(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2020-9450
(P2020-9450A)
(43)公開日　令和2年1月16日(2020.1.16)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 15/177	(2006.01)	G06F	15/177	A	5B045
G06F 9/445	(2018.01)	G06F	9/445		5B376
G06F 1/24	(2006.01)	G06F	1/24		

審査請求 有　請求項の数 15　ＯＬ　（全 40 頁）

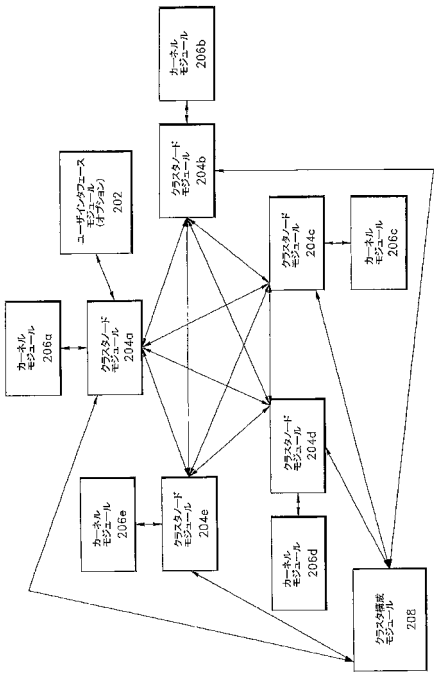
(21)出願番号　特願2019-128168 (P2019-128168)
(22)出願日　令和1年7月10日 (2019.7.10)
(62)分割の表示　特願2016-185469 (P2016-185469)の分割
原出願日　平成19年6月7日 (2007.6.7)
(31)優先権主張番号　60/813,738
(32)優先日　平成18年6月13日 (2006.6.13)
(33)優先権主張国・地域又は機関　米国 (US)
(31)優先権主張番号　60/850,908
(32)優先日　平成18年10月11日 (2006.10.11)
(33)優先権主張国・地域又は機関　米国 (US)

(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　９４０４０、マウンテン　ビュー、サン　アントニオ　ロード　５６５、アパートメント　５０７
(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　９２６４９、ハンティングトン　ビーチ、マクファデン　アベニュー　５２５１
(74)代理人　110000729
特許業務法人　ユニアス国際特許事務所

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】
【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が提供される。
【解決手段】一実施形態では、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】
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、
クラスタ構成モジュール（２０８）によって、前記ノードの１または２以上が、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、
前記クラスタ初期化処理の後で、前記ノードのそれぞれが、ユーザ命令を翻訳処理する機能をマイクロプロセッサに実現させるための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し、前記マイクロプロセッサによって前記プログラムコードが実行され、
少なくとも一つの前記ノードと通信する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、前記少なくとも一つの前記ノードを介して、前記マイクロプロセッサが前記プログラムコードを実行した結果を返す、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前記シングルノードカーネルモジュールに接続されたローカルの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場所であるターゲットノード、および前記数式を送信するノードであ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、前記第二パケットの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前記シングルノードカーネルモジュールに接続されたローカルの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、
クラスタ構成モジュールによって、前記ノードの１または２以上が、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、
前記クラスタ初期化処理の後で、前記ノードのそれぞれが、ユーザ命令を翻訳処理する機能を特別目的のマイクロプロセッサに実現させるための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し、前記マイクロプロセッサによって前記プログラムコードが実行され、
少なくとも一つの前記ノードと通信する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、前記少なくとも一つの前記ノードを介して、前記マイクロプロセッサが前記プログラムコードを実行した結果を返す、コンピュータクラスタ。
【請求項７】
前記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】
２つ以上のノードのサブセットグループのそれぞれは、前記複数のノードを含む、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つ以上のノードのサブセットグループの少なくとも一つにおいて、前記複数のノードが、前記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、
クラスタノードモジュールのそれぞれが、
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機能をマイクロプロセッサに実現させることを特徴とする、請求項６に記載のコンピュータクラスタ。
【請求項１１】
前記マイクロプロセッサで実現される際に、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記憶されている、請求項１０に記載のコンピュータクラスタ。
【請求項１２】
前記マイクロプロセッサで実現される際に、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記憶されている、請求項６に記載のコンピュータクラスタ。
【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】
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、ｇｒｉｄＭａｔｈｅｍａｔｉｃａ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。ｇｒｉｄＭａｔｈｅｍａｔｉｃａ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、そこから使用できるようにされる。

【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の少なくとも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。

【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の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の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。

【００１１】

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。

【００１２】

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。

【００１３】

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。

【００１４】

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。

【図面の簡単な説明】

【００１５】

【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】

【００１６】

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。

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。

【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。

【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。

【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～ｅは、カーネルモジュール２０６ａ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２０４ａは、他の１つまたは複数のクラスタノードモジュール２０４ｂ、２０４ｃ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２０４ａと別のクラスタノードモジュール２０４ｃ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２０４ａ～ｅのそれぞれは、それぞれのカーネルモジュール２０６ａ～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２０４ａは、カーネルモジュール２０６ａ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２０６ａ～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２０４ａ～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２０６ａ～ｅ間の通信を提供する。カーネルモジュール２０６ａ～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２０４ａ～ｅを介して第１のクラスタノードモジュール２０４ａへ返され、第１のクラスタノードモジュール２０４ａ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。

【００２３】

スレッド実行の間のカーネルモジュール２０６ａ～ｅ間の相互通信は、クラスタノードモジュール２０４ａ～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。

【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

【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１１２ａ～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１２２ａ～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１１２ａ～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。

１．プロセッサ１１２ａ～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１１２ａ～ｂを含む。プロセッサ１１２ａ～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１１２ａ～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１１２ａ～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１１２ａ～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。

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なども使用可能であることを理解されたい。

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、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。

４．コンピュータシステム１１０情報

【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）、ＭａｃＯＳ Ｘ（登録商標）、ＭａｃＯＳ 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。

【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。

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。

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。

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　Ｐｉｌｏｔ　ＯＳ、ＯＳ／２、Ａｐｐｌｅ（登録商標）　ＭａｃＯＳ（登録商標）、ＭａｃＯＳ　Ｘ（登録商標）、ＭａｃＯＳ　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。

【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。

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

【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。

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。

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。

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）、ＭａｃＯＳ Ｘ（登録商標）、ＭａｃＯＳ 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。

【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。

【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、任意の種類の、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　Ｄａｔａｌｉｎｋ　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、他の任意の種類の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。

【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の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よう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。

【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。

【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。

【００５８】

これに対し、ｇｒｉｄＭａｔｈｅｍａｔｉｃａ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。

【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。

＜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）がわかっていれば、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。

【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。

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級のルーチン群など、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。

【００６３】

以下の最も基本的なＭＰＩ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、対応するルーチンを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ｍｐｉＲｅｃｖを呼び出していない可能性があるため、ｍｐｉＳｅｎｄ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。

表Ｂ

＜非同期ＭＰＩ呼び出し＞

【００６４】

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]	「ｒｅｑｌｉｓｔ」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。

表Ｃ

【００６５】

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。

【００６６】

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を期待する場所、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。

【００６７】

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。

【００６８】

他のＭＰＩ呼び出しは、基本呼び出しｍｐｉＳｅｎｄ、ｍｐｉＲｅｃｖ、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

【００６９】

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。

【表４】

呼び出し	説明
mpiBcast[expr, root, comm]	コミュニケータワールド「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「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「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。

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。

【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。

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。

【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。

表Ｆ

＜ＭＰＩコミュニケータ呼び出し＞

【００７１】

一実施形態では、ＭＰＩモジュール３０２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ＭＰＩ呼び出しでｍｐｉＣｏｍｍＷｏｒｌｄの代わりに使用可能である。

【表７】

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。

表Ｇ

＜他のＭＰＩサポート呼び出し＞

【００７２】

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
mpiWtime[]	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。
mpWtick[]	ｍｐｉＷｔｉｍｅ［］の時間分解能を返す。
MaxByElement[in]	リスト「ｉｎ」の各リストのすべてのｎ番目の要素について、Ｍａｘ［］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。ｍｐｉＭａｘリダクション操作において用いられる。
MinByElement[in]	リスト「ｉｎ」の各リストのすべてのｎ番目の要素について、Ｍｉｎ［］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。ｍｐｉＭｉｎリダクション操作において用いられる。

表Ｈ

２．高度関数モジュール３０４

【００７３】

一実施形態では、クラスタノードモジュール２０４は、高度関数モジュール３０４を含む。高度関数モジュール３０４は、ＭＰＩモジュール３０２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３０４は、高度関数を実装するにあたり、ＭＰＩモジュール３０２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３０４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３０４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３０４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３０４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。

【００７４】

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＞

【００７５】

一実施形態では、高度関数モジュール３０４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ループなどに匹敵しうる。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３０４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「ｉｎｃｒｅｍｅｎｔ」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。

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。

【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。

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。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１１】

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。

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ビンが、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。

【表１２】

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。

表Ｌ

＜フーリエ変換＞

【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１３】

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。

【表１４】

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、コミュニケータワールド「ｃｏｍｍ」を使用する。
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、コミュニケータワールド「ｃｏｍｍ」を使用する。
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、コミュニケータワールド「ｃｏｍｍ」を使用する。
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、コミュニケータワールド「ｃｏｍｍ」を使用する。

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２．６ＧＨ

ｚのコア周波数で動作していて、他のプロセッサが１ＧＨｚ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。

【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。

【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、それ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] LoadBalanceFunctionToList[f, count, root]	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。

表Ｏ

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。

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。

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。

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、機械語やアセンブリ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２０４ａ～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２０６ａ～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。

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。

【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１１２ａで実行されているクラスタノードモジュール２０４ａ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１１２ｂ、１２２ａ～ｂ、１３２にアクセスして、クラスタノードモジュール２０４ｂ～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１１２ａ～ｂ、１２２ａ～ｂ、１３２をサーチし、プロセッサ１１２ａ～ｂ、１２２ａ～ｂ、１３２のそれぞれにおいてクラスタノードモジュール２０４ａ～ｅを起動する。

【００９３】

４０４では、クラスタノードモジュール２０４ａ～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２０４ａ～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２０４ａ～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２０４ａ～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。

【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２０４ａ～ｅおよびカーネルモジュール２０６ａ～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２０６ａ～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。

【００９８】

最後に、４１４では、クラスタノードモジュール２０４ａ～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２０４ａ～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１１２ａにおけるクラスタノードモジュール２０４ａ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２０６ａ～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。

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。

【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２０４ｂ～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２０４ａ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２０４ａ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２０４ａ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。

【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２０４ａ～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２０４ａ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２０４ｂ～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が（たとえば、「１＋１」）、カーネルモジュール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。

【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。

【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。

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シージャを含む。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「Ｅ

ｘｉｔ」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２０４ａ～ｅおよびカーネルモジュール２０６ａ～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２０４ａは、他のすべてのクラスタノードモジュール２０４ｂ～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。

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}
```

【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

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
```

【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】

Ｉｆ［］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π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ＭＰＩ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。

[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、＄ＮＰｒｏｃが８の場合、プロセッサ３がプロセッサ２にπを送信し、プロセッサ５がプロセッサ４にπ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。

【０１１５】

ｍｐｉＳｅｎｄおよびｍｐｉＲｅｃｖ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、他のプロセッサがビ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]]
```

【０１１６】

変数ｅは、メッセージを識別する重要データを有し、ｍｐｉＴｅｓｔ［ｅ］は、それらの数式がアクセス対象になるまでは、Ｔｒｕｅ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、ｍｐｉＴｅｓｔ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]
```

【０１１７】

ｍｐｉＷａｉｔ［ｅ］コマンドも使用されていることが可能であり、これは、ｍｐｉＴｅｓｔ［ｅ］がＴｒｕｅ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ＭＰＩ

【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

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]
```

【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}
```

【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 ]


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。

【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有する。

[数１２]

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

＜高級呼び出し＞

【０１２５】
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１３]

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

【０１２６】
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１４]

```
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}
```

【０１２７】
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＞

【０１２８】
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１５]

```
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}}
```

＜フーリエ変換＞

【０１２９】
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。

[数１６]

```
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＞

【０１３０】
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

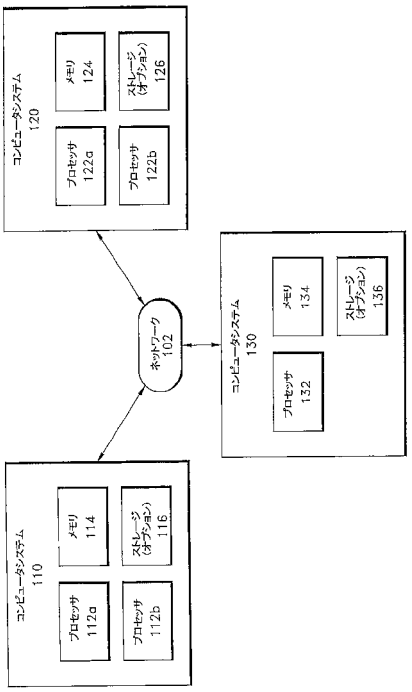
【０１３４】

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。

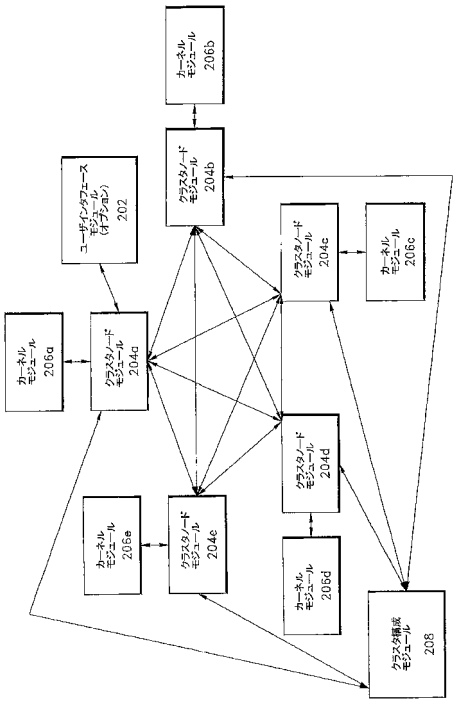
【０１３５】

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。

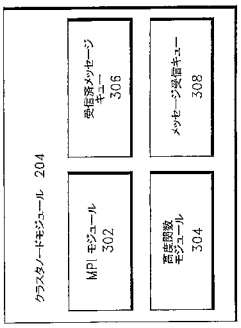
【図１】


コンピュータシステム
120
プロセッサ
122a
プロセッサ
122b
メモリ
124
ストレージ
(オプション)
126
ネットワーク
102
コンピュータシステム
130
プロセッサ
132
メモリ
134
ストレージ
(オプション)
136
コンピュータシステム
110
プロセッサ
112a
プロセッサ
112b
メモリ
114
ストレージ
(オプション)
116

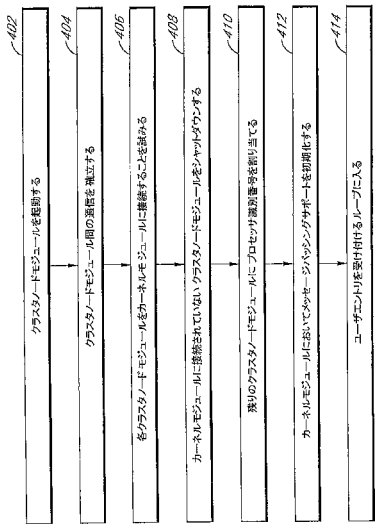
【図２】


カーネル
モジュール
206b
ユーザインタフェース
モジュール
(オプション)
202
クラスタノード
モジュール
204b
クラスタノード
モジュール
204c
カーネル
モジュール
206c
カーネル
モジュール
206a
クラスタノード
モジュール
204a
クラスタノード
モジュール
204d
カーネル
モジュール
206e
クラスタノード
モジュール
204e
カーネル
モジュール
206d
クラスタ構成
モジュール
208

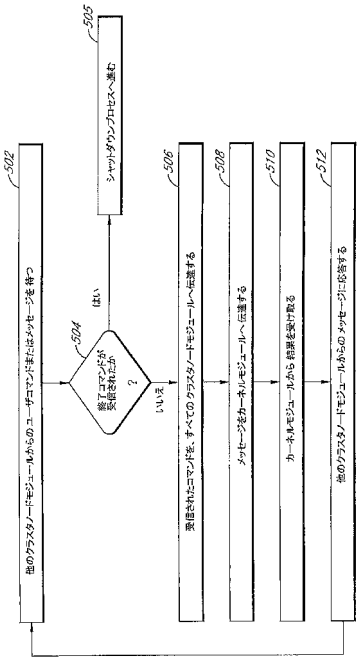
【図３】


クラスタノードモジュール 204
MPI モジュール
302
受信済メッセージ
キュー
306
着信順序
モジュール
304
メッセージ送信キュー
308

【図４】


402
クラスタノードモジュールを起動する
404
クラスタノードモジュール間の通信を確立する
406
各クラスタノードモジュールをカーネルモジュールに接続することを試みる
408
カーネルモジュールに接続されていないクラスタノードモジュールをシャットダウンする
410
残りのクラスタノードモジュールにプロセッサ識別番号を割り当てる
412
カーネルモジュールにおいてメッセージパッシングサポートを初期化する
414
ユーザエントリを受け付けるループに入る

【図５】


502
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
504
終了コマンドが受信されたか？
はい
505
シャットダウンプロセスへ進む
いいえ
506
受信されたコマンド文、すべてのクラスタノードモジュールへ伝達する
508
メッセージをカーネルモジュールへ伝達する
510
カーネルモジュールから結果を受け取る
512
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

【手続補正書】
【提出日】令和1年8月8日(2019.8.8)
【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、前記ノードの１または２以上が、命令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、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ピアツーピアアーキテクチャを用いて互いに通信することを確立する処理であり、前記命令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から受信され、
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、
クラスタ初期化処理の後で、前記ノードの少なくとも一つと通信するユーザインターフェースモジュールまたはスクリプトからユーザ命令を受信することによって、前記ノードのそれぞれが、ユーザ命令を翻訳処理する機能をマイクロプロセッサに実現させるための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し、前記マイクロプロセッサによって前記プログラムコードが実行されるように構成され、
前記ノードの少なくとも一つが、前記ユーザインターフェースモジュールまたは前記スクリプトへ結果を返す、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、
前記ノードの少なくとも一つが、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前記シングルノードカーネルモジュールに接続されたローカルの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場所であるターゲットノード、および前記数式を送信するノードであ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、
前記ノードの少なくとも一つが、前記第二パケットの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前記シングルノードカーネルモジュールに接続されたローカルの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、前記ノードの１または２以上が、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命令を受信するように構成され、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、
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、
前記ノードのそれぞれが、ユーザ命令を翻訳処理する機能を、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へ実現させるための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、

前記ノードの少なくとも一つが、ユーザインターフェースモジュールまたはスクリプトへ結果を返す、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】
前記複数のノードは、２つ以上のノードのサブセットグループに編成される、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つ以上のノードのサブセットグループの少なくとも一つにおいて、前記複数のノードが、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、
前記クラスタノードモジュールのそれぞれが、１以上の他のクラスタノードモジュールおよびシングルノードカーネル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機能をマイクロプロセッサに実現させることを特徴とする、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記憶されるように構成されている、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記憶されるように構成されている、請求項６に記載のコンピュータクラスタ。
【請求項１３】
複数のクラスタノードモジュールと、
前記複数のクラスタノードモジュールの少なくとも１つと通信するクラスタ構成モジュールと、
通信ネットワークを介して互いに接続し、前記クラスタ構成モジュールによって検出可能な複数のプロセッサと、
前記クラスタノードモジュールと接続する複数のカーネルノードと、を有し、
（１）プロセッサ上で実行している前記カーネルモジュールの一つは、前記クラスタ構成モジュールを介して他のプロセッサにアクセスし、他のクラスタノードモジュールを起動して後、相互の通信を確立し、
（２）前記クラスタノードモジュールのそれぞれが、前記カーネルモジュールの一つに接続することを試み、
（３）カーネルモジュールと接続されていない前記クラスタノードモジュールがシャットダウンされる、
コンピュータクラスタ。
【請求項１４】
前記クラスタノードモジュールのそれぞれが、前記カーネルモジュールが見つからないか、または接続されていないかを判定する、請求項１３に記載のコンピュータクラスタ。
【請求項１５】
前記クラスタノードモジュールのそれぞれが、カーネルモジュールとの接続に失敗したことを、前記コンピュータクラスタ上の他のクラスタノードモジュールへ報告する、請求項１４に記載のコンピュータクラスタ。

フロントページの続き

(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・地域又は機関
米国(US)

（特許庁注：以下のものは登録商標）
１．ＵＮＩＸ
２．ＳＯＬＡＲＩＳ

(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
Ｆターム(参考) 5B045 HH06
5B376 AE15　AE42　FA10