(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7066694号
(P7066694)

(45)発行日　令和4年5月13日(2022.5.13)

(24)登録日　令和4年5月2日(2022.5.2)

(51)国際特許分類

		FI		
G06F	*9/50 (2006.01)*	G06F	9/50	150A
G06F	*15/167 (2006.01)*	G06F	15/167	
G06F	*15/173 (2006.01)*	G06F	15/173	

請求項の数　11　（全19頁）

(21)出願番号　特願2019-518274(P2019-518274)
(86)(22)出願日　平成29年10月5日(2017.10.5)
(65)公表番号　特表2019-533249(P2019-533249A)
(43)公表日　令和1年11月14日(2019.11.14)
(86)国際出願番号　PCT/EP2017/075375
(87)国際公開番号　WO2018/065530
(87)国際公開日　平成30年4月12日(2018.4.12)
審査請求日　令和1年8月7日(2019.8.7)
(31)優先権主張番号　16192430.3
(32)優先日　平成28年10月5日(2016.10.5)
(33)優先権主張国・地域又は機関
欧州特許庁(EP)
前置審査

(73)特許権者　507402886
パルテック・クラスター・コンペテンス・センター・ゲゼルシャフト・ミット・ベシュレンクテル・ハフツング
ＰＡＲＴＥＣ ＣＬＵＳＴＥＲ ＣＯＭＰＥＴＥＮＣＥ ＣＥＮＴＥＲ ＧＭＢＨ
ドイツ、８１６７９ ミュンヘン、ポッサートシュトラーセ、２０
(74)代理人　110001195
特許業務法人深見特許事務所
(72)発明者　リッペルト，トーマス
ドイツ、６３７４３ アシャッフェンブルク、ゾンネンシュトラーセ、４３
審査官　井上 宏一

最終頁に続く

(54)【発明の名称】　高性能のコンピューティングシステムおよび方法

(57)【特許請求の範囲】
【請求項１】
アプリケーションプログラムの計算を行なうためのモジュラーコンピューティングシステムであって、
少なくともクラスタモジュールとブースタモジュールとによって形成される異なるモジュールを備え、各モジュールは複数のノードを含み、
前記モジュラーコンピューティングシステムは、前記複数のノードに分散されたモジュラーコンピューティング抽象化層をさらに備え、
前記モジュラーコンピューティング抽象化層は、前記異なるモジュールの前記ノードのために、共有メモリ通信を設定し、前記モジュラーコンピューティング抽象化層によって共有仮想アドレス空間通信を使用するモジュール内およびモジュール間通信ならびに管理機能を提供する、モジュラーコンピューティングシステム。
【請求項２】
前記異なるモジュールは、少なくとも、ストレージモジュールによって形成されていることを特徴とする、請求項１に記載のモジュラーコンピューティングシステム。
【請求項３】
前記モジュラーコンピューティング抽象化層は、前記ノード内に設けられたノードマネージャによって実現されることを特徴とする、請求項１または２に記載のモジュラーコンピューティングシステム。
【請求項４】

前記ノードマネージャは、管理ネットワークを介して互いに通信することを特徴とする、請求項３に記載のモジュラーコンピューティングシステム。
【請求項５】
前記複数のノードは、通信ネットワークを介して通信することを特徴とする、請求項１～４のいずれか１項に記載のモジュラーコンピューティングシステム。
【請求項６】
前記通信ネットワークは、共有メモリ通信を用いて実現されることを特徴とする、請求項５に記載のモジュラーコンピューティングシステム。
【請求項７】
前記ノードマネージャ間の通信は、前記モジュール間の通信から分離されている、請求項３に記載のモジュラーコンピューティングシステム。
【請求項８】
前記ノードマネージャは、前記ノードマネージャの各々が各ノードおよび前記ノード間の各インターコネクトのステータスについての情報のセットを維持するように情報をやり取りする、請求項３に記載のモジュラーコンピューティングシステム。
【請求項９】
前記情報のセットに応じて前記ノードにリソースを割り当てることによって前記システムの動作を動的に調整するように前記情報のセットが用いられる、請求項８に記載のモジュラーコンピューティングシステム。
【請求項１０】
アプリケーションプログラムの計算を行なうためのモジュラーコンピューティングシステムを管理するための方法であって、前記モジュラーコンピューティングシステムは、少なくともクラスタモジュールとブースタモジュールとによって形成される異なるモジュールを備え、各モジュールは複数のノードを含み、前記モジュラーコンピューティングシステムは、前記複数のノードに分散されたモジュラーコンピューティング抽象化層を備え、前記方法は、
前記モジュラーコンピューティング抽象化層を用いて、前記異なるモジュールの前記ノードのために、共有メモリ通信を設定し、前記モジュラーコンピューティング抽象化層によって共有仮想アドレス空間通信を使用するノード内およびノード間通信ならびに管理機能を提供するステップを含む、方法。
【請求項１１】
ノードマネージャをさらに備え、前記方法はさらに、
前記ノードマネージャが、前記ノードのステータスについての情報を収集し、前記収集されたステータスの情報に応じて、前記ノードにリソースを割り当てるステップをさらに含む、請求項１０に記載のモジュラーコンピューティングシステムを管理するための方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、高性能のコンピューティング（ＨＰＣ）の技術分野に関する。特に、本発明はヘテロジニアスコンピューティングシステムに関し、より特定的には、協働して計算タスクを処理するように自由に割り当てられ得る異なるモジュールを含むコンピューティングシステムに関する。モジュールコンピューティング抽象化層（ＭＣＡＬ）と呼ばれる制御エンティティが設けられる。この制御エンティティにより、異なるモジュールによって提供されるさまざまなリソースを動的に割り当てることが可能になる。本発明は、コンピューティングにおけるさまざまな要求に対する調整において柔軟性があるため、クラウドコンピューティングサービスを提供するための基礎となるシステムとしても適用可能である。クラウドコンピューティングサービスは、共有のコンピュータ処理リソースおよびデータを、要求に応じて（多くの場合、インターネットを通じて）コンピュータや他のデバイスに提供するものである。
【背景技術】

【０００２】
ＨＰＣシステムの用途は、幅広い技術分野をカバーする。ほんの数例挙げてみただけでも、人間の脳のシミュレーションを始めとして、天気および気候の予測、コンピュータによる流体工学、地震学的描像、電波天文学、データ解析などがある。高性能のコンピューティングへの要求は、絶えず高まっている。
【０００３】
現在主流のＨＰＣアーキテクチャは、クラスタコンピュータおよび超並列処理（ＭＰＰ）システムである。クラスタコンピュータは、標準ネットワーク技術を用いて互いに通信する複数のクラスタノードを含む。個々のクラスタノードは、既製の汎用プロセッサを備える。ＭＰＰ（超並列処理）システムでは、多数のプロセッサが同時に（したがって、並列して）計算を行なう。それらもまた、通常は既製の汎用プロセッサを使用する。しかしながら、プロセッサは、通信のために専有のネットワーク技術および個別的に適合されたソフトウェアを使用する傾向にある。
【０００４】
より高い計算能力への強い要求を満たすために、コンピュータクラスタ内のクラスタノードの数、およびＭＰＰシステム内のプロセッサの数が近年大幅に増加した。しかしながら、既製の汎用プロセッサを用いてシステム内での数を増やすだけでは解決策にならないことも分かってきた。
【０００５】
既製の汎用プロセッサにも、もちろんプラスの側面はある。それらの単一スレッド性能は高い。マルチコアプロセッサの場合、それらは、プロセッサまたはコア１つ当たり大容量のメモリを有する。標準プログラミングを使用することができるため、これらのシステムにアプリケーションを移植することが容易である。しかしながら、この技術は、限界に達しつつある。既製の汎用プロセッサに関して、そのクロック速度（頻度）は、この１０年で実際的には増加していない。このことは、特にシステムの冷却に関して限界とならないように、チップ１つ当たりのエネルギ消費が１００ワットを大きく超えるべきではないということに密接に関係している。さらに、それらのプロセッサは比較的高価であり、かつエネルギ効率が低い。
【０００６】
解決策は、アクセラレータを採用することに見出された。アクセラレータを用いることによって、非常にスケーラブルなアプリケーションの部分を計算することができる。アプリケーションが費用効果の高い方法で多くのリソースを追加することによって高負荷を処理することができる場合、そのアプリケーションは非常にスケーラブルであると言える。コードにおける並列して計算することのできないシーケンシャル部分が、通常、スケーラビリティへの最も重大な制約要因のうちの１つである。
【０００７】
２種類以上のプロセッサを用いたシステムは、ヘテロジニアスコンピュータシステムと呼ばれる。異なるアクセラレータは、汎用プロセッサ、具体的には、メニーコアプロセッサおよびＧＰＵ（グラフィックス・プロセッシング・ユニット）と組み合わせて使用され得る。メニーコアプロセッサは、５０以上のコアを有する。ＧＰＵは、何百個もの単純なコンピューティングコアを有する。これらのタイプのプロセッサは、ともにエネルギ効率が良い。
【０００８】
ヘテロジニアスコンピュータシステムでは、各クラスタノードが、１つ以上のマルチコアプロセッサまたはＧＰＵによって形成されるアクセラレータを備える。アクセラレータは、基本的には、クラスタノードのプライマリープロセッサの機能を補うコプロセッサとして機能する。クラスタノードは、コンポーネント同士がインターコネクトスイッチを通じてデータを伝送し合うファブリックを介して通信する。「ファブリック」という用語は、高性能のコンピューティングの分野では、システムトポロジーを意味するために使用される。また、インターコネクトという用語は、クラスタノード間の通信インフラストラクチ

ャについて使用され、別々のデバイスを接続する電気的または光学的な接続またはケーブルを意味する。各ノードにアクセラレータを取り付けることの大きな問題は、割り当てが静的であることである。汎用プロセッサとＧＰＵとの比は、システムの設計時に固定される。

【０００９】

国際出願ＷＯ２０１２－０４９２４７Ａ１は、コンピュータクラスタ配列、および、当該紹介されたコンピュータクラスタ配列を動作させるための方法によって、上記欠点に取り組んでいる。このコンピュータクラスタ配列は計算ノードＣＮを備える。これにより、ブースタＢに対する特定の計算タスクを動的に外部調達している。このように、計算ノードＣＮへのブースタＢの割り当て手法が紹介されている。割り当ては、実行時に動的に行なわれる。このアーキテクチャでは、ブースタを形成するクラスタノードおよびアクセラレータノードが１つのファブリックに入る。これは、加速クラスタノードと比較して、より柔軟性がある。なぜなら、プロセスの開始時に、加速処理のためにブースタをクラスタノードに割り当てることができるからである。この配列については、Norbert Eicker等による「メニーコア時代におけるクラスタコンピューティングを追究するＤＥＥＰプロジェクト（The DEEP project Pursuing cluster-computing in the many-core era）」、第４２回並列処理に関する国際会議（International Conference on Parallel Processing）、８８５～８９２頁、２０１３年、および、「メニーコア時代におけるヘテロジニアスクラスタコンピューティングへの代替アプローチであるＤＥＥＰプロジェクト（The DEEP Project An alternative approach to heterogeneous cluster-computing in the many-core era）」、同時実行および計算：実践と経験（Concurrency and Computation: Practice and Experience）、２８、２３９４～２４１１頁、２０１５年にさらに記載されている。

【００１０】

１つのアプリケーションがクラスタブースタ型などのヘテロジニアスシステムによって計算されるとき、個々の計算タスクは、複数の異なるクラスタノード上で処理される必要があるだけでなく、２つの異なるアーキテクチャ、すなわち、コンピュータクラスタ上およびアクセラレータ上で処理される必要がある。クラスタコンピュータ内、およびブースタ内、さらには、クラスタコンピュータとブースタとの間において、効率的な通信が確保されなければならない。さらに、個々のパーツの信頼性が高い場合であっても、コンポーネント数の増加によってコンポーネントが故障するリスクが生じる。したがって、システムは、高い回復力、すなわち、個々のコンポーネントの故障への耐性を確保しつつ、機能性も保つ必要がある。

【発明の概要】

【課題を解決するための手段】

【００１１】

本発明は、モジュラーコンピューティングシステムを提供する。モジュラーコンピューティングシステムは、クラスタモジュール、ブースタモジュール、ストレージモジュール、ニューロモーフィックモジュール、データ解析モジュール、グラフィックスモジュール、量子コンピューティングモジュールなどの異なるモジュールを含む。各モジュールは、複数のノードを含み得る。クラスタモジュールは複数のクラスタノードを含み、ブースタモジュールは複数のブースタノードを含み、ストレージモジュールは複数のストレージノードを含む。その他同様である。

【００１２】

個々のノードは、すべて１つのファブリックに接続されている。異なるモジュールは異なるインターコネクトを有してもよい。クラスタモジュールは、例えばインフィニバンドインターコネクトを使用する。一方、ブースタモジュールは、特殊インターフェースノードを介してインフィニバンドインターコネクトに接続されたインテルＯｍｎｉ－Ｐａｔｈファブリックを使用する。同じことが、他のタイプのモジュールについても当てはまる。

【００１３】

異なるモジュールを結合して１つのシステムとするために、モジュラーコンピューティング抽象化層（ＭＣＡＬ）が設けられる。ＭＣＡＬは、通信機能および管理機能を提供する。Par Tec Cluster Competence Center社によるPara Stationを使用すれば、以下で説明する機能および特性を有するＭＣＡＬを実現するための基盤を形成することができる。

【００１４】

ＭＣＡＬは、モジュール内での通信（モジュール内通信と呼ぶ）と、モジュール間の通信（モジュール間通信と呼ぶ）とを可能にする。そのために、ＭＣＡＬは、それぞれのファブリックにインターフェースを提供する。例えば、システム全体に対してファブリックが１つの場合もあるし、または、モジュール毎にファブリックが特化されている場合もある（例えば、クラスタモジュールにはインフィニバンド、ブースタモジュールにはインテルＯｍｎｉ－Ｐａｔｈファブリック、など）。イーサネット（登録商標）、１０Ｇ、ギガビット、または共有メモリ通信など、他のインターコネクトも可能である。ＭＣＡＬは、如何なる種類の通信を用いてもよく、可用性、アプリケーションプログラムによる指示、性能、またはコストなどの要素に基づいて、特定の通信方法を選択する。

【００１５】

共有メモリ通信は、２つ以上のプロセスが通信するための非常に高速かつ効率的な方法を提供する。異なるタイプの共有メモリ通信が、ＭＣＡＬによって採用され得る。共有メモリを有する同じ物理ノード上で並列タスクの２つ以上のプロセスが実行される場合、これらのプロセス間の共有メモリ通信のために、この共有メモリが使用され得る。これは、典型的には、ＣＰＵ１つ当たり１つのプロセスが生成されるＳＭＰノード上で起こり得る。対称型マルチプロセッシング（ＳＭＰ）は、２つ以上の同一のプロセッサが単一の共有メインメモリに接続しているアーキテクチャを意味する。ＭＣＡＬはプロセスについての知識、および、どのノードでプロセスが行なわれているかについての知識を有しているため、ＭＣＡＬは、そのような状況で共有メモリ通信を確立することができる。

【００１６】

しかしながら、共有メモリ通信はＳＭＰ状況に限定されない。プロセスが１つのモジュール内の異なるノード上で実行される場合、または、異なるモジュールに属するノード上で実行される場合であっても、ＭＣＡＬは、共有の仮想アドレス空間通信を確立する。共有の仮想アドレス空間通信を用いると、メッセージのメタデータおよびデータ自体の仮想アドレスを共有するだけで、サイズの大きなメッセージをノード間で伝達することができる。実際のメッセージは、プロセスにより、その割り当てられた仮想アドレス空間を介してアクセス可能である。したがって、高速通信が可能である。ＭＣＡＬは、通信に関係するプロセスまたはノードによって直接共有されるメモリ領域を設けることによって、仮想アドレス空間通信を確立する。

【００１７】

上記の例から分かるように、ＭＣＡＬは、通信機能を提供するだけでなく、通信方法を選択し、それによって性能および効率を向上させる。ＭＣＡＬは、例えば、プロセスを生成しながら、すなわち、新たな子プロセスを作成および実行しながら、どのインターコネクトが通信に使用可能かを決定することができる。同時に、ＭＣＡＬは、現在行なわれているプロセスを監視し、子プロセスを見つけて終了させるか、または非同期的に実行し続け得る。

【００１８】

ＭＣＡＬが提供する第２の主要な機能は、管理である。管理機能は、さまざまな局面、具体的には、ノード管理、モジュール管理、プロセス管理、およびリソース管理をカバーする。

【００１９】

ノード管理は、個々のノードを対象とする。例えば、クラスタモジュールにおけるクラスタノード、ブースタモジュールにおけるブースタノード、ストレージモジュールにおけるストレージノードなどであり、上で挙げたようなシステム内に存在する他のタイプのモジュールに関しても同様である。初めに、ＭＣＡＬは使用可能なノードを検出する。使用可

能なノードとは、使用できる状態にあるノードのことである。これは、例えば、温度、メモリの空き容量など、および、それらの特性の経時的な変化などのシステムパラメータに基づいて、ＭＣＡＬが、正常に機能していないノードまたは故障寸前のノードを特定するように構成されていることを含む。したがって、ＭＣＡＬはフォールトトレランスを実現し、それによって高い回復力を確保している。

【００２０】

これは、ＭＣＡＬが仮想ノードの概念を採用することによって実現する。仮想ノードは、クラスタノード、ブースタノード、ストレージノードなどの実際のハードウェアを表わすものであるが、ＭＣＡＬは、自由に動的にノードをマップすることができる。例えば、アプリケーションにおいてＸ個のクラスタノード、Ｙ個のブースタノード、およびＺ個のストレージノード（Ｘ、Ｙ、Ｚは整数）が必要である場合、ＭＣＡＬは、すべての利用可能なリソースの中からそれらのノードを提供することができる。ノードが故障している場合、または、ノードがシステムに追加された場合には、ＭＣＡＬは、仮想ノードと物理ノードとの間のマッピングを変更するだけで、迅速に反応することができる。利用可能なノードのリストの中から正常に機能していないノードを取り出すことができ、システム全体が機能不全に陥ることがない。

【００２１】

また、ＭＣＡＬは、到達可能性、レイテンシ、または物理的配置などの要因に応じて、特定のノードを選択することもできる。これは、実行中のアプリケーションに完全に透過的に行なわれ得る。

【００２２】

ＭＣＡＬは、システム全体を見渡し、各ノードおよび各インターコネクトの状態を絶えず最新の状態に維持している。ＭＣＡＬは、ノード１つ当たり利用可能なＣＰＵの数、実行中のサービス、利用可能なストレージ、インターコネクト帯域幅、空きメモリ、温度、システムクロック速度などの、各ノードおよび各インターコネクトの特定の特性について、常時通知される。

【００２３】

もう一つの機能は、モジュール管理である。モジュール管理は、ノード管理に類似しており、特定のモジュールについてのノード管理を含み得るが、モジュールに特有の側面を考慮する。モジュール管理は、モジュールが他のモジュールと通信するために用いるインターフェースノード（設けられる場合）の状態など、モジュールに特有の側面に焦点を当てる。ノードと同様に、ＭＣＡＬは、モジュールの利用増大を可能にするために、モジュールの状態を監視する。

【００２４】

ＭＣＡＬは、プロセス管理も担う。実行中のアプリケーションの各々は、複数の個々のプロセスに分割される。複数の個々のプロセスは、開始、分散、および監視する必要がある。同時に、リソースを割り当てる必要がある。これは、プロセスが計算能力およびストレージを必要とすることに対応する。プロセスが特定の処理を必要とする場合、ＭＣＡＬは、１つ以上のブースタノード、または、ニューロモーフィックノード、データ解析ノード、グラフィックスノード、または、量子コンピューティングノードなどのさらに特定的なノードを、それぞれのモジュールから割り当てる。

【００２５】

そのために、ＭＣＡＬは、プロセスとその要件との間の依存を認識および制御するように構成されている。これに基づいて、ＭＣＡＬは、ブースタモジュールまたはクラスタモジュールの場合にはノード上にプロセスを作成し、または、ストレージモジュールの場合にはストレージを割り当てる。プロセスは監視され、依存プロセスはグループとして扱われる。これにより、並列タスクのうちのいずれか１つで発生するエラーに迅速に応答することが可能である。したがって、プロセスを速やかに再開させることができ、その結果、実行中のアプリケーションの全体の遅延を最小限に抑えることができる。

【００２６】

リソース管理は、ＭＣＡＬがモジュラーコンピューティングシステムにおいて行なうもう一つのタスクである。リソース管理の一部として、ＭＣＡＬは、オフロード機構を実現する。すなわち、例えばクラスタモジュール上で実行中のプロセスは、ブースタモジュール中のいくつかのブースタノードを必要とし得る。そのような場合には、ＭＣＡＬは、タスクを並列して実行するために、選択されたブースタノード上にプロセスを生成することができる。

【００２７】

オフロードの一部として、１つのブースタ上で実行されるように作成されるプロセスは、１つだけでなく複数であってもよい。複数のプロセスは、ホストプロセッサの介入を必要とせずに、モジュール通信ファブリックを用いて互いに直接通信する。

【００２８】

また、リソース管理は、例えば異なるプロセス間の接続数を減少させることによって、メモリ消費を低減することも保証する。これは、必要な時にのみ実際に確立するオンデマンド接続を用いることによって行なわれる。これにより、接続１つ当たりのメモリ消費が劇的に低減される。したがって、ノードなどのリソースは仮想的に扱われるだけでなく、仮想割り当ては物理ネットワークにも適用される。

【００２９】

リソースを割り当てる際に、ＭＣＡＬは、１つのプロセスのリクエストを考慮するだけでなく、他のプロセスのリクエストも考慮する。これにより、統合制御に基づいて通信インフラストラクチャの変更に柔軟に応答することができる。

【００３０】

統合制御は、ＭＣＡＬによって提供されるもう一つの機能局面である。統合制御は、ＭＣＡＬが、管理機能から収集した情報を用いて通信を制御すること（およびその逆）を意味する。このように、通信機能と管理機能とは別個のものではなく、システム性能全体をさらに最適化するために統合されたものである。

【００３１】

すべてのモジュールにおけるすべてのノードのすべての異なる局面、ならびにすべてのインターコネクト、モジュール内通信、およびモジュール間通信のステータスについてのＭＣＡＬの知識を用いて、システムを作動させる。これにより、システムの挙動の変更、および、システム上で実行中のアプリケーションの要求の変更に対する動的な調整が可能になる。

【００３２】

統合制御は、実際のシステム状態およびアプリケーション要求を考慮するだけでなく、次に起こるモジュールのリクエストまたはインターコネクト使用の予測も行なう。

【００３３】

システム状態の情報は、ノードの動的な割り当てのために用いられる。これは、フィードバックループ登録システムの割り当て決定の広範な影響を用いてリソースの割り当てが行なわれることを含む。その結果、将来の割り当てが相応に調整される。また、システムは、自己学習の方法で、そのような情報を記憶することによって将来のケースの割り当て決定を向上させる。

【００３４】

アプリケーションの主要部分が通常はクラスタモジュールで開始されるとしても、ＭＣＡＬは、クラスタモジュールまたはブースタモジュール上でアプリケーションのどの部分を実行させるかを特定する。決定の根拠は、例えば、実行中のコード性能の監視結果をアプリケーションコード自体で示すものである。これにより、アプリケーションのコンピューティング中に割り当てを調整することができる。

【００３５】

ＭＣＡＬの実現例としては、ＭＣＡＬは、システム全体にわたって分散されている。ＭＣＡＬは、各ノード上に実装されるノードマネージャ（ＮＭ）を含む。ノードマネージャはハードウェアで実現されてもよく、またはソフトウェアで実現されてもよく、例えば、そ

れぞれのノード上で実行されるデーモンプロセスとして実現されてもよい。ノードマネージャは、モジュールおよびシステム全体を大域的に見るために、絶えず情報を収集する。システム上で実行中のアプリケーションは、管理機能と通信することによって、または個々のノードマネージャと直接通信することによって、この情報を活用することができる。これは特定のインターフェースによって実現される。この特定のインターフェースは、ハードウェアであってもよいし、ＡＰＩ（アプリケーションプログラミングインターフェース）であってもよい。このように、ＭＣＡＬがアプリケーションから入力を得るだけでなく、アプリケーションも、ＭＣＡＬによって収集された情報を個々のノードマネージャを介して活用する。

【００３６】

ノードマネージャの通信トラフィックは、モジュール内およびモジュール間通信から厳密に分離されている。これにより、モジュール内およびモジュール間通信は、管理機能によって中断されることなく高速で実行される。さらに、このような分離によって、システムセキュリティを高めることができる。

【００３７】

実際の通信ネットワークの使用は、ＭＣＡＬによって制御される。ＭＣＡＬは、この機能をメッセージパッシングインターフェース（ＭＰＩ）を介してアプリケーションに提供する。ＭＣＡＬの一部は、モジュール毎のＭＰＩスタックであるとともに、システム全体に及ぶグローバルＭＰＩである。個々のＭＰＩは、例えば、コンピュータクラスタモジュール、ブースタモジュール、ストレージモジュールなどについて実現される。

【００３８】

ＭＣＡＬの通信は、管理ネットワークを用いる。管理ネットワークは、モジュール内およびモジュール間通信に用いられる物理ネットワークと同じ物理ネットワークであってもよいし、異なる物理ネットワークであってもよい。例えば、モジュール内およびモジュール間通信は、インフィニバンド、ミリネットなどであってもよく、管理ネットワークは、ノードマネージャを接続するイーサネットによって形成されてもよい。

【００３９】

システムには、管理タスクまたは制御機能がシステムを作動させることを可能にするフロントエンドを提供するための特殊ノードが含まれてもよい。フロントエンドが通信ネットワークに接続されることは必要ではないが、通信ネットワークへだけ接続されている。例えばワークステーションコンピュータ、パーソナルコンピュータ（ＰＣ）、ラップトップコンピュータ、または任意のスマートコンピューティングデバイスによって形成され得るフロントエンドノードを介して、モジュラーコンピューティングシステムは管理され得る。モジュラーコンピューティングシステム上で実行中のアプリケーションは、スケジュール、開始、停止、および監視され得る。これは、バッチ処理によって行なわれ得る。すなわち、手動による介入なく、したがって非対話方式で、モジュラーコンピューティングシステム上で一連のジョブを実行することによって行なわれ得る。

【００４０】

手動による介入なくモジュラーコンピューティングシステム上で一連のジョブを実行することに加えて、ＭＣＡＬは、対話型コンピューティングを提供し得る。対話型コンピューティングでは、モジュラーコンピューティングシステム上で実行中のアプリケーションが、その実行中に人からの入力を受け付ける。この入力は、ワークステーションコンピュータ、パーソナルコンピュータ（ＰＣ）、ラップトップコンピュータ、または任意のスマートコンピューティングデバイスを介して提供され得る。そのようなシナリオでは、例えばラップトップコンピュータを介してモジュラーコンピューティングシステムにアクセスし、ラップトップコンピュータがモジュラーコンピューティングシステムの計算能力を有しているかのように見せることができる。これは、即時または現在進行中であるとユーザが感じるような時間枠内での応答が必要なリアルタイムのアプリケーションに非常に有用である。また、膨大な量の構造化データ、半構造化データ、および非構造化データを情報のマイニングのためにアクセス可能にするビッグデータアプリケーションにおいても、その

ような配列を利用することができる。これは、計算能力およびストレージを提供するための基礎となるリソースとしてモジュラーコンピューティングシステムを有するクラウドコンピューティングサービスにも当てはまる。

【発明の効果】

【００４１】

上述のモジュラーコンピューティングシステムにおいて、ＭＣＡＬとシステムとが強固に統合されることによって、通信レイテンシが低減され、通信を向上させることができる。

【図面の簡単な説明】

【００４２】

【図１】本発明に従うモジュラーコンピューティングシステムの機能図である。

【図２】本発明に従うモジュラーコンピューティングシステムの第１の実施形態を示す図である。

【図３】本発明に従うモジュラーコンピューティングシステムの第２の実施形態を示す図である。

【発明を実施するための形態】

【００４３】

図１は、本発明に従うモジュラーコンピューティングシステム１００の機能図である。アプリケーション層１１０がモジュラーコンピューティングシステムの最上部を形成する。モジュラーコンピューティングシステム１００において、アプリケーション層では、ソフトウェアプログラムなどのアプリケーションプログラムが実行され得る。アプリケーション層１１０がシステムのリソースにアクセスすることを可能にするために、モジュラーコンピューティング抽象化層１２０（ＭＣＡＬ）が設けられる。システムのリソースは、クラスタモジュール１３０、ブースタモジュール１４０、ストレージモジュール１５０、データ解析モジュール１６０、および他のモジュール１７０、例えば、ニューロモーフィックモジュール、グラフィックスモジュール、および量子コンピューティングモジュールからなるグループのうちの１つ以上など、さまざまなモジュールによって形成される。同じ種類のモジュールを複数有することも可能である。

【００４４】

ＭＣＡＬ１２０は、モジュール内での通信（モジュール内通信と呼ぶ）と、モジュール間の通信（モジュール間通信と呼ぶ）とを可能にする。ＭＣＡＬ１２０は、如何なる種類の通信を用いてもよく、可用性、アプリケーションプログラムによる指示、性能、またはコストなどの要素に基づいて、特定の通信方法を選択する。

【００４５】

さらに、ＭＣＡＬ１２０は管理機能を提供する。管理機能は、さまざまな局面、具体的には、ノード管理、モジュール管理、プロセス管理、およびリソース管理をカバーする。ノード管理は、個々のノードを対象とするものであり、図２および図３を参照して、さらに説明する。

【００４６】

ＭＣＡＬ１２０は、システム全体を見渡し、モジュール１３０、１４０、１５０、１６０、および１７０の各々の状態を絶えず最新の状態に維持している。ＭＣＡＬ１２０は、利用可能なプロセッサの数、実行中のサービス、利用可能なストレージ、インターコネクト帯域幅、空きメモリ、温度、システムクロック速度などの、モジュール１３０、１４０、１５０、１６０、および１７０の特定の特性について、常時通知される。ＭＣＡＬ１２０が提供するもう一つの機能は、モジュール管理である。モジュール管理は、モジュールに特有の側面を考慮する。したがって、ＭＣＡＬは、クラスタモジュール１３０、ブースタモジュール１４０、ストレージモジュール１５０、データ解析モジュール１６０、および他のモジュール１７０を管理するように特別に適合された機能を含む。ＭＣＡＬ１２０は、プロセス管理も担う。実行中のアプリケーション１１０の各々は、複数の個々のプロセスに分割される。複数の個々のプロセスは、開始、分散、および監視する必要がある。同時に、リソースを割り当てる必要がある。これは、プロセスが計算能力およびストレージ

を必要とすることに対応する。プロセスが特定の処理を必要とする場合、ＭＣＡＬ１２０は、１つ以上のブースタノード、または、ニューロモーフィックノード、データ解析ノード、グラフィックスノード、または、量子コンピューティングノードなどのさらに特定的なノードを、それぞれのモジュールから割り当てる。そのために、ＭＣＡＬ１２０は、プロセスとその要件との間の依存を認識および制御するように構成されている。リソース管理は、ＭＣＡＬ１２０がモジュラーコンピューティングシステムにおいて行なうもう一つのタスクである。リソース管理の一部として、ＭＣＡＬ１２０は、オフロード機構を実現する。すなわち、例えばクラスタモジュール１３０上で実行中のプロセスは、ブースタモジュール１４０中のいくつかのブースタノードを必要とし得る。そのような場合には、ＭＣＡＬ１２０は、タスクを並列して実行するために、ブースタモジュール１４０から選択されたブースタノード上にプロセスを生成することができる。

【００４７】

統合制御は、ＭＣＡＬ１２０によって提供されるもう一つの機能局面である。統合制御は、ＭＣＡＬ１２０が、管理機能から収集した情報を用いて通信を制御すること（およびその逆）を意味する。このように、通信機能と管理機能とは別個のものではなく、システム性能全体をさらに最適化するために統合されたものである。これにより、システムの挙動の変更、および、システム上で実行中のアプリケーションの要求の変更に対する動的な調整が可能になる。統合制御は、実際のシステム状態およびアプリケーション要求を考慮するだけでなく、次に起こるモジュールのリクエストまたはインターコネクト使用の予測も行なう。

【００４８】

図２は、本発明に従うモジュラーコンピューティングシステム２００の第１の実施形態を示す。モジュラーコンピューティングシステム２００は、クラスタモジュール２０２と、ブースタモジュール２０４と、ストレージモジュール２０６とを含む。

【００４９】

クラスタモジュール２０２は、複数のクラスタノード２０８（ＣＮ）を含む。各クラスタノード２０８上には、ノードマネージャ２０９（ＮＭ）が設けられる。ノードマネージャ２０９（ＮＭ）は、他のノードマネージャ２０９と協働して、モジュラーコンピューティングシステム２００に通信機能および管理機能を提供する。ブースタモジュール２０４は、複数のブースタノード２１０（ＢＮ）を含む。各ブースタノード２１０上には、ノードマネージャ２０９が設けられる。ノードマネージャ２０９は、他のノードマネージャ２０９と協働して、モジュラーコンピューティングシステム２００に通信機能および管理機能を提供する。ストレージモジュール２０６は、複数のストレージノード２１２（ＳＮ）を含む。各ストレージノード２１２上には、ノードマネージャ２０９が設けられる。ノードマネージャ２０９は、他のノードマネージャ２０９と協働して、モジュラーコンピューティングシステム２００に通信機能および管理機能を提供する。

【００５０】

クラスタノードは、汎用マイクロプロセッサ（例えば、インテルＸｅｏｎプロセッサ）、メモリ、インターフェース（例えば、ネットワークカード）を有するコンピュータによって形成され得る。ブースタノードは、ＧＰＵまたはメニーコアプロセッサ、メモリ、および、インターフェース（例えば、ネットワークカード）を備える処理システムによって形成され得る。ストレージノードは、ＲＡＭ（ランダムアクセスメモリ）、不揮発性メモリ（例えば、フラッシュメモリ）、ＳＳＤ（ソリッドステートディスク）、ハードディスクなどであり得るメモリを含む。さらに、ストレージノードは、ノードコントローラ（例えば、マイクロプロセッサ）、およびインターフェース（例えば、ネットワークカード）を含む。

【００５１】

２つの異なるネットワーキングインフラストラクチャが設けられる。１つ目は、クラスタノード２０８、ブースタノード２１０、およびストレージノード２１２が通信可能になるように、すべてのノード２０８、２１０、２１２を互いに接続する通信ネットワーク２２

０である。
【００５２】
２つ目は、さまざまなノード２０８、２１０、および２１２において設けられたすべてのノードマネージャ２０９を接続するように提供される管理ネットワーク２２４である。モジュラーコンピューティングシステム全体の制御を可能にするフロントエンド２３０が、管理ネットワーク２２４に取り付けられている。フロントエンド２３０もまた、ノードマネージャ２０９を含む。このノードマネージャ２０９は、他のノードマネージャ２０９への実際の通信を行なう。
【００５３】
異なるモジュールを結合して１つのシステムとするために、図１を参照して説明したモジュラーコンピューティング抽象化層（ＭＣＡＬ）が設けられる。ＭＣＡＬは、通信機能および管理機能を提供する。Par Tec Cluster Competence Center社によるPara Stationを使用すれば、本明細書で説明する機能および特性を有するＭＣＡＬを実現するための基盤を形成することができる。
【００５４】
ＭＣＡＬは、個々のノード上のノードマネージャ２０９によって実現され、モジュール内での通信（モジュール内通信と呼ぶ）と、モジュール間の通信（モジュール間通信と呼ぶ）とを可能にする。そのために、ノードマネージャは、それぞれのファブリックにインターフェースを提供する。例えば、システム全体に対してファブリックが１つの場合もあるし、または、モジュール毎にファブリックが特化されている場合もある（例えば、クラスタモジュール２０２にはインフィニバンド、ブースタモジュール２０４にはインテルＯｍｎｉ－Ｐａｔｈファブリック、など）。イーサネット、１０Ｇ、ギガビット、または共有メモリ通信など、他のインターコネクトも可能である。ノードマネージャ３０９は、如何なる種類の通信を用いてもよく、可用性、アプリケーションプログラムによる指示、性能、またはコストなどの要素に基づいて、特定の通信方法を選択する。
【００５５】
共有メモリ通信および共有の仮想アドレス空間通信が採用されてもよい。これらは、２つ以上のプロセスが通信するための非常に高速かつ効率的な方法を提供する。
【００５６】
ノードマネージャ２０９は、一括してノード管理を提供する。ノード管理は、個々のノードを対象とする。例えば、クラスタモジュール２０２におけるクラスタノード２０８、ブースタモジュール２０４におけるブースタノード２１０、ストレージモジュール２０６におけるストレージノード２１２などである。
【００５７】
さらに、ノードマネージャ２０９は、到達可能性、レイテンシまたは物理的配置などの要因に応じて、特定のノードを選択する。これは、実行中のアプリケーションに完全に透過的に行なわれ得る。ノードマネージャ２０９は、システム全体についての情報を収集およびやり取りし、各ノードおよび各インターコネクトの状態を絶えず最新の状態に維持している。これにより、ノードマネージャ３０９は、ノード１つ当たり利用可能なＣＰＵの数、実行中のサービス、利用可能なストレージ、インターコネクト帯域幅、空きメモリ、温度、システムクロック速度などの、各ノードおよび各インターコネクトの特定の特性について、常時通知される。
【００５８】
ノードマネージャによって実施されるもう一つの機能は、モジュール管理である。モジュール管理は、ノード管理に類似しており、特定のモジュールについてのノード管理を含み得るが、モジュールに特有の側面を考慮する。モジュール管理は、モジュールが他のモジュールと通信するために用いるインターフェースノード（設けられる場合）の状態など、モジュールに特有の側面に焦点を当てる。ノードと同様に、ノードマネージャ３０９は、モジュールの利用増大を可能にするために、モジュールの状態を監視する。
【００５９】

リソース管理は、ノードマネージャ２０９がモジュラーコンピューティングシステム２００において行なうもう一つのタスクである。リソース管理の一部として、ノードマネージャ２０９は、オフロード機構を実現する。すなわち、例えばクラスタモジュール２０２上で実行中のプロセスは、ブースタモジュール２０４中のいくつかのブースタノード２１０を必要とし得る。そのような場合には、ノードマネージャは、タスクを並列して実行するために、選択されたブースタノード２１０上にプロセスを生成することができる。

【００６０】

統合制御は、ノードマネージャによって実施されるもう一つの機能局面である。統合制御は、ノードマネージャが、管理機能から収集した情報を用いて通信を制御すること（およびその逆）を意味する。このように、通信機能と管理機能とは別個のものではなく、システム性能全体をさらに最適化するために統合されたものである。

【００６１】

すべてのモジュールにおけるすべてのノードのすべての異なる局面、ならびにすべてのインターコネクト、モジュール内通信、およびモジュール間通信のステータスについてのノードマネージャ２０９の知識を用いて、システムを作動させる。これにより、システムの挙動の変更、および、システム上で実行中のアプリケーションの要求の変更に対する動的な調整が可能になる。

【００６２】

統合制御は、実際のシステム状態およびアプリケーション要求を考慮するだけでなく、次に起こるモジュールのリクエストまたはインターコネクト使用の予測も行なう。

【００６３】

ノードマネージャ２０９はハードウェアで実現されてもよく、またはソフトウェアで実現されてもよく、例えば、それぞれのノード上で実行されるデーモンプロセスとして実現されてもよい。ノードマネージャ２０９は、モジュールおよびシステム全体を大域的に見るために、絶えず情報を収集する。システム上で実行中のアプリケーションは、管理機能と通信することによって、または個々のノードマネージャ２０９と直接通信することによって、この情報を活用することができる。これは、特定のインターフェースによって実現される。この特定のインターフェースは、ハードウェアであってもよいし、ＡＰＩ（アプリケーションプログラミングインターフェース）であってもよい。

【００６４】

ノードマネージャ２０９の通信トラフィックは、モジュール内およびモジュール間通信から厳密に分離されている。これにより、モジュール内およびモジュール間通信は、管理機能によって中断されることなく高速で実行される。さらに、このような分離によって、システムセキュリティを高めることができる。

【００６５】

実際の通信ネットワーク２２０の使用は、ノードマネージャ２０９によって制御される。ノードマネージャ２０９は、この機能をメッセージパッシングインターフェース（ＭＰＩ）を介してアプリケーションに提供する。

【００６６】

ノードマネージャ２０９は、管理ネットワーク２２４を用いて互いに通信する。管理ネットワーク２２４は、モジュール内およびモジュール間通信に用いられる物理ネットワーク（ここでは、通信ネットワーク２２０と称する）と同じ物理ネットワークであってもよいし、異なる物理ネットワークであってもよい。例えば、モジュール内およびモジュール間通信は、インフィニバンド、ミリネットなどであってもよく、管理ネットワークは、ノードマネージャ２０９を接続するイーサネットによって形成されてもよい。

【００６７】

システムには、管理タスクまたは制御機能がシステムを作動させることを可能にするフロントエンド２３０を提供するための特殊ノードが含まれてもよい。フロントエンドが通信ネットワーク２２０に接続されることは必要ではないが、通信ネットワーク２２０へだけ接続されている。例えばワークステーションコンピュータ、パーソナルコンピュータ（Ｐ

Ｃ）、ラップトップコンピュータ、または任意のスマートコンピューティングデバイスによって形成され得るフロントエンドノードを介して、モジュラーコンピューティングシステム２００は管理され得る。モジュラーコンピューティングシステム２００上で実行中のアプリケーションは、スケジュール、開始、停止、および監視され得る。これは、バッチ処理によって行なわれ得る。すなわち、手動による介入なく、したがって非対話方式で、モジュラーコンピューティングシステム２００上で一連のジョブを実行することによって行なわれ得る。

【００６８】

手動による介入なくモジュラーコンピューティングシステム２００上で一連のジョブを実行することに加えて、ノードマネージャ３０９は、対話型コンピューティングを提供し得る。対話型コンピューティングでは、モジュラーコンピューティングシステム２００上で実行中のアプリケーションが、その実行中に人からの入力を受け付ける。この入力は、ワークステーションコンピュータ、パーソナルコンピュータ（ＰＣ）、ラップトップコンピュータ、または任意のスマートコンピューティングデバイスを介して提供され得る。そのようなシナリオでは、例えばラップトップコンピュータを介してモジュラーコンピューティングシステム２００にアクセスし、ラップトップコンピュータがモジュラーコンピューティングシステム２００の計算能力を有しているかのように見せることができる。これは、即時または現在進行中であるとユーザが感じるような時間枠内での応答が必要なリアルタイムのアプリケーションに非常に有用である。また、膨大な量の構造化データ、半構造化データ、および非構造化データを情報のマイニングのためにアクセス可能にするビッグデータアプリケーションにおいても、そのような配列を利用することができる。これは、計算能力およびストレージを提供するための基礎となるリソースとしてモジュラーコンピューティングシステム２００を有するクラウドコンピューティングサービスにも当てはまる。

【００６９】

図３は、本発明に従うモジュラーコンピューティングシステム３００の第２の実施形態を示す。モジュラーコンピューティングシステム３００は、クラスタモジュール３０２と、ブースタモジュール３０４と、ストレージモジュール３０６とを含む。

【００７０】

クラスタモジュール３０２は、複数のクラスタノード３１０（ＣＮ）を含む。各クラスタノード３１０上には、ノードマネージャ３０９（ＮＭ）が設けられる。ノードマネージャ３０９（ＮＭ）は、他のノードマネージャ３０９と協働して、モジュラーコンピューティングシステム３００に通信機能および管理機能を提供する。クラスタノード３１０は、クラスタインターコネクト３１２を介して互いに接続されている。

【００７１】

ブースタモジュール３０４は、複数のブースタノード３２０（ＢＮ）を含む。各ブースタノード３２０上には、ノードマネージャ３０９が設けられる。ノードマネージャ３０９は、他のノードマネージャ３０９と協働して、モジュラーコンピューティングシステム３００に通信機能および管理機能を提供する。ブースタノード３２０は、ブースタインターコネクト３２２を介して互いに接続されている。

【００７２】

ストレージモジュール３０６は、複数のストレージノード３３０（ＳＮ）を含む。各ストレージノード３３０上には、ノードマネージャ３０９が設けられる。ノードマネージャ３０９は、他のノードマネージャ３０９と協働して、モジュラーコンピューティングシステム３００に通信機能および管理機能を提供する。ストレージノード３３０は、ストレージインターコネクト３３２を介して互いに接続されている。

【００７３】

クラスタノードは、汎用マイクロプロセッサ（例えば、インテルＸｅｏｎプロセッサ）、メモリ、インターフェース（例えば、ネットワークカード）を有するコンピュータによって形成され得る。ブースタノードは、ＧＰＵまたはメニーコアプロセッサ、メモリ、およ

び、インターフェース（例えば、ネットワークカード）を備える処理システムによって形成され得る。ストレージノードは、ＲＡＭ（ランダムアクセスメモリ）、不揮発性メモリ（例えば、フラッシュメモリ）、ＳＳＤ（ソリッドステートディスク）、ハードディスクなどであり得るメモリを含む。さらに、ストレージノードは、ノードコントローラ（例えば、マイクロプロセッサ）、およびインターフェース（例えば、ネットワークカード）を含む。

【００７４】

クラスタモジュール３０２は、クラスタインターコネクト３１２とストレージインターコネクト３３２とを接続するインターフェースノード３４０を介して、ストレージモジュール３０６と通信可能である。ストレージモジュール３０６は、ストレージインターコネクト３３２とブースタインターコネクト３２２とを接続するインターフェースノード３４２を介して、ブースタモジュール３０４と通信可能である。ブースタモジュール３０４は、ブースタインターコネクト３２２とクラスタインターコネクト３１２とを接続するインターフェースノード３４４を介して、クラスタモジュール３０２と通信可能である。

【００７５】

クラスタモジュール３０２は、例えばインフィニバンドインターコネクトを使用する。一方、ブースタモジュール３０４は、インターフェースノード３４４を介してインフィニバンドインターコネクトに接続されたインテルＯｍｎｉ－Ｐａｔｈファブリックを使用する。性能およびスループットを向上させるために、各インターフェースノードは複数のノードによって形成されてもよい。同じことが、他のタイプのモジュールについても当てはまる。

【００７６】

さまざまなノード３１０、３１０、３１２において設けられたすべてのノードマネージャ３０９は、同じ通信インフラストラクチャを使用する。モジュラーコンピューティングシステム全体の制御を可能にするフロントエンド３５０が、クラスタインターコネクト３１２に取り付けられている。フロントエンド３５０もまた、ノードマネージャ３０９を含む。このノードマネージャ３０９は、他のノードマネージャ３０９への実際の通信を行なう。

【００７７】

異なるモジュールを結合して１つのシステムとするために、図１を参照して説明したモジュラーコンピューティング抽象化層（ＭＣＡＬ）が設けられる。ＭＣＡＬは、通信機能および管理機能を提供する。Par Tec Cluster Competence Center社によるPara Stationを使用すれば、以下で説明する機能および特性を有するＭＣＡＬを実現するための基盤を形成することができる。

【００７８】

ＭＣＡＬは、各ノードにおいてノードマネージャ３０９によって実現される。プロセス間の通信のために、提供される如何なる種類の通信リンクを用いてもよい。

【００７９】

前述のように、ノードマネージャ３０９はノード管理も提供する。ノード管理は、個々のノードを対象とする。例えば、クラスタモジュール３０２におけるクラスタノード３１０、ブースタモジュール３０４におけるブースタノード３２０、ストレージモジュール３０６におけるストレージノード３３０などである。また、ノードマネージャ３０９は、到達可能性、レイテンシ、または物理的配置などの要因に応じて、特定のノードを選択することもできる。これは、実行中のアプリケーションに完全に透過的に行なわれ得る。ノードマネージャ３０９は、常にシステム全体を見渡し、各ノードおよび各インターコネクトの状態を絶えず最新の状態に維持している。ノードマネージャ３０９は、ノード１つ当たり利用可能なＣＰＵの数、実行中のサービス、利用可能なストレージ、インターコネクト帯域幅、空きメモリ、温度、システムクロック速度などの、各ノードおよび各インターコネクトの特定の特性について、常時、互いに通知し合う。

【００８０】

ノードマネージャ３０９が実施するもう一つの機能は、モジュール管理である。モジュー

ル管理は、ノード管理に類似しており、特定のモジュールについてのノード管理を含み得るが、モジュールに特有の側面を考慮する。モジュール管理は、モジュールが他のモジュールと通信するために用いるインターフェースノード（設けられる場合）の状態など、モジュールに特有の側面に焦点を当てる。

【００８１】

ノードマネージャ３０９は、プロセス管理も実施する。実行中のアプリケーションの各々は、複数の個々のプロセスに分割される。複数の個々のプロセスは、開始、分散、および監視する必要がある。同時に、リソースを割り当てる必要がある。これは、プロセスが計算能力およびストレージを必要とすることに対応する。ノードマネージャ３０９は、プロセスとその要件との間の依存を認識および制御するように構成されている。これに基づいて、ノードマネージャ３０９は、ブースタモジュール３０４またはクラスタモジュール３０２の場合にはノード上にプロセスを作成し、または、ストレージモジュール３０６の場合にはストレージを割り当てる。プロセスは監視され、依存プロセスはグループとして扱われる。これにより、並列タスクのうちのいずれか１つで発生するエラーに迅速に応答することが可能である。したがって、プロセスを速やかに再開させることができ、その結果、実行中のアプリケーションの全体の遅延を最小限に抑えることができる。

【００８２】

リソース管理は、ノードマネージャ３０９がモジュラーコンピューティングシステム３００において実施するもう一つのタスクである。リソース管理の一部として、ノードマネージャ３０９は、オフロード機構を実現する。すなわち、例えばクラスタモジュール３０２上で実行中のプロセスは、ブースタモジュール３０４中のいくつかのブースタノード３２０を必要とし得る。そのような場合には、ノードマネージャ３０９は、タスクを並列して実行するために、選択されたブースタノード３２０上にプロセスを生成する。

【００８３】

オフロードの一部として、１つのブースタ上で実行されるように作成されるプロセスは、１つだけでなく複数であってもよい。複数のプロセスは、ホストプロセッサの介入を必要とせずに、モジュール通信ファブリックを用いて互いに直接通信する。

【００８４】

また、リソース管理は、例えば異なるプロセス間の接続数を減少させることによって、メモリ消費を低減することも保証する。これは、必要な時にのみ実際に確立するオンデマンド接続を用いることによって行なわれる。これにより、接続１つ当たりのメモリ消費が劇的に低減される。したがって、ノードなどのリソースは仮想的に扱われるだけでなく、仮想割り当ては物理ネットワークにも適用される。

【００８５】

リソースを割り当てる際に、ノードマネージャ３０９は、１つのプロセスのリクエストを考慮するだけでなく、他のプロセスのリクエストも考慮する。これにより、統合制御に基づいて通信インフラストラクチャの変更に柔軟に応答することができる。

【００８６】

統合制御は、ノードマネージャ３０９によって提供されるもう一つの機能局面である。統合制御は、ノードマネージャが、管理機能から収集した情報を用いて通信を制御すること（およびその逆）を意味する。このように、通信機能と管理機能とは別個のものではなく、システム性能全体をさらに最適化するために統合されたものである。

【００８７】

すべてのモジュールにおけるすべてのノードのすべての異なる局面、ならびにすべてのインターコネクト、モジュール内通信、およびモジュール間通信のステータスについてのノードマネージャの知識を用いて、システムを作動させる。これにより、システムの挙動の変更、および、システム上で実行中のアプリケーションの要求の変更に対する動的な調整が可能になる。

【００８８】

統合制御は、実際のシステム状態およびアプリケーション要求を考慮するだけでなく、次

に起こるモジュールのリクエストまたはインターコネクト使用の予測も行なう。システム状態の情報は、ノードの動的な割り当てのために用いられる。これは、フィードバックループ登録システムの割り当て決定の広範な影響を用いてリソースの割り当てが行なわれることを含む。その結果、将来の割り当てが相応に調整される。また、システムは、自己学習の方法で、そのような情報を記憶することによって将来のケースの割り当て決定を向上させる。

【００８９】

アプリケーションの主要部分が通常はクラスタモジュール３０２で開始されるとしても、ノードマネージャは、クラスタモジュール３０２またはブースタモジュール３０４上でアプリケーションのどの部分を実行させるかを特定する。決定の根拠は、例えば、実行中のコード性能の監視結果をアプリケーションコード自体で示すものである。これにより、アプリケーションのコンピューティング中に割り当てを調整することができる。

【００９０】

ノードマネージャ３０９はハードウェアで実現されてもよく、またはソフトウェアで実現されてもよく、例えば、それぞれのノード上で実行されるデーモンプロセスとして実現されてもよい。ノードマネージャ３０９は、モジュールおよびシステム全体を大域的に見るために、絶えず情報を収集する。システム上で実行中のアプリケーションは、管理機能と通信することによって、または個々のノードマネージャ３０９と直接通信することによって、この情報を活用することができる。これは特定のインターフェースによって実現される。この特定のインターフェースは、ハードウェアであってもよいし、ＡＰＩ（アプリケーションプログラミングインターフェース）であってもよい。

【００９１】

ノードマネージャの通信トラフィックは、モジュール内およびモジュール間通信から厳密に分離されている。これにより、モジュール内およびモジュール間通信は、管理機能によって中断されることなく高速で実行される。さらに、このような分離によって、システムセキュリティを高めることができる。

【００９２】

実際の通信ネットワークの使用は、ノードマネージャ３０９によって制御される。ノードマネージャ３０９は、この機能をメッセージパッシングインターフェース（ＭＰＩ）を介してアプリケーションに提供する。ノードマネージャ３０９は、管理ネットワーク（図示せず）を用いる。管理ネットワーク（図示せず）は、モジュール内およびモジュール間通信に用いられる物理ネットワークと同じ物理ネットワークであってもよいし、異なる物理ネットワークであってもよい。例えば、モジュール内およびモジュール間通信は、インフィニバンド、ミリネットなどであってもよく、管理ネットワーク（図示せず）は、ノードマネージャ３０９を接続するイーサネットによって形成されてもよい。

【００９３】

システムには、管理タスクまたは制御機能がシステムを作動させることを可能にするフロントエンドを提供するための特殊ノードが含まれてもよい。フロントエンド３５０が通信ネットワークに接続されることは必要ではないが、通信ネットワークへだけ接続されている。例えばワークステーションコンピュータ、パーソナルコンピュータ（ＰＣ）、ラップトップコンピュータ、または任意のスマートコンピューティングデバイスによって形成され得るフロントエンド３５０ノードを介して、モジュラーコンピューティングシステム３００は管理され得る。モジュラーコンピューティングシステム３００上で実行中のアプリケーションは、スケジュール、開始、停止、および監視され得る。これは、バッチ処理によって行なわれ得る。すなわち、手動による介入なく、したがって非対話方式で、モジュラーコンピューティングシステム３００上で一連のジョブを実行することによって行なわれ得る。

【００９４】

手動による介入なくモジュラーコンピューティングシステム３００上で一連のジョブを実行することに加えて、ノードマネージャは、対話型コンピューティングを提供し得る。対

話型コンピューティングでは、モジュラーコンピューティングシステム３００上で実行中のアプリケーションが、その実行中に人からの入力を受け付ける。この入力は、ワークステーションコンピュータ、パーソナルコンピュータ（ＰＣ）、ラップトップコンピュータ、または任意のスマートコンピューティングデバイスを介して提供され得る。そのようなシナリオでは、例えばラップトップコンピュータを介してモジュラーコンピューティングシステム３００にアクセスし、ラップトップコンピュータがモジュラーコンピューティングシステム３００の計算能力を有しているかのように見せることができる。これは、即時または現在進行中であるとユーザが感じるような時間枠内での応答が必要なリアルタイムのアプリケーションに非常に有用である。また、膨大な量の構造化データ、半構造化データ、および非構造化データを情報のマイニングのためにアクセス可能にするビッグデータアプリケーションにおいても、そのような配列を利用することができる。これは、計算能力およびストレージを提供するための基礎となるリソースとしてモジュラーコンピューティングシステム３００を有するクラウドコンピューティングサービスにも当てはまる。

【図面】

【図１】

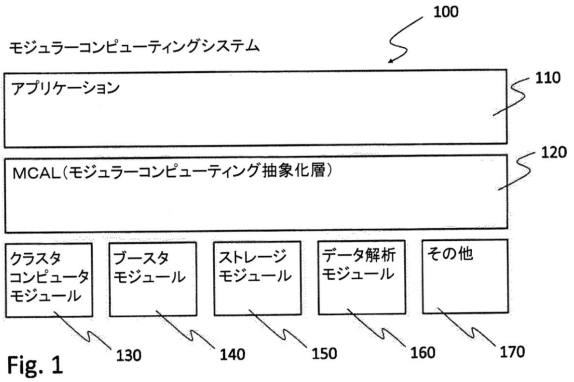

Fig. 1

【図２】

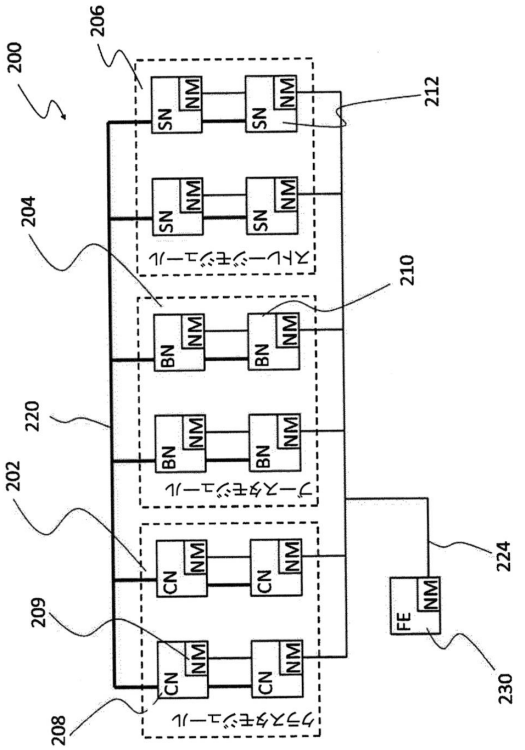

Fig. 2

【図３】

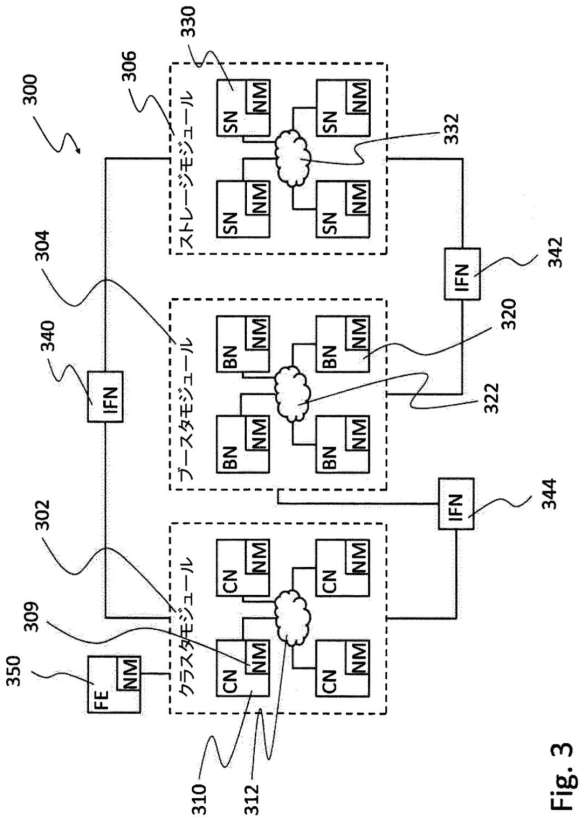
300
302
304
306
309
310
312
320
322
330
332
340
342
344
350
ストレージモジュール
ブースタモジュール
クラスタモジュール
SN
BN
CN
NM
IFN
FE

Fig. 3

フロントページの続き

(56)参考文献 特表２０１３－５３９８８１（ＪＰ，Ａ）
特開２０１６－０１２３４７（ＪＰ，Ａ）
特開２０１２－０４８３３０（ＪＰ，Ａ）
特表２０１３－５１５９９１（ＪＰ，Ａ）

(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／５０
Ｇ０６Ｆ　１５／１６７
Ｇ０６Ｆ　１５／１７３