメッセージ受信キュー308を含む。

- 1. MPIモジュール302

【0060】

一実施形態では、クラスタノードモジュール204は、MPIモジュール302を含む。MPIモジュール302は、少なくとも5種類のMPI命令またはMPI呼び出しのうちの1つまたは複数に対応するプログラムコードを含むことが可能である。MPIモジュール302によって実装されることが可能な、選択された定数、命令、および/または呼び出しは、以下のとおりである。

20

MPI定数

【0061】

ノード識別子は、メッセージをノードへ送信したり、メッセージをノードから受信したりするために使用される。MPIでは、これは、0から始まる一意の整数（\$IdProc）を各ノードに割り当てることによって行われる。総数（\$NProc）がわかっているならば、このデータは、任意の測定可能なエンティティをプログラムで分割することを可能にする。

30

【表1】

定数	説明
\$IdProc	現在のプロセッサの識別番号。
\$NProc	現在のクラスタにあるプロセッサの数。
\$mpiCommWorld	クラスタ全体のコミュニケータワールド（後述のMPIコミュニケータルーチンを参照）。
mpiCommWorld	高級ルーチンに対するデフォルトのコミュニケータワールド。

40

表A
基本MPI呼び出し

【0062】

一実施形態では、MPIモジュール302は、たとえば、他の言語（CやFortranなど）でよく使用されるMPI呼び出しをマッピングする比較的低位のルーチン群など、基本MPI呼び出しを含むことが可能であり、それによって、そのような呼び出しをMathematicaユーザインタフェース204から直接使用できるようにすることが可能になる。実施形態によっては、基本MPI呼び出しは、データ、方程式、公式、および/または他の数式を送信する呼び出しを含む。

50

【 0 0 6 3 】

以下の最も基本的な M P I 呼び出しを用いれば、1つのノードから別のノードへ数式を単純に送信することが可能である。1つのノードが、数式を送信することを呼び出し、他のノードが、送信された数式を受信する、対応するルーチン呼び出すことが可能である。メッセージが送信側ノードを離れても、受信側がまだ m p i R e c v を呼び出していない可能性があるため、m p i S e n d の完了は、メッセージが受信されたことの確認にはならない。

【表 2】

呼び出し	説明
mpiSend[expr, target, comm, tag]	コミュニケーターワールド「comm」において、ID「target」を有するノードへ数式「expr」を送信し、その数式がこのカーネルを離れるまで待機する。
mpiRecv [expr, target, comm, tag]	コミュニケーターワールド「comm」において、ID「target」を有するノードから数式「expr」を受信し、その数式が到着するまで待機する。
mpiSendRecv[sendexpr, dest, recvexpr, source, comm]	コミュニケーターワールド「comm」において、ID「target」を有するノードへ数式「sendexpr」を送信することと、ID「source」を有するノードから数式「recvexpr」を受信することとを同時に行い、両方の操作が戻るまで待機する。

10

20

表 B

非同期 M P I 呼び出し

【 0 0 6 4 】

非同期呼び出しは、複数の通信が同時進行している間にカーネルが作業を行うことを可能にする。また、待機中に、別のノードがデータをまだ送信または受信できないようにして、1つのカーネルが作業を続行できるようにすることが可能である。

【表 3】

呼び出し	説明
mpiISend[expr, target, comm, tag, req]	コミュニケーターワールド「comm」において、ID「target」を有するプロセッサへ数式「expr」を送信し、ただちに戻る。mpiTest [req] が True を返すまでに、mpiTest [req] の呼び出しとバランスをとることが可能である。
mpiIRecv[expr, target, comm, tag, req]	コミュニケーターワールド「comm」において、ID「target」を有するプロセッサから数式「expr」を受信し、ただちに戻る。mpiTest [req] が True を返すまでに、mpiTest [req] の呼び出しとバランスをとることが可能である。「expr」は、mpiTest [req] が True を返すまで、アクセスに対して安全ではない。
mpiTest[req]	mpiISendおよびmpiIRecvの非同期動作を完了する。
mpWait[req]	mpiTestがTrueを返すまで、mpiTestを呼び出す。
mpiWaitall[reglist]	「reglist」のどの要素に対してもすべてmpiWaitを呼び出す。
mpiWaitany[reqlist]	「realist」のいずれかの要素がTrueを返すまで「realist」の各要素に対してmpiTestを呼び出す。

30

40

50

表 C

【 0 0 6 5 】

`mpiISend []` コマンドは、カーネルモジュール 206 (たとえば、`Mathematica` カーネル) の中から呼び出されることが可能である。`mpiISend []` コマンドは、ペイロードとして送信される `Mathematica` 数式と、その数式の送信先とを含むパケットを作成する。このパケット自体は、そのローカルのクラスタノードモジュールだけを宛先とされる。このパケットは、そのローカルのクラスタノードモジュールによって受信された後に復号され、そのペイロードが、パケットで指定されたクラスタノードモジュールに転送される。

【 0 0 6 6 】

`mpiIRecv []` コマンドも、カーネルモジュール 206 の中から呼び出されることが可能である。`mpiIRecv []` コマンドは、数式を受け取ることが期待する場所、およびこの数式の出所として期待されるプロセッサを指定するパケットを作成する。このパケットは、そのローカルのクラスタノードモジュールによって受信された後に復号され、そのコンテンツが、メッセージ受信キュー (MRQ) 308 (図 3) に記憶される。

【 0 0 6 7 】

`mpiTest []` コマンドは、カーネルモジュール 206 の中から呼び出されることが可能である。`mpiTest []` コマンドは、どのメッセージを完了のためにテストするかを指定するパケットを作成し、その後、評価すべき数式の応答を待つ。このパケットは、そのカーネルモジュールが関連付けられたクラスタノードモジュール 204 によって受信された後に復号され、そのメッセージ指定子を用いて、その受信済みメッセージキュー (RMQ) 306 の中で、完了されたものとしてリストされている、一致する数式が検索される。そのような完了された数式が見つかった場合は、その数式が、`mpiTest []` 内の応答の一部として、そのローカルのカーネルモジュールに送信される。カーネルモジュールは、この応答の数式を受け取って評価し、これによって、カーネルモジュールの変数が、必要に応じて更新される。

【 0 0 6 8 】

他の MPI 呼び出しは、基本呼び出し `mpiISend`、`mpiIRecv`、および `mpiTest` の上に構築される。たとえば、`mpiBcast` (ブロードキャスト) は、他のプロセッサが `Recv` を実行している間にブロードキャストプロセッサから他のすべてのプロセッサへ情報を送信する命令を作成する。同様に、ツールキットの高級呼び出しは、MPI 呼び出しの集合体の最上部に構築されることが可能である。

集団 MPI 呼び出し

【 0 0 6 9 】

一実施形態では、MPI モジュール 302 は、集団 MPI 呼び出し (たとえば、ノード間の基本マルチノードデータ移動を与える呼び出し) を実施するプログラムコードを含むことが可能である。集団 MPI 呼び出しは、たとえば、ブロードキャスト、収集、転置、および他のベクトル操作や行列操作を含むことが可能である。集団呼び出しはまた、ノードのグループ間で数式を送信するために一般に使用されているメカニズムを提供することも可能である。

10

20

30

40

【表 4】

呼び出し	説明
mpiBcast[expr, root, comm]	コミュニケーターワールド「comm」において、「root」プロセッサから他のすべてのプロセッサへの「expr」のブロードキャストを実行する。数式は、「root」プロセッサによって供給されることが期待され、他のすべてのプロセッサは、到着する数式によって「expr」が上書きされることを期待する。
mpiGather[sendexpr, recvexpr, root, comm]	コミュニケーター「comm」内の（「root」を含む）すべてのプロセッサが、それぞれの数式を「sendexpr」に入れて「root」プロセッサへ送信し、「root」プロセッサは、これらの数式の、「comm」に従う順番のリストを、「recvexpr」に生成する。「root」以外のプロセッサでは、「recvexpr」は無視される。
mpiAllgather[sendexpr, recvexpr, comm]	コミュニケーター「comm」内のすべてのプロセッサが、それぞれの数式を「sendexpr」に入れて送信し、これらの数式は、「comm」内のすべてのプロセッサの「recvexpr」において、これらの数式の、「comm」に従う順番のリストにまとめられる。
mpiScatter[sendexpr, recvexpr, root, comm]	プロセッサ「root」が、（可能であれば）「sendexpr」内のリストを均等に分割し、分割された各部分を、コミュニケーターワールド「comm」内の（「root」を含む）すべてのプロセッサの「recvexpr」に、「comm」の順番およびサイズに従って配置する。
mpiAlltoall[sendexpr, recvexpr, comm]	各プロセッサが、リストの均等部分を「sendexpr」に入れて、コミュニケーターワールド「comm」内の他のすべてのプロセッサに送信し、各プロセッサは、他のすべてのプロセッサから集めたものを、「comm」に従う順番で整理する。

10

20

表 D

一実施形態では、MPI モジュール 302 は、多数のノードにわたって記憶されているデータの並列和および他のリダクション操作を実施するプログラムコードを含む。MPI モジュール 302 はまた、単純な並列入出力呼び出し（たとえば、複数のノードにあるオブジェクトをクラスタシステム 200 がロードおよび記憶することを可能にする呼び出し）を実施するプログラムコードを含むことが可能である。

30

【表 5】

呼び出し	説明
mpiReduce[sendexpr, recvexpr, operation, root, comm]	「sendexpr」内のリストにある要素ごとに、コミュニケーターワールド「comm」内のすべてのプロセッサにおいて数式間の集団リダクション操作を実行し、ID「root」を有するプロセッサの「recvexpr」に結果のリストを返す。
mpiAllreduce[sendexpr, recvexpr, operation, comm]	「sendexpr」内のリストにある要素ごとに、コミュニケーターワールド「comm」内のすべてのプロセッサにおいて数式間の集団リダクション操作を実行し、すべてのプロセッサの「recvexpr」に結果のリストを返す。
mpiReduceScatter[sendexpr, recvexpr, operation, comm]	「sendexpr」内のリストにある要素ごとに、コミュニケーターワールド「comm」内のプロセッサにおいて数式間の集団リダクション操作を実行し、結果のリストを各プロセッサの「recvexpr」への部分に分割する。

40

表 E

【0070】

以下の追加の集団呼び出しは、データを並列にリダクションする操作を実行する。操作の引数は、下記の定数のいずれかであることが可能である。

50

【表 6】

定数	説明
mpiSum	リダクション呼び出しにおいて、別々のプロセッサにあるすべての要素が合計されるように指定する。
mpiMax	リダクション呼び出しにおいて、別々のプロセッサにあるすべての要素のうちの最大値が選択されるように指定する。
mpiMin	リダクション呼び出しにおいて、別々のプロセッサにあるすべての要素のうちの最小値が選択されるように指定する。

10

表 F

M P I コミュニケータ呼び出し

【 0 0 7 1 】

一実施形態では、M P I モジュール 3 0 2 は、コミュニケータワールド呼び出し（たとえば、ノードのサブセットがあたかもサブクラスタであるかのように動作することを可能にする呼び出し）を実施するプログラムコードを含む。コミュニケータは、ノードのグループをユーザ定義サブセットに編成する。mpiCommSplit[] によって返されるコミュニケータ値は、他の M P I 呼び出しで mpiCommWorld の代わりに使用可能である。

【表 7】

20

呼び出し	説明
mpiCommSize[comm]	コミュニケータ「comm」内のプロセッサの数を返す。
mpiCommRank[comm]	このプロセッサの、コミュニケータ「comm」内でのランクを返す。
mpiCommDup[comm]	コミュニケータ「comm」の重複しているコミュニケータを返す。
mpiCommSplit[comm, color, key]	それぞれ「color」で識別された複数の互いに素なサブセットに新しいコミュニケータを作成する。各サブセット内のソート順は、第1にキーにより、第2に前のコミュニケータでの順番による。どの新しいコミュニケータにも参加することが意図されていないプロセッサは、そのことを、定数 mpiUndefined を渡すことによって示す。各呼び出し側プロセッサに対して、対応するコミュニケータが返される。
mpiCommMap[comm] mpiCommMap[comm, target]	\$ mpiCommWorld に従ってインデックス付けされたプロセッサに、コミュニケータ「comm」のマッピングを返す。第2の引数を追加すると、コミュニケータ「comm」において ID「target」を有するプロセッサの ID だけが返される。
mpiCommFree[comm]	コミュニケータ「comm」を解放する。

30

40

表 G

他の M P I サポート呼び出し

【 0 0 7 2 】

共通機能を提供する他の呼び出しとして、以下のものがある。

【表 8】

呼び出し	説明
<code>mpiWtime[]</code>	過去のある固定時刻からのウォールドック時間を提供する。この時間が、すべてのプロセッサにおいて同じ読みになる保証はない。
<code>mpWtick[]</code>	<code>mpiWtime []</code> の時間分解能を返す。
<code>MaxByElement[in]</code>	リスト「in」の各リストのすべてのn番目の要素について、 <code>Max []</code> に従って最大値を選択し、その結果を1つのリストとして返す。 <code>mpiMax</code> リダクション操作において用いられる。
<code>MinByElement[in]</code>	リスト「in」の各リストのすべてのn番目の要素について、 <code>Min []</code> に従って最小値を選択し、その結果を1つのリストとして返す。 <code>mpiMin</code> リダクション操作において用いられる。

10

表 H

2 . 高度関数モジュール 3 0 4

【 0 0 7 3 】

一実施形態では、クラスタノードモジュール 2 0 4 は、高度関数モジュール 3 0 4 を含む。高度関数モジュール 3 0 4 は、MPIモジュール 3 0 2 によって実装される MPI 命令および MPI 呼び出しを用いて実行することが不便または非現実的である関数のツールキットを提供するプログラムコードを含むことが可能である。高度関数モジュール 3 0 4 は、高度関数を実装するにあたり、MPIモジュール 3 0 2 によって実装される呼び出しおよび命令に、少なくとも部分的に依存することが可能である。一実施形態では、高度関数モジュール 3 0 4 は、指示または関数のカスタムセットを含む。一代替実施形態では、高度関数モジュール 3 0 4 は、標準の `Mathematica` 言語をインタセプトし、これを、クラスタ実行に最適化された1つまたは複数の関数に変換する。そのような実施形態は、`Mathematica` 関数になじんだユーザにとってはより使いやすいものであり得るが、プログラムデバッグプロセスが複雑になる可能性もある。高度関数モジュール 3 0 4 によって実装されるいくつかの関数は、並列コンピューティングを用いるセットアップが困難または複雑である操作を簡略化することが可能である。高度関数モジュール 3 0 4 によって実装可能な、そのような関数のいくつかの例を、以下に示す。

20

30

【 0 0 7 4 】

以下で説明される呼び出しは、MPI呼び出しの上に構築され、`Mathematica` 機能の、よく使用される通信パターンまたは並列バージョンを提供する。これらは、特に断らない限り、デフォルトが `$mpiCommWorld` であるコミュニケータ `mpiCommWorld` において実行されるが、実行時に、有効なコミュニケータに変更可能である。

一般的な分割統治並列評価

【 0 0 7 5 】

一実施形態では、高度関数モジュール 3 0 4 は、基本的な並列化に備える関数、たとえば、多数のノードに記憶されている多数のデータ要素またはデータ入力に対して同じ操作を行うルーチンなどを含む。これらの関数は、並列化された `for` ループなどに匹敵する。以下の呼び出しは、一般的なタスクの単純な並列化を扱う。呼び出しの説明において、「`expr`」は数式を意味し、「`loopspec`」は、数式の評価方法を決定するルールのセットを意味する。実施形態によっては、高度関数モジュール 3 0 4 は、`{var, count}`、`{var, start, stop}`、および `{var, start, stop, increment}` を含む、少なくとも3つの形式の `loopspec` をサポートし、`{var, count}` の場合、呼び出しは、1から整数「`count`」までの変数「`var`」を反復し、`{var, start, stop}` の場合、呼び出しは、「`start`」から「`stop`」までの変数「`var`」（すべて整数）を反復し、`{var, start, stop, increment}` の場合、呼び出しは、反復のたびに「`in`

40

50

crement」が追加される「start」から、「stop」を超えるまでの変数「var」（非整数であってよい）を反復する。

【表9】

呼び出し	説明
ParallelDo[expr, loopspec]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって「expr」を評価すること以外は、Do []と同様。「expr」の評価方法に関するルールは、Do []の場合と同様に、「loopspec」で指定される。
ParallelFunctionToList[f, count] ParallelFunctionToList[f, count, root]	1から「count」までの関数 f [i] を、クラスタ全体にわたって評価し、これらの結果をリストのかたちで返す。第3の引数は、このリストが、IDが「root」であるプロセッサに集められるようにする。
ParallelTable[expr, loopspec] ParallelTable[expr, loopspec, root]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって「expr」を評価し、ローカルに評価された部分を返すこと以外は、Table []と同様。第3の引数は、このテーブルが、IDが「root」であるプロセッサに集められるようにする。
ParallelFunction[f, inputs, root]	プロセッサ「root」からクラスタ全体に分散され、再度「root」に集められた「inputs」のサブセットに関して f を評価すること以外は、f [inputs]と同様。
ParallelNintegrate[expr, loopspec] ParallelNintegrate[expr, loopspec, digits]	クラスタ内のプロセッサの数に分割されたドメインの全体にわたって「expr」の数値積分を評価し、その合計を返すこと以外は、Nintegrate []と同様。第3の引数は、少なくともその複数の「digits」（桁数）の精度で各数値積分が実行されるようにする。

10

20

表 I

保護セル管理

【0076】

一実施形態では、高度関数モジュール304は、保護セル操作に備える関数、たとえば、（1D、2D、および/または3Dに最適化された）任意の数の次元のローカル配列のエッジを維持するために最近隣通信を実行するルーチンなどを含む。典型的には、問題の空間は、パーティションに分割されている。しかしながら、各パーティションの隣接するエッジ同士は、相互作用する可能性が非常に高いため、両方のエッジに対して、「保護セル」が、隣接するデータの代わりに挿入される。したがって、あるプロセッサから見える空間は、そのプロセッサが担当する実際の空間より要素2つ分だけ広い。EdgeCellは、これらの保護セルの維持を支援する。

30

【表10】

呼び出し	説明
EdgeCell[list]	「list」の第2の要素を左側のプロセッサの最後の要素にコピーし、「list」の第2から最後への要素を右側のプロセッサの第1の要素にコピーし、両隣から同じものを同時に受け取る。

40

表 J

行列およびベクトルの操作

【0077】

高度関数モジュール304は、線形代数演算に備える関数、たとえば、多数のノードに分割された構造に対する、基本線形代数の並列化バージョンなどを含むことも可能である

50

。そのような線形代数演算では、行列およびベクトルの操作や、たとえば、行列式、トレースなどの他の演算を実行するために、必要に応じてデータを再編成することが可能である。行列は、クラスタ全体にわたる各プロセッサに分割されて記憶される。以下の呼び出しは、これらの行列を一般的な様式で操作する。

【表 1 1】

呼び出し	説明
ParallelTranspose[matrix]	1つのプロセッサだけに対してではなく、実際にはクラスタ全体にわたって表されている「matrix」を転置すること以外は、Transpose [] と同様。転置された「matrix」の、そのプロセッサに対応する部分を返す。
ParallelProduct[matrix, vector]	「matrix」がクラスタ全体にわたって表されていること以外は、1つのプロセッサに対する場合と同様に、「matrix」と「vector」との積を評価する。
ParallelDimensions[matrix]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって「matrix」が表されていること以外は、Dimensions [] と同様。各次元のリストを返す。
ParallelTr[matrix]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって「matrix」が表されていること以外は、Tr [] と同様。この行列のトレースを返す。
ParallelIdentity[rank]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって「matrix」が表されていること以外は、Identity [] と同様に、新しい単位行列を生成する。新しい行列の、このプロセッサに対応する部分を返す。
ParallelOuter[f, vector1, vector2]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって表されている「matrix」が答えになること以外は、Outer [f, vector1, vector2] と同様。新しい行列の、このプロセッサに対応する部分を返す。
ParallelInverse[matrix]	1つのプロセッサだけに対してではなく、クラスタ全体にわたって「matrix」が表されていること以外は、Inverse [] と同様。この行列の逆行列を返す。

10

20

30

表 K

要素管理

【0078】

一実施形態では、高度関数モジュール304は、要素管理操作を含む。たとえば、空間内の複数のノードに切り分けられた要素または小片の大規模ピンが、ルールまたは条件（たとえば、それらの空間座標）に基づいてノード間を移動しなければならない場合がある。そのような操作によって、データはノード間を移動する。分割統治アプローチに加えて、要素のリストを、任意の様式で分割することも可能である。これは、要素を複数のプロセッサに編成またはソートしなければならない場合に有用である。たとえば、システムの各小片が、1つのプロセッサの空間から別のプロセッサの空間へ流れ出る場合があるので、それらのデータは、定期的に再分配される必要がある。

40

【表 1 2】

呼び出し	説明
ElementManage[list, switch]	「list」の各要素に関して関数「switch」[] が評価された結果に従って、「list」のどの要素をどのプロセッサに送信するかを選択する。「switch」が関数であれば、「switch」[] は、その要素が送信される先のプロセッサのIDを返す。「switch」が整数であれば、この呼び出しは、各要素がそれ自体リストであって、その第1の要素が、0から渡された引数までの範囲の数であると仮定している。この呼び出しは、任意のプロセッサから、そのプロセッサに対して選択された「switch」である、それらの要素のリストを返す。
ElementManage[list]	「list」の各要素は、2つの要素からなるリストであることが可能であり、第1の要素は、その元の要素が送信される先のプロセッサのIDであり、第2の要素は、送信される任意のデータである。この呼び出しは、ありとあらゆるプロセッサから、第1の要素が、リストにあるそのプロセッサIDである、それらのリスト要素を返す。この呼び出しは、ElementManage [] の2引数バージョンによって内部的に使用される。

10

表 L

フーリエ変換

20

【0079】

一実施形態では、高度関数モジュール304は、大規模並列高速フーリエ変換（「FFT」）を実施するプログラムコードを含む。たとえば、そのような関数は、1つのノードではなく多数のノードに記憶されている大量のデータに対して、1次元、2次元、および/または3次元のFFTを実行することが可能である。非常に大きな配列のフーリエ変換は、メモリ要件の問題にとどまらず、管理が困難である可能性がある。フーリエ変換を並列化することにより、クラスタ全体におけるすべての空きメモリを活用することが可能になり、1つのプロセッサが単独で行うことができないと考えられる問題サイズの操作を行うことが可能になる。

【表 1 3】

30

呼び出し	説明
ParallelFourier[list]	前述の行列の場合と同様に、「list」が、クラスタ全体にわたって表されている2次元または3次元のリストであること以外は、Fourier [] と同様。フーリエ変換された配列の、そのプロセッサに対応する部分を返す。

表 M

並列ディスク I/O

【0080】

一実施形態では、高度関数モジュール304は、並列ディスクの入力呼び出しおよび出力呼び出しを含む。たとえば、データがクラスタ全体にわたって均一に分散しているようなクラスタにデータを読み込ませたり、そのようなクラスタからデータを読み出したりすることが必要になる場合がある。以下の表の各呼び出しは、データを1つまたは複数のプロセッサからストレージに保存したり、ストレージからデータを取り出したりすることを可能にする。

40

【表 1 4】

呼び出し	説明	
ParallelPut[expr, filename] ParallelPut[expr, filename root] ParallelPut[expr, filename, root, comm]	プロセッサ0において、「filename」という名前のファイルに「expr」を順番に書き込む。第3の引数は、IDが「root」であるプロセッサにおいて、そのファイルが書き込まれることを指定する。第4の引数は、コミュニケーターワールド「comm」を使用する。	
ParallelGet[filename] ParallelGet[filename, root] ParallelGet[filename, root, comm]	クラスタ上の各プロセッサに分割されているプロセッサ0において、「filename」という名前のファイルからデータを読み出して返す。第2の引数は、IDが「root」であるプロセッサにおいて、そのファイルが読み出されることを指定する。第3の引数は、コミュニケーターワールド「comm」を使用する。	10
ParallelBinaryPut[expr, type, filename] ParallelBinaryPut[expr, filename, root] ParallelBinaryPut[expr, filename, root, comm]	プロセッサ0において、「filename」という名前の、「type」バイナリフォーマットのファイルに「expr」を順番に書き込む。第4の引数は、IDが「root」であるプロセッサにおいて、そのファイルが書き込まれることを指定する。第5の引数は、コミュニケーターワールド「comm」を使用する。	20
ParallelBinaryGet[type, filename] ParallelBinaryGet[type, filename, root] ParallelBinaryGet[type, filename, root, comm]	クラスタ上の各プロセッサに分割されているプロセッサ0において、「filename」という名前のファイルから、「type」バイナリフォーマットのデータを読み出して返す。第3の引数は、IDが「root」であるプロセッサにおいて、そのファイルが読み出されることを指定する。第4の引数は、コミュニケーターワールド「comm」を使用する。	
ParallelGetPerProcessor [expr, filename] ParallelGetPerProcessor [filename, root] ParallelGetPerProcessor [filename, root, comm]	プロセッサ0において、「filename」という名前のファイルに「expr」を順番に、プロセッサごとに1行ずつ書き込む。第3の引数は、IDが「root」であるプロセッサにおいて、そのファイルが書き込まれることを指定する。第4の引数は、コミュニケーターワールド「comm」を使用する。	30
ParallelGetPerProcessor [filename] ParallelGetPerProcessor [filename, root] ParallelGetPerProcessr [filename, root, comm]	プロセッサ0において、「filename」という名前のファイルからデータを、プロセッサごとに1行ずつ読み出して返す。第2の引数は、IDが「root」であるプロセッサにおいて、そのファイルが読み出されることを指定する。第3の引数は、コミュニケーターワールド「comm」を使用する。	40

表 N

自動ロードバランシング

【0081】

いくつかの関数呼び出しは、完了までの処理時間の長さに一貫性がない可能性がある。たとえば、Mathematicaでは、呼び出しf[20]の評価には、一般に、f[19]の場合よりかなり長い時間がかかる可能性がある。さらに、クラスタ内の1つまたは複数のプロセッサの速度が異なる場合（たとえば、いくつかのプロセッサが2.6 GHzのコア周波数で動作していて、他のプロセッサが1 GHz未満のコア周波数で動作して

いる場合)は、1つのプロセッサが別のプロセッサより早くタスクを終了する可能性がある。

【0082】

実施形態によっては、高度関数モジュール304は、そのような状況におけるコンピュータクラスタ100の動作を改良することが可能な呼び出しを含む。実施形態によっては、rootプロセッサは、可能な関数呼び出しの小さなサブセットを、クラスタ100上の各プロセッサに割り当てる。結果を最初に返したプロセッサには、可能な呼び出しの第2の小さなサブセットが割り当てられる。rootプロセッサは、評価が完了するまでは結果を受け取るので、可能な呼び出しの小さなサブセットを割り当て続ける。各プロセッサが終了する順番は、数式が評価されるたびに変わる可能性があるが、rootプロセッサは、使用可能になったプロセッサにさらなる作業を割り当て続ける。

10

【0083】

一実例として、4つのプロセッサと、評価対象のf[1]からf[100]までを考える。これは、プロセッサ0から3までのそれぞれに、f[1]、f[2]、f[3]、f[4]を割り当てることによって実装可能である(rootは自身に割り当てることが可能)。f[2]の結果が最初に返されると、プロセッサ1にはf[5]が割り当てられる。f[4]の結果が次に返されると、プロセッサ3にはf[6]が割り当てられる。これらの割り当ては、すべての結果が計算されるまで続けられる。これらの結果は、ユーザに返される出力のために編成される。

20

【0084】

代替実施形態では、可能な呼び出しのサブセットは、順次的ではなく、任意の順番で割り当てられることが可能であり、あるいは、バッチ割り当てされることが可能である(たとえば、f[1]、f[5]、f[9]がプロセッサ1に割り当てられる、など)。また、これらのサブセットは、委託によって編成されることも可能である。たとえば、あるプロセッサノードが、その他のプロセッサによって直接制御されるわけでは必ずしもない。代わりに、大きなサブセットをプロセッサに割り当てることが可能であり、そのプロセッサは、その作業のサブセットを他のプロセッサに割り当てるであろう。この結果として、巨大な軍隊のような割り当ての階層が作成されるであろう。

【表15】

LoadBalanceFunctionToList[f, count]	1から「count」までの関数f[i]を、ロードバランシング手法を用いて、クラスタ全体にわたって評価し、これらの結果をリストのかたちで返す。第3の引数は、このリストが、IDが「root」であるプロセッサに集められるようにする。
LoadBalanceFunctionToList[f, count, root]	

30

表0

3. 受信済みメッセージキュー306

【0085】

一実施形態では、クラスタノードモジュール204は、受信済みメッセージキュー306を含む。受信済みメッセージキュー306は、他のクラスタノードモジュールから受信されたメッセージを記憶するデータ構造を含む。たとえば数式が完了しているかどうかなど、受信されたメッセージに関係する関連データも、受信済みメッセージキュー306に記憶されることが可能である。受信済みメッセージキュー306は、キューおよび/または別の種類のデータ構造、たとえば、スタック、リンクされたリスト、配列、ツリーなどを含むことが可能である。

40

4. メッセージ受信キュー308

【0086】

一実施形態では、クラスタノードモジュール204は、メッセージ受信キュー308を含む。メッセージ受信キュー308は、式の送信先として期待されるロケーション、およ

50

びその式の送信元として期待されるプロセッサに関する情報を記憶するデータ構造を含む。メッセージ受信キュー308は、キューおよび/または別の種類のデータ構造、たとえば、スタック、リンクされたリスト、配列、ツリーなどを含むことが可能である。

B. クラスタ構成モジュール208

【0087】

クラスタ構成モジュール208は、コンピュータシステム110、120、130にクラスタコンピューティングサポートを追加するために、複数のクラスタノードモジュールを初期化するプログラムコードを含む。参照により全体が本明細書に組み込まれて、本明細書の一部をなしている、Daugerに対して発行された米国特許第7136924号(「'924特許」)明細書には、コンピュータクラスタの並列動作および制御のための方法およびシステムが開示されている。1つの方法は、一般に、ネットワークサービスによって発見可能なオペレーティングシステムを有する1つまたは複数のパーソナルコンピュータを取得することを含む。実施形態によっては、この方法は、カーネルモジュールを実行することが可能な1つまたは複数のプロセッサまたはプロセッサコアを取得することを含む。'924特許に記載されるように、ソフトウェアアプリケーションのクラスタノード制御およびインタフェース(CNCI: cluster node control and interface)グループが各ノードにコピーされる。CNCIアプリケーションがノードで実行されている場合、クラスタ構成モジュール208は、クラスタノードモジュール204が、カーネルモジュール206との組み合わせで、そのノードの処理用リソースを使用して、コンピュータクラスタの一部として並列計算タスクを実行できるようにすることが可能である。クラスタ構成モジュール208は、本開示に関連して、クラスタ作成プロセスを大幅に自動化することが可能である。

10

20

C. ユーザインタフェースモジュール202

【0088】

実施形態によっては、コンピュータクラスタ100は、たとえば、Mathematicaフロントエンドやコマンドラインインタフェースのようなユーザインタフェースモジュール202を含み、ユーザインタフェースモジュール202は、カーネルモジュール206が、グラフィカル出力を提供し、グラフィカル入力を受け付け、他の、グラフィカルユーザインタフェースまたはコマンドラインインタフェースが提供するユーザ通信方法を提供するための、プログラムコードを含む。ユーザインタフェースモジュール202をサポートするために、実施形態によっては、クラスタノードモジュール204aの動作が変更される。ユーザインタフェースモジュール202は、出力をユーザに直接送信したり、入力をユーザから直接受信したりする代わりに、ユーザインタフェースモジュール202が接続されているクラスタノードモジュール204aを起動し、クラスタノードモジュール204aとユーザインタフェースモジュール202との間の接続、たとえば、MathLink接続などを形成するパラメータを指定する。ユーザインタフェースモジュールによるクラスタノードモジュール204aの起動により、クラスタ上の残りのクラスタノードモジュール204b~eを起動する命令、およびクラスタ上のすべてのカーネルモジュール206a~eを起動するシーケンスを完了する命令の実行が開始されることが可能である。正規にはカーネルモジュール206aに宛てられた、ユーザインタフェースモジュール202からのパケットは、クラスタノードモジュール204aによって、ユーザコマンドとして受け付けられる。クラスタノードモジュール204aに関連付けられたカーネルモジュール206aからの出力は、ユーザに対する表示のために、ユーザインタフェースモジュール202に転送されることが可能である。クラスタノードモジュール204a~eのいずれもが、ユーザインタフェースモジュール202と通信するように構成可能である。

30

40

D. カーネルモジュール206

【0089】

カーネルモジュール206は、典型的には、ユーザまたはスクリプトから与えられる高級コード、コマンド、および/または命令を、低級コード、たとえば、機械語やアセンブ

50

り言語などに翻訳するプログラムコードを含む。一実施形態では、各クラスタノードモジュール 204 a ~ e は、他のすべてのクラスタノードモジュールに接続され、各カーネルモジュール 206 a ~ e は、1つのクラスタノードモジュール 204 だけが割り当てられ、それに接続される。一実施形態では、プロセッサごとに、クラスタノードモジュールとカーネルモジュールのペアが1つ存在する。たとえば、シングルプロセッサコンピュータシステムを含むコンピュータクラスタ 100 の一実施形態では、クラスタノードモジュールとカーネルモジュールの各ペアが、シングルプロセッサコンピュータに常駐することが可能である。コンピュータが複数のプロセッサまたはプロセッサコアを含む場合、このコンピュータは、クラスタノードモジュールとカーネルモジュールの複数のペアを含むことが可能であるが、その場合でも、それらのペアは、クラスタノードモジュールのネットワーク接続を介して通信することが可能である。

10

IV. クラスタコンピューティング方法

【0090】

一実施形態では、コンピュータクラスタ 100 は、クラスタ初期化プロセス、クラスタノードモジュールの操作方法、およびクラスタシャットダウンプロセスを含む。

A. クラスタ初期化プロセス

【0091】

一実施形態では、クラスタ構成モジュール 202 は、図 4 に示されるように、1つまたは複数のカーネルモジュール 206 にクラスタコンピューティングサポートを提供するために、1つまたは複数のクラスタノードモジュール 204 を初期化する。

20

【0092】

402 では、コンピュータクラスタ 100 において、クラスタノードモジュールが起動される。一実施形態では、(たとえば、ユーザが位置する) 第 1 のプロセッサ 112 a で実行されているクラスタノードモジュール 204 a が、クラスタ構成モジュール 208 を介して、コンピュータクラスタ 100 上の他のプロセッサ 112 b、122 a ~ b、132 にアクセスして、クラスタノードモジュール 204 b ~ e をクラスタ全体に向けて起動する。一代替実施形態では、クラスタ構成モジュール 208 が、通信ネットワーク 102 を介して互いに接続されているプロセッサ 112 a ~ b、122 a ~ b、132 をサーチし、プロセッサ 112 a ~ b、122 a ~ b、132 のそれぞれにおいてクラスタノードモジュール 204 a ~ e を起動する。

30

【0093】

404 では、クラスタノードモジュール 204 a ~ e が、互いの間の通信を確立する。一実施形態では、クラスタノードモジュール 204 a ~ e のそれぞれが、クラスタ構成モジュール 208 によってコンピュータクラスタ 100 上で起動された他のクラスタノードモジュール 204 a ~ e との直接通信を、MPI_Init コマンドを用いて確立する。

【0094】

406 では、各クラスタノードモジュール 204 が、カーネルモジュール 206 に接続することを試みる。一実施形態では、クラスタノードモジュール 204 a ~ e の各インスタンスが、居場所を定め、起動され、MathLink 接続および/または同様の、たとえば、カーネルモジュール 206 に組み込まれた接続ツールを介してローカルカーネルモジュールに接続する。

40

【0095】

408 では、カーネルモジュール 206 に接続されていないクラスタノードモジュール 204 がシャットダウンされる。一実施形態では、各クラスタノードモジュール 204 は、ローカルカーネルモジュールが見つからないかどうか、あるいは接続されていないかどうかを判定する。一実施形態では、各クラスタノードモジュール 204 は、カーネルモジュール 206 との接続に失敗した場合には、そのことを、コンピュータクラスタ 100 上の他のクラスタノードモジュールに報告し、終了する。

【0096】

410 では、残りのクラスタノードモジュール 204 にプロセッサ識別番号が割り当て

50

られる。一実施形態では、残りの各クラスタノードモジュール 204 は、アクティブなプロセッサの総数 (N) を計算し、アクティブなクラスタノードモジュール 204 a ~ e およびカーネルモジュール 206 a ~ e の残りのサブセットを表す識別番号を決定する。この新しい、クラスタノードモジュールとカーネルモジュールのペアのセットは、たとえば、0 から N - 1 の番号が付けられることが可能である。

【0097】

412 では、カーネルモジュール 206 a ~ e においてメッセージパッシングサポートが初期化される。一実施形態では、各クラスタノードモジュール 204 は、メッセージパッシングをサポートするために、ローカルカーネルモジュール 206 に初期化コード (たとえば、`Mathematica` 初期化コード) を与える。

10

【0098】

最後に、414 では、クラスタノードモジュール 204 a ~ e は、ユーザエントリを受け付けるループに入る。一実施形態では、他のクラスタノードモジュール 204 のそれぞれが、ネットワーク 102 に接続されている他のすべてのクラスタノードモジュール 204 a ~ e からのメッセージを待っている間に、第 1 のプロセッサ 112 a におけるクラスタノードモジュール 204 a がユーザ制御に戻った後に、主ループ (たとえば、クラスタ動作ループ) が実行を開始する。

【0099】

初期化プロセスでは、カーネルモジュール 206 a ~ e がメッセージを互いに送信するための様式を可能にする構造が作成される。実施形態によっては、初期化が完了した時点で、クラスタ内において、任意のカーネルモジュールが他の任意のカーネルモジュールにデータを送信したり、他の任意のカーネルモジュールからデータを受信したりすることが可能になる。クラスタノードモジュールは、カーネルモジュールが他のカーネルモジュールと直接通信しているという錯覚を起こさせる。初期化プロセスは、図 2 の例で示されるような、コンピュータクラスタ 100 上のカーネルモジュール間の関係を作り上げることが可能である。

20

B. クラスタノードモジュールの操作

【0100】

一実施形態では、クラスタノードモジュール 204 は、図 5 に示されるように、主ループの間に、カーネルモジュール 206 に対するクラスタコンピューティングサポートを実装する。

30

【0101】

502 では、クラスタノードモジュール 204 は、他のクラスタノードモジュールからのユーザコマンドまたはメッセージを待つ。一実施形態では、他のクラスタノードモジュール 204 b ~ e がメッセージのチェックを続けている間に、ユーザインタフェースモジュール 202 に接続されたクラスタノードモジュール 204 a は、ユーザコマンドを待つ。

【0102】

コマンドまたはメッセージが受信されたら、504 へ進む。504 では、クラスタノードモジュール 204 a は、受信されたメッセージが終了 (`quit`) コマンドかどうかを判定する。終了コマンドが受信された場合、クラスタノードモジュール 204 a は、ループを終了し、505 のクラスタノードモジュールシャットダウンプロセスへ進む。受信されたメッセージが終了コマンドでない場合は、プロセスは 506 へ進む。

40

【0103】

506 では、受信されたコマンドが、コンピュータクラスタ 100 上のすべてのクラスタノードモジュール 204 a ~ e へ伝達される。一実施形態では、ユーザがユーザインタフェースモジュール 202 にコマンドを入力すると、ユーザインタフェースモジュール 202 に接続されたクラスタノードモジュール 204 a が、そのユーザコマンドを、コンピュータクラスタ 100 の他のすべてのクラスタノードモジュール 204 b ~ e にサブミットする。ユーザコマンドは、シンプルであってよい (たとえば、「1 + 1」)、カーネ

50

ルモジュール206a~e(たとえば、Mathematicaカーネル)間のメッセージパッシングを実行する、ユーザインタフェースモジュール202(たとえば、Mathematicaフロントエンド)内からのMPI呼び出しを含む、コード(たとえば、Mathematicaコードなど)の完全なサブルーチンおよびシーケンスであってもよい。これらは、クラスタノードモジュール204とそのローカルカーネルモジュール206との間で特別に識別されたメッセージを用いて実装される基本MPI呼び出しを含む。

【0104】

508では、メッセージ(またはユーザコマンド)が、カーネルモジュール206a~eへ伝達される。一実施形態では、ユーザインタフェースモジュール202に接続されたクラスタノードモジュール204aは、ユーザコマンドを、クラスタノードモジュール204aが接続されているカーネルモジュール206aへサブミットする。他のクラスタノードモジュール204b~eのそれぞれは、メッセージを受信した後に、接続されている、カーネルモジュール206b~eのそれぞれへコマンドをサブミットする。

10

【0105】

510では、クラスタノードモジュール204がカーネルモジュール206から結果を受け取る。一実施形態では、カーネルモジュール206は、その評価を完了した後、そのカーネルモジュールの出力を、そのカーネルモジュールが接続されているクラスタノードモジュール204に返す。カーネルモジュールからの結果の性質に応じて、クラスタノードモジュール204は、その結果を、ローカルコンピュータシステムに報告するか、メッセージとして別のクラスタノードモジュール204に渡すことが可能である。たとえば、第1のプロセッサ112aで実行されているクラスタノードモジュール204aは、出力を、そのローカルコンピュータシステム110に報告する。たとえば、第1のプロセッサ112aでは、クラスタノードモジュール204aが、カーネルモジュール206aの出力を直接報告するだけである。

20

【0106】

512では、他のクラスタノードモジュール204からのメッセージに対する応答が行われる。一実施形態では、各クラスタノードモジュール(たとえば、クラスタノードモジュール204a)は、他のクラスタノードモジュール204b~eおよびカーネルモジュール206aからのメッセージを、それらがなくなるまで繰り返しチェックし、それらに応答する。一実施形態では、カーネルモジュール206からの出力メッセージが、ローカルコンピュータシステムの出力に転送される。他のクラスタノードモジュール204からのメッセージは、受信済みメッセージキュー306(「RMQ」)に転送される。メッセージ受信キュー308(「MRQ」)内の各エントリからのデータが、RMQ306内のエントリと照合される(たとえば、前述のmpiIRecv[]呼び出しの説明を参照)。MRQ308からのデータが合致すれば、そのデータが、RMQ306内の対応するデータと結合され、「完了」としてマーキングされる(たとえば、前述のmpiTest[]呼び出しの説明を参照)。このプロセスは、クラスタノードモジュール204a~eのピアツーピア動作を提供する。この仕組みにより、複数の同時に実行されるカーネルモジュール(たとえば、Mathematicaカーネル)において実行されるコードは、ペア単位または集団で対話することにより、1つのカーネルが単独で実行した場合より大規模かつ/または高速に、計算、処理、または他の作業を実行することが可能である。このようにして、どのような作業が行われるかをユーザコマンドで指定する、ユーザが入力した命令およびデータが、より迅速に、かつ/または、より確実に実行されることが可能である。メッセージへの応答が完了すると、プロセスは502へ戻る。

30

40

C. クラスタシャットダウンプロセス

【0107】

一実施形態では、コンピュータクラスタ100は、システムをシャットダウンするプロセスを含む。ユーザインタフェースモジュール202に接続されたクラスタノードモジュール204aにおける動作プロセス(または主ループ)が、「Quit」または「Exit」コマンド(終了コマンド)を検出するか、他の方法で、シャットダウンを示すメ

50

ッセージをユーザから受け取った場合は、クラスタノードモジュール 204 a ~ e およびカーネルモジュール 206 a ~ e をシャットダウンするシーケンスが起動される。一実施形態では、ユーザインタフェースモジュール 202 に接続されたクラスタノードモジュール 204 a は、他のすべてのクラスタノードモジュール 204 b ~ e に終了メッセージを送信する。各クラスタノードモジュール 204 は、その終了コマンドを、それぞれのローカルカーネルモジュール 206 に転送する。各クラスタノードモジュール 204 は、それぞれの `Mathematica` カーネルが終了した後、それぞれの、他のクラスタノードモジュールとの通信ネットワークを切断することに進む（たとえば、前述の `MPI_Finalize` コマンドの説明を参照）。プロセスの最後に、各クラスタノードモジュール 204 が実行を終了する。

10

V. 運用例

【0108】

例示を目的として、コンピュータクラスタシステムが実運用で使用されるサンプルシナリオを説明する。以下のサンプルシナリオでは、`Mathematica` コードの例が与えられ、クラスタシステムによってコードがどのように実行されるかの説明が行われる。

基本 MPI

【0109】

各ノードが使用できる基本データは、ノードの識別番号およびプロセッサの総数を含む。

【数 1】

```
In[1]:= {$IdProc, $NProc}
Out[1]:= {0, 2}
```

20

【0110】

第 1 の要素は、プロセッサごとに一意でなければならず、第 2 の要素は、一般に、すべてのプロセッサに対して同じである。プロセッサ 0 は、他のどのような値が `mpiGather[]` などの集団（後述）通信呼び出しを使用しているかを知ることが可能である。

【数 2】

```
In[2]:= mpiGather[{$IdProc, $NProc}, list, 0]; list
```

30

```
Out[2]:= {{0, 2}, {1, 2}}
```

ピアツーピア MPI

【0111】

`mpiSend` および `mpiRecv` コマンドは、可能な基本メッセージパッシングを行うが、どのプロセッサをターゲットとするかを定義する必要がある。以下では、プロセッサの各ペアが互いをポイントするように、新しい変数 `targetProc` を定義している。

【数 3】

```
In[3]:= targetProc=If[1==Mod[$IdProc, 2], $IdProc-1, $IdProc+1]
Out[3]:= 1
```

40

【0112】

この例では、偶数プロセッサが、それぞれの「右」プロセッサをターゲットとし、奇数プロセッサが、それぞれの「左」プロセッサをポイントとする。たとえば、プロセッサが列に並べられ、順に番号が付けられた場合は、すべての偶数番号プロセッサが、その列における直後のプロセッサとペアになり、すべての奇数番号プロセッサが、直前のプロセッサとペアになる。その場合は、次のようにメッセージが送信されることが可能である。

【数4】

```
In[4]:= If [ 1==Mod[ $IdProc ,
2],mpiSend[N[Pi,22],targetProc,
    mpiCommWorld,d], mpiRecv[a,targetProc,mpiCommWorld,d]]
```

【0113】

I f []文は、それらのプロセッサに別々のコードを評価させる。すなわち、奇数プロセッサが の22桁を送信し、偶数プロセッサがそのメッセージを受信する。これらのM P I呼び出しは何も返さないことに注意されたい。受信済みメッセージは、変数 a の中にある。

10

【数5】

```
In[5]:= a
Out[5]:= 3.1415926535897932384626
In[6]:= Clear[a]
```

【0114】

奇数プロセッサの変数 a は、何も定義されない。さらに、\$ N P r o c が8の場合、プロセッサ3がプロセッサ2に を送信し、プロセッサ5がプロセッサ4に を送信し、以降も同様である。これらのメッセージは、プロセッサ0を介しては送信されず、それらだけで伝達されている。

20

【0115】

m p i I S e n d および m p i I R e c v コマンドの文字「I」は、非同期動作を示しており、非同期動作は、メッセージが送受信されている間、または、他のプロセッサがピジーの場合に他の作業を行うことを可能にする。したがって、前述の例は、次のように、非同期で行われることが可能である。

【数6】

```
In[7]:= If[1==Mod[$IdProc, 2],mpiISend[N[Pi,22],targetProc,
    mpiCommWorld,d,e],
    mpiIRecv[a,targetProc,mpiCommWorld,d,e]]
```

30

【0116】

変数 e は、メッセージを識別する重要データを有し、m p i T e s t [e] は、それらの数式がアクセス対象になるまでは、T r u e を返すことが可能である。この時点で、他の多くの評価を実行することが可能である。次に、これらのデータが必要になるタイミングを、以下のように、m p i T e s t を用いてチェックすることが可能である。

【数7】

```
In[29]:= mpiTest[e]
Out[29]:= True
In[30]:= a
Out[30]:= 3.1415926535897932384626
In[31]:= Clear[a,e]
```

40

【0117】

m p i W a i t [e] コマンドも使用されていることが可能であり、これは、m p i T e s t [e] が T r u e を返すまで戻らない。これらのピアツーピア呼び出しを使用する権限を持つと、任意の問題に対して任意のメッセージパッシングパターンを構築することが可能になる。

集団 M P I

【0118】

ケースによっては、そのような明示的な制御は不要であり、一般に使用される通信パタ

50

ーンで十分である。プロセッサ 0 が、すべてのプロセッサが有することを意図された、 b という数式を有するものとする。ブロードキャスト MPI 呼び出しが、以下を行う。

【数 8】

```
In[8]:= mpiBcast[b, 0, mpiCommWorld]
```

【0119】

第 2 の引数は、どのプロセッサがこのブロードキャストの「root」かを指定する。他のすべてのプロセッサは、それぞれの b が上書きされる。すべてのプロセッサから値を収集するために、次のように、`mpiGatherD` を用いる。

【数 9】

```
In[9]:= mpiGather[b, c, 0, mpiCommWorld]
```

10

【0120】

プロセッサ 0 の変数 c は、`mpiCommWorld` にあるすべてのプロセッサのすべての b のリストを用いて書き込まれる。次のように、時間的に反対であるのが、`mpiScatter` である。

【数 10】

```
In[10]:= Clear[b]; a = {2, 4, 5, 6}; mpiScatter[a, b, 0,
      mpiCommWorld]; b
```

```
Out[10]:= {2, 4}
```

20

【0121】

`mpiScatter` コマンドは、(可能な場合には) 変数 a を偶数個の小片に切り分け、それらを各プロセッサに分散させる。これは、 $\$NProc = 2$ の場合の結果であり、 $\$NProc = 4$ の場合、 b は {2} を有するだけである。

【0122】

MPI は、メッセージングと混合されたシンプルな計算を実行するためにリダクション操作を行う。以下について考える。

【数 11】

```
In[11]:= a = {{2 + $IdProc, 45[], 3, {1 + $IdProc, $NProc[]}};
      mpiReduce [a,d,mpiSum, 0,mpiCommWorld ]
```

30

```
In[12]:= d
```

```
Out[12]:= {{5, 90}, 6, {3, 4}}
```

【0123】

`mpiSum` 定数は、すべてのプロセッサの変数 a が合計されることを示す。この場合では、 $\$NProc$ が 2 なので、同一でない要素は奇数側の合計をもたらし、同一である要素は偶数側の合計をもたらしている。

40

【0124】

すべてが指定されているとは限らない場合、これらの呼び出しのほとんどは、デフォルト値を有する。たとえば、以下の呼び出しのそれぞれは、前述の `mpiGather[]` 呼び出しと等価の効果をもたらし、

【数 12】

```
mpiGather[b, c, 0]
```

```
mpiGather[b, c]
```

```
c = mpiGather[b]
```

高級呼び出し

50

【 0 1 2 5 】

高級呼び出しは、よく使用されるアプリケーションプログラム呼び出し（たとえば、`Mathematica`呼び出し）の便利な並列バージョンを含むことが可能である。たとえば、`ParallelTable[]`は、評価が分散様式で自動的に行われること以外は、`Table[]`と同様である。

【 数 1 3 】

```
In[13]:= ParallelTable[i, {i, 100}, 0]
Out[13]:= {1, 2, 3, 4, 5, ..., 99, 100}
```

10

【 0 1 2 6 】

第3の引数は、答えをプロセッサ0と突き合わせることを指定する。これは、多数の呼び出しを並列化して複雑な関数にするための、有用かつシンプルな方法である。以下のように、広い範囲の入力に対して、複雑な関数を定義し、それを評価することが可能である。

【 数 1 4 】

```
In[14]:= g[x_] := Gamma[2 + 0.5*(x-1)];
ParallelTable[g[i], {i, 100}, 0]
Out[14]:= {1, 1.32934, 2., 3.32335, 6., 11.6317, 24.,
52.3428, 120., 287.885, 720}
```

20

【 0 1 2 7 】

`ParallelFunctionToList[]`も、この形式の並列化を実行するための簡略化された方法を提供する。

単純でない通信の操作

行列操作

【 0 1 2 8 】

実施形態によっては、以下のように、1つまたは複数の関数が並列に、行列計算の解決を支援することが可能である。

【 数 1 5 】

```
In[15]:= a = Table[i + 3* $IdProc + 2 j, {i, 2}, {j, 4}]
Out[15]:= {{3, 5, 7, 9}, {4, 6, 8, 10}}
In[16]:= t = ParallelTranspose[a]
Out[16]:= {{3, 4, 6, 7}, {5, 6, 8, 9}}
```

30

フーリエ変換

【 0 1 2 9 】

大きな配列のフーリエ変換は、並列化によって高速の解決が可能である。あるいは、クラスタ上で解決可能にされることが可能である。これは、フーリエ変換がすべてメモリに保持されることが可能だからである。前述の例の2次元フーリエ変換は、以下のとおりである。

40

【 数 1 6 】

```
In[17]:= f = ParallelFourier[a]
Out[17]:= {{32. + 0. I, -4. - 4. I, -4., -4. + 4. I}, {-3. -
3. I, 0. + 0. I, 0., 0. + 0. I}}
```

エッジセル管理

【 0 1 3 0 】

多くの問題が、パーティション間の対話を必要とするが、これはエッジ要素上でのみ行われる。これらのエッジを維持することは、`EdgeCell[]`を用いて行われること

50

が可能である。

【数 17】

```
In[18]:= a = {2, 4, 5, 6, 7}+8*$IdProc
Out[18]:= {2, 4, 5, 6, 7}
In[19]:= EdgeCell[a]; a
Out[19]:= {14, 4, 5, 6, 12}
```

要素管理

【0131】

小片ベースの問題では、各アイテムが空間内をドリフトする可能性があり、場合によっては、個々のプロセッサのパーティションの外へ出る可能性がある。これは、以下のように、ElementManage[]で解決可能である。

【数 18】

```
In[20]:= list={{0,4},{1,3},{1,4},{0,5}}; fcn[x_]:=x[[1]]
In[21]:= ElementManage[list, fcn]
Out[21]:= {{0, 4}, {0, 5}, {0, 4}, {0, 5}}
In[21]:= ElementManage[list, 2]
Out[21]:= {{0, 4}, {0, 5}, {0, 4}, {0, 5}}
```

【0132】

ElementManageの第2の引数は、リストの要素をテストする方法を示している。fcn識別子は、どのプロセッサがその要素の「ホーム」であるかを返す。整数を渡すことは、各要素がそれ自体リストであって、その第1の要素が、0から渡された引数までの範囲の数であることを仮定している。

【0133】

前述の各例は、Mathematicaソフトウェアと、MPI呼び出しおよびクラスタコマンドの特定の実施形態とに関連していたが、これらの実施形態は、本発明のシステムおよび方法の種々の実施形態の特徴を例示するためにのみ用いられていることを理解されたい。

VI.さらなる実施形態

【0134】

特定の実施形態を参照して、クラスタコンピューティングの手法、モジュール、呼び出し、および関数が開示されているが、本開示は、それによって限定されることを意図するものではない。むしろ、当業者であれば、本明細書における開示から、クラスタ呼び出し、関数、および管理システムの本開示そのものの選択に対して幅広い代替があることを理解されよう。たとえば、本明細書に記載のように、シングルノードカーネルは、様々な管理ツールを用いて管理されることが可能であり、かつ/または、ユーザによって手動で管理されることが可能である。別の例として、クラスタノードモジュールは、クラスタコンピューティングに無関係の呼び出しおよびプロシージャを含む、本明細書で開示されていない、さらなる呼び出しおよびプロシージャを含むことが可能である。

【0135】

当業者であれば、本明細書における開示から、他の実施形態も明らかであろう。さらに、記載された実施形態は、例としてのみ提示されており、本開示の範囲を限定するものではない。実際、本明細書に記載の新規な方法およびシステムは、本発明の趣旨から逸脱しない他の様々な形態で実施されることが可能である。したがって、当業者であれば、本明細書における開示に鑑みて、他の組み合わせ、省略、置換、および修正が明らかであろう。したがって、本開示は、開示された実施形態によって限定されるものではなく、添付の特許請求の範囲の参照によって規定されるものとする。添付の特許請求の範囲およびそれらの等価物は、本発明の範囲および趣旨を逸脱しない形態または修正を包含するものとする。

10

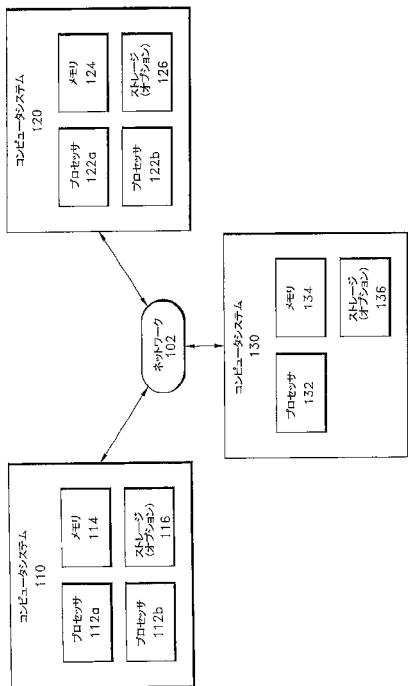
20

30

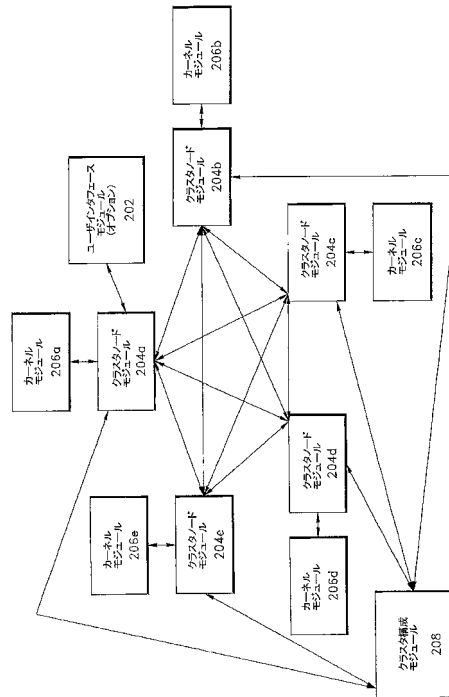
40

50

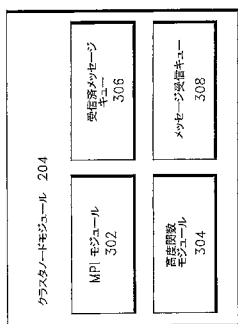
【 図 1 】



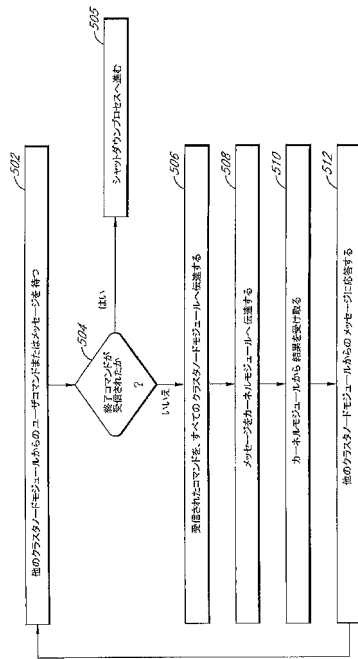
【 図 2 】



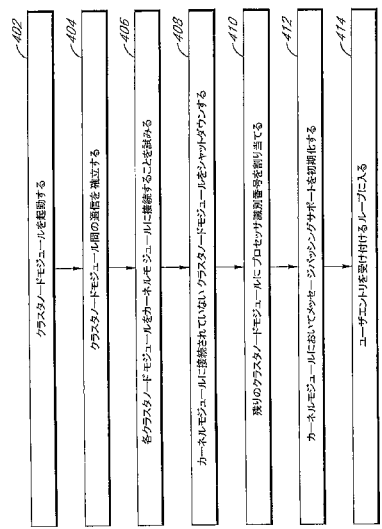
【 図 3 】



【 図 5 】



【 図 4 】



【手続補正書】【提出日】平成28年11月16日(2016.11.16)【手続補正1】【補正対象書類名】特許請求の範囲【補正対象項目名】全文【補正方法】変更【補正の内容】【特許請求の範囲】【請求項1】

複数のノードと、

非同期呼び出しを用いて、互いに通信する前記ノードのためのメカニズムと、を有し、前記ノードの1または2以上が、コマンドを受信し、コンピュータクラスタのクラスタ初期化処理を開始するように構成され、ここで前記クラスタ初期化処理が2または3以上の前記ノードの通信を確立する処理であり、

前記ノードのそれぞれが、ユーザ命令を翻訳処理するように構成されたシングルノードカーネルモジュールのプログラムコードを含む非一時的なコンピュータ可読媒体にアクセスするように構成され、

少なくとも一つの前記ノードが、ユーザインターフェースモジュールまたはスクリプトに結果を返す、コンピュータクラスタ。

【請求項2】

前記非同期呼び出しは、ペイロードとして送信される数式とその数式が送信されるターゲットノードとを含む第一パケットを作る第一命令を含み、

少なくとも一つの前記ノードが、前記数式を前記ターゲットノードへ送るように構成される、請求項1に記載のコンピュータクラスタ。

【請求項3】

前記シングルノードカーネルモジュールが、前記第一パケットをローカルクラスタノードモジュールへ送るように構成される、請求項2に記載のコンピュータクラスタ。

【請求項4】

前記非同期呼び出しは、前記数式が受信されることが期待される場所、および前記数式が受信されることが期待される送信ノードとを含む第二パケットを作る第二命令を含み、

少なくとも一つの前記ノードが、前記第二パケットコンテンツをメッセージ受信キューに記憶するように構成される、請求項2に記載のコンピュータクラスタ。

【請求項5】

前記シングルノードカーネルモジュールが、前記第二パケットをローカルクラスタノードモジュールへ送るように構成される、請求項4に記載のコンピュータクラスタ。

【請求項6】

複数のノードと、

互いに通信する前記ノードのためのメカニズムと、を有し、

前記ノードの1または2以上が、コマンドを受信し、コンピュータクラスタのクラスタ初期化処理を開始するように構成され、ここで前記クラスタ初期化処理が2または3以上の前記ノードの通信を確立する処理であり、

前記ノードのそれぞれが、特別目的のマイクロプロセッサによって実行されるコマンドの中にユーザ命令を翻訳処理するように構成されたシングルノードカーネルモジュールのプログラムコードを含む非一時的なコンピュータ可読媒体にアクセスするように構成され、

少なくとも一つの前記ノードが、ユーザインターフェースモジュールまたはスクリプトに結果を返す、コンピュータクラスタ。

【請求項7】

前記特別目的のマイクロプロセッサは、デジタル信号プロセッサを含む、請求項6に記載のコンピュータクラスタ。

【請求項 8】

前記複数のノードが、2つ以上のノードサブセットグループの中に組織されている、請求項 6 に記載のコンピュータクラスタ。

【請求項 9】

前記2つ以上のノードサブセットグループの少なくとも一つが、前記特別目的のマイクロプロセッサとデータを交換する、請求項 8 に記載のコンピュータクラスタ。

【請求項 10】

複数のクラスタノードモジュールをさらに含み、
前記クラスタノードモジュールのそれぞれが、コンピュータ可読媒体に記憶され、および、シングルノードカーネルモジュールと通信し、かつ1または2以上のその他のクラスタノードモジュールと通信し、指令を受け、かつ前記複数のクラスタノードモジュールが他の一つとピアツーピアアーキテクチャを用いて互いに通信するように、少なくともいくつかの前記指令を翻訳する、請求項 6 に記載のコンピュータクラスタ。

【請求項 11】

前記複数のクラスタノードモジュールがプロセッサキャッシュメモリに格納されている、請求項 10 に記載のコンピュータクラスタ。

【請求項 12】

前記シングルノードカーネルモジュールがプロセッサキャッシュメモリに格納されている、請求項 6 に記載のコンピュータクラスタ。

フロントページの続き

1. UNIX

- (72)発明者 タンネンバウム、ズヴィ
アメリカ合衆国 カリフォルニア州 92657、ニューポート ビーチ、86 シドニー ベイ
ドライブ
- (72)発明者 ドーガー、ディーン、イー .
アメリカ合衆国 カリフォルニア州 92649、ハンティングトン ビーチ、3582 ベンチ
ャー ドライブ