

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4451806号
(P4451806)

(45) 発行日 平成22年4月14日(2010.4.14)

(24) 登録日 平成22年2月5日(2010.2.5)

(51) Int.Cl.

F I

G O 6 F 15/80 (2006.01)

G O 6 F 15/80

請求項の数 26 外国語出願 (全 97 頁)

(21) 出願番号	特願2005-117406 (P2005-117406)	(73) 特許権者	503455363
(22) 出願日	平成17年4月14日(2005.4.14)		レイセオン カンパニー
(65) 公開番号	特開2006-146865 (P2006-146865A)		アメリカ合衆国 マサチューセッツ州 O
(43) 公開日	平成18年6月8日(2006.6.8)		2451 ウォルサム ウィンター スト
審査請求日	平成17年6月13日(2005.6.13)		リート 870
(31) 優先権主張番号	10/991,994	(74) 代理人	100070150
(32) 優先日	平成16年11月17日(2004.11.17)		弁理士 伊東 忠彦
(33) 優先権主張国	米国 (US)	(74) 代理人	100091214
			弁理士 大貫 進介
前置審査		(74) 代理人	100107766
			弁理士 伊東 忠重
		(72) 発明者	シャノン ヴイ デイヴィッドソン
			アメリカ合衆国 ミズーリ州 63050
			-3822 ヒルズボロ ジャーヴィス・
			ロード 3484
			最終頁に続く

(54) 【発明の名称】 高性能計算 (HPC) システムにおけるオン・デマンド・インスタンスエーション

(57) 【特許請求の範囲】

【請求項1】

高性能計算 (HPC) システムにおけるオン・デマンド・インスタンスエーションのシステムであって：

第1ポート番号と第1ホスト名とを規定するクライアント端末からの接続要求をHPCサーバで受信する手段であって、前記第1ポート番号と前記第1ホスト名は、前記HPCサーバに対して外部に共用宣言され、前記HPCサーバはノード・クラスタを備える手段と、

前記第1ポート番号と前記第1ホスト名とに対応する前記HPCサーバでのサービスを識別する手段と、

前記識別サービスが、前記クライアント端末に対して利用可能か否かを判定し、前記識別サービスが利用可能な場合、前記HPCサーバにより、前記クラスタにおける1つ又は複数のノードの何れかでの前記識別サービスを提供するホストをインスタンス化する手段とを備え、前記ホストは、前記識別サービスを包含するアプリケーションを含むことを特徴とするシステム。

【請求項2】

請求項1記載のシステムであって、前記接続要求が：

伝送制御プロトコル (TCP) 接続要求又はユーザ・データグラム・プロトコル (UDP) 接続要求であることを特徴とするシステム。

【請求項3】

請求項 1 記載のシステムであって、

サービスと、及び前記サービスに対応する前記 H P C サーバに対して外部に共用宣言されるポート番号及びホスト名とをそれぞれが規定する複数のエントリを備える、前記 H P C サーバにあるサービス・リストをアクセスする手段と、

前記 H P C サーバにある前記サービス・リストによって、前記第 1 のポート番号及び前記第 1 のホスト名に対応する前記 H P C サーバでの前記サービスを識別する手段と更に備えるシステム。

【請求項 4】

請求項 1 記載のシステムであって、

サービスと、

前記クラスタにおける 1 つ又は複数のノードで実行されると、前記サービスを提供するホストに対応する、前記 H P C サーバの内部にあるポート番号及びホスト名とをそれぞれが規定する複数のエントリを備える、前記 H P C サーバにあるサービス・リストをアクセス手段と、

前記 H P C サーバでの前記サービス・リストによって、前記クラスタにおける 1 つ又は複数のノードの何れかで前記識別サービスを提供する前記ホストをインスタンス化する手段とを更に備えるシステム。

【請求項 5】

請求項 1 記載のシステムであって、

サービスと、前記サービスが利用可能か否かを示す 1 つ又は複数の規則をそれぞれが規定する複数のエントリを備える、前記 H P C サーバでのサービス・リストをアクセスする手段と、

前記サービス・リストに応じて、前記識別リストが利用可能か否かを判定する手段とを更に備えるシステム。

【請求項 6】

請求項 5 記載のシステムであって、前記規則は：

前記サービスが利用可能か否かを、前記クライアント端末からの前記接続要求に関連した時間に基づいて示すことを特徴とするシステム。

【請求項 7】

請求項 1 記載のシステムであって、

前記識別サービスが利用可能でない旨の判定に応じて、前記第 1 ポート番号に対応する前記 H P C サーバ上のポートをブロック化して、前記識別サービスを前記クライアント端末がアクセスすることを妨げる手段を更に備えることを特徴とするシステム。

【請求項 8】

請求項 1 記載のシステムであって：

前記クラスタにおける 1 つ又は複数のノードで前記識別サービスを提供する前記ホストに対応する前記 H P C サーバで、ブート画像、ファイル・システム、及びオペレーティング・システム（O S）構成ファイルにアクセスする手段と、

前記ブート画像、前記ファイル・システム及び前記 O S 構成ファイルを用いて、前記クラスタにおける 1 つ又は複数のノードで前記識別サービスを提供する前記ホストをインスタンス化する手段とを更に備えることを特徴とするシステム。

【請求項 9】

請求項 1 記載のシステムであって、前記識別サービスが：

ウェブ・サーバであることを特徴とするシステム。

【請求項 10】

請求項 1 記載のシステムであって、

前記クラスタにおける 1 つ又は複数のノードでの前記識別サービスを提供する前記ホストのアドレスをスプーフィングし、前記ホストのために、前記ホストがブートしている間に接続シーケンスを開始する手段を更に備えることを特徴とするシステム。

【請求項 11】

請求項 10 記載のシステムであって：

前記アドレスが、インターネット・プロトコル・オーバ媒体アクセス制御（IP/MAC）アドレスであり；かつ

前記接続シーケンスが、伝送制御プロトコル・オーバ IP（TCP/IP）接続シーケンス又は、ユーザ・データグラム・プロトコル（UDP）オーバ IP（UDP/IP）接続シーケンスであることを特徴とするシステム。

【請求項 12】

請求項 1 記載のシステムであって、前記ノード・クラスタが：

3 次元のトラスを備えるトポロジによって互いに結合される複数のノード；

を備えることを特徴とするシステム。

10

【請求項 13】

請求項 1 記載のシステムであって、前記ノード・クラスタにおけるノードが：

2 つのスイッチに結合された中央処理装置（CPU）であることを特徴とするシステム

。

【請求項 14】

高性能計算（HPC）システムにおけるオン・デマンド・インスタンスーションの方法であって：

第 1 ポート番号と第 1 ホスト名とを規定するクライアント端末からの接続要求を HPC サーバで受信する工程であって、前記第 1 ポート番号と前記第 1 ホスト名が、HPC サーバに対して外部に共用宣言され、前記 HPC サーバがノード・クラスタを備える工程と、

20

前記第 1 ポート番号と前記第 1 ホスト名とに対応する前記 HPC サーバでのサービスを識別する工程と、

前記識別サービスが、前記クライアント端末に対して利用可能か否かを判定する工程と

、

前記識別サービスが利用可能な場合、前記 HPC サーバにより、前記クラスタにおける 1 つ又は複数のノードの何れかでの前記識別サービスを提供するホストをインスタンス化する工程とを備え、前記ホストが、前記識別サービスを包含するアプリケーションを含むことを特徴とする方法。

【請求項 15】

請求項 14 記載の方法であって、前記接続要求が、伝送制御プロトコル（TCP）接続要求又はユーザ・データグラム・プロトコル（UDP）接続要求であることを特徴とする方法。

30

【請求項 16】

請求項 14 記載の方法であって、

サービスと、

前記サービスに対応する前記 HPC サーバに対して外部に共用宣言される、ポート番号及びホスト名と

をそれぞれが規定する複数のエントリを備える、前記 HPC サーバでのサービス・リストをアクセスする工程と、

前記 HPC サーバでの前記サービス・リストによって、前記第 1 ポート番号と前記第 1 ホスト名とに対応する前記 HPC サーバでの前記サービスを識別する工程とを更に備えることを特徴とする方法。

40

【請求項 17】

請求項 14 記載の方法であって、

サービスと、

前記クラスタにおける 1 つ又は複数のノードで実行されると、前記サービスを提供するホストに対応する前記 HPC サーバの内部にあるポート番号及びホスト名

とをそれぞれが規定する複数のエントリを備える、前記 HPC サーバでのサービス・リストをアクセスする工程と、

前記 HPC サーバでの前記サービス・リストによって、前記クラスタにおける 1 つ又は

50

複数のノードでの前記識別サービスを提供する前記ホストをインスタンス化する工程とを更に備えることを特徴とする方法。

【請求項 18】

請求項 14 記載の方法であって、

サービスと、前記サービスが利用可能か否かを示す 1 つ又は複数の規則とをそれぞれが規定する複数のエントリを備える、前記 H P C サーバでのサービス・リストをアクセスする工程と、

前記サービス・リストによって、前記識別サービスが利用可能か否かを判定する工程とを更に備えることを特徴とする方法。

【請求項 19】

請求項 18 記載の方法であって、前記規則は：

前記サービスが利用可能か否かを、前記クライアント端末からの前記接続要求に関連した時間に基づいて示すことを特徴とする方法。

【請求項 20】

請求項 14 記載の方法であって、

前記識別サービスが利用可能でない旨の判定に応じて、前記第 1 ポート番号に対応する前記 H P C サーバ上のポートをブロック化して、前記識別サービスを前記クライアント端末がアクセスすることを妨げる工程を更に備える方法。

【請求項 21】

請求項 14 記載の方法であって、

前記クラスタにおける 1 つ又は複数のノードで前記識別サービスを提供する前記ホストに対応する前記 H P C サーバで、ブート画像、ファイル・システム、及びオペレーティング・システム（O S）構成ファイルにアクセス工程と、

前記ブート画像、前記ファイル・システム及び前記 O S 構成ファイルを用いて、前記クラスタにおける 1 つ又は複数のノードで前記識別サービスを提供する前記ホストをインスタンス化する工程とを備えることを特徴とする方法。

【請求項 22】

請求項 14 記載の方法であって、前記識別サービスが：

ウェブ・サーバであることを特徴とする方法。

【請求項 23】

請求項 14 記載の方法であって、

前記クラスタにおける 1 つ又は複数のノードでの前記識別サービスを提供する前記ホストのアドレスをスプーフィングし、前記ホストのために、前記ホストがブートしている間に接続シーケンスを開始する工程；

を更に備えることを特徴とする方法。

【請求項 24】

請求項 23 記載の方法であって：

前記アドレスが、インターネット・プロトコル・オーバ媒体アクセス制御（I P / M A C）アドレスであり；かつ

前記接続シーケンスが、伝送制御プロトコル・オーバ I P（T C P / I P）接続シーケンス又は、ユーザ・データグラム・プロトコル（U D P）オーバ I P（U D P / I P）接続シーケンスであることを特徴とする方法。

【請求項 25】

請求項 14 記載の方法であって、前記ノード・クラスタが：

3 次元のトーラスを備えるトポロジによって互いに結合される複数のノード；

を備えることを特徴とする方法。

【請求項 26】

請求項 14 記載の方法であって、前記ノード・クラスタにおけるノードが：

2 つのスイッチに結合された中央処理装置（C P U）であることを特徴とする方法。

10

20

30

40

50

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、一般的に、データ処理に関し、特に、HPCシステムにおけるオン・デマンド・インスタンスーションに関する。

【背景技術】

【0002】

高性能計算（HPC）は多くの場合、複雑な物理的現象又はアルゴリズム現象をモデリングし、シミュレーションし、解析する科学者及び技術者によって用いられる計算システムによって特徴付けられる。現在、HPCマシンは通常、ノードと呼ばれる数多くの、1つ又は複数のプロセッサから成るHPCクラスタを用いて設計されている。大部分の大規模の科学アプリケーション及び工学アプリケーションについては、性能は、個々のノードの速度ではなく、個々のノードの並列スケーラビリティによって主に判定される。したがって、スケーラビリティは多くの場合、そのような高性能クラスタを構築するか購入するうえでの制約要因である。スケーラビリティは通常、i) ハードウェア、ii) メモリ帯域幅、I/O帯域幅、並びに通信帯域幅、iii) ソフトウェア、iv) アーキテクチャ、及びv) アプリケーションに基づいたものとみなされる。

10

【発明の開示】

【発明が解決しようとする課題】

【0003】

大部分の通常のHPCの環境における処理帯域幅、メモリ帯域幅、及びI/O帯域幅は通常、うまくバランスがとれておらず、したがって、うまくスケーリングされていない。多くのHPC環境は、ハイエンドのデータ処理要件を満たすI/O帯域幅を有していないか、インストールされている不必要な構成部分が多すぎるブレードによって構築されており、このことはシステムの信頼度を劇的に低減する傾向にある。よって、多くのHPC環境は、生産指向環境における効率的な処理を行ううえでの頑健なクラスタ管理ソフトウェアを備えない場合がある。

20

【課題を解決するための手段】

【0004】

本発明は、HPCシステムに関連した欠点と問題との何れか又は両方を削減するか除去する場合がある。

30

【0005】

一実施例では、高性能計算（HPC）システムにおけるオン・デマンド・インスタンスーションの方法は、ノード・クラスタを含むHPCサーバに対して外部に共用宣言される第1ポート番号と第1ホスト名とを規定するクライアントからの接続要求を受信する工程、第1ポート番号と第1ホスト名とに相当するHPCサーバでのサービスを識別する工程、識別サービスが利用可能であるか否かを判定する工程、及びクラスタにおける1つ又は複数のノードでの識別サービスを備えるホストを、識別サービスが利用可能な場合、インスタンス化する工程を備える。

【0006】

40

本発明の実施例は1つ又は複数の技術上の効果を備え得る。例として、特定の実施例は、HPCサーバでのサービスをクライアントが要求することを可能にし得る。特定の実施例では、サービスにアクセスするうえでの1つ又は複数の基準を、サービスをアクセスする旨のクライアントからの要求が満たす場合にのみ、HPCサーバでのサービスがクライアントに利用可能である。特定の実施例は、仮想クラスタにおけるホストの高利用可能性を備える。特定の実施例は、インターネット・サービス要求を動的に監視し、要求サービスを備えるホストに対してそのような要求をマッピングし、そのホストをインスタンス化する。本発明の特定の実施例は、上記の技術上の効果の全部又は一部を備えるか、上記の技術上の効果を何ら備えるものでない。特定の実施例は、1つ又は複数の別の技術上の効果を備える場合があり、その1つ又は複数の、添付図面、明細書、及び特許請求の範囲か

50

ら当業者に容易に明らかとなり得る。

【発明を実施するための最良の形態】

【0007】

本発明の特定の実施例は、1つ又は複数の技術上の効果を備え得る。例として、特定の実施例は、HPCシステムで実行するジョブをスケジューリングすることに通常関連する時間要件を軽減し得る。特定の実施例は、HPCシステムで実行するジョブをスケジューリングすることに通常関連する計算量上の要件を軽減し得る。本発明の特定の実施例は、上記の技術上の効果の全部又は一部を備えるか、上記の技術上の効果を何ら備えない。特定の実施例は、その1つ又は複数の添付図面、明細書、及び特許請求の範囲から当業者に容易に明らかであり得る、その1つ又は複数の別の、技術上の効果を備える場合がある。

10

【0008】

本願開示及びその特徴並びに効果がより徹底的に分かるよう、次に、下記明細書を添付図面とともに参照する。

【実施例】

【0009】

図1は、ソフトウェアのアプリケーション及び処理、例えば、大気シミュレーション、気象シミュレーションや衝撃シミュレーションを、HPC手法を用いて実行する高性能計算(HPC)システム100を示す構成図である。システム100は、処理性能にかなり同様なI/O性能を備えている、種々の計算ノード115の間で動的に割り当てられるHPC機能をユーザに備える。一般的に、これらのノード115は、とりわけ、この、入出力(I/O)性能の増加とファブリック・レーテンシの低減が理由で容易にスケール可能である。例えば、分散アーキテクチャにおけるノード115のスケラビリティは：

20

$$S(N) = 1 / ((FP/N) + FS) * (1 - FC * (1 - RR/L));$$

である、アムダールの法則の派生形によって表す場合があり、 $S(N) = N$ プロセッサでの高速化であり、 FP = 並列コードの割合であり、 FS = 非並列コードの割合であり、 FC = 通信に充てられる処理の割合であり、 RR/L = 遠隔メモリ帯域幅の局所メモリ帯域幅に対する比率である。したがって、HPCシステム100が、処理性能にかなり等しいか、かなり近づいているI/O性能を備えることによって、HPCシステム100はHPCアプリケーションの全体効率を向上させ、システム管理の容易化を可能にする。

【0010】

30

HPCシステム100は、(科学者や技術者などの)ユーザがジョブ150を投入してHPCサーバ102上で処理することを可能にする分散クライアント/サーバ・システムである。例えば、システム100は、1つ又は複数の管理ワークステーション又は局所クライアント120にネットワーク106を介して接続されるHPCサーバ102を含み得る。しかし、スタンドアロン型計算環境又は何れかの別の適切な環境であり得る。要約すれば、システム100は、スケラビリティの高いノード115を含み、かつ、ユーザが、ジョブ150を投入し、スケラブルなノード150をジョブ150に動的に割り当て、割り当てられたノード115を用いてジョブ150を自動的に実行することを可能にする何れかのHPC計算環境である。ジョブ150は、HPC手法を用いて処理され、何れかの適切なユーザによって投入されるよう処理可能な何れかのバッチ・ジョブ又はオンライン・ジョブであり得る。例えば、ジョブ150は、シミュレーション、モデル、又は何れかの別の高性能要件に対する要求であり得る。ジョブ150は、クラスタ化データベース、オンライン・トランザクション処理システム、又はクラスタ化アプリケーション・サーバなどのデータ・センター・アプリケーションを実行する要求である場合もある。本明細書及び特許請求の範囲の原文記載の「*dynamically*」の語は一般的に、特定の処理が少なくとも部分的には実行時に1つ又は複数の変数に基づいて判定されるということを表す。本明細書及び特許請求の範囲の原文記載の「*automatically*」の語は一般的に、適切な処理がHPCシステム100の少なくとも部分によって実質的に行われるということを表す。この「*automatically*」の語が更に、何れかの適切なユーザ又は管理者による、システム100との相互作用を、本願の開示の範囲から

40

50

逸脱することなく想定しているということが分かるものである。

【0011】

HPCサーバ102は、複数の、バランスされたノード115及びクラスタ管理ノード130を用いてジョブ150を処理するよう動作可能な何れかの、局所コンピュータ又は遠隔コンピュータを備える。一般的に、HPCサーバ102は、ブレード・サーバ又は別の分散サーバなどの分散コンピュータを備える。サーバ102は、その構成がどんなものであっても、複数のノード115を含む。ノード115は、例えば、ブレード、汎用パーソナル・コンピュータ（PC）、マッキントッシュ、ワークステーション、ユニックス（Unix）ベースのコンピュータ、又は何れかの別の適切な装置などの何れかのコンピュータ装置又は処理装置を備える。一般的に、図1は、本願の開示とともに用い得るコンピュータの一例に過ぎないものを備える。例えば、図1は本願開示とともに用い得る一サーバ102を示すが、システム100は、サーバ以外のコンピュータや、サーバ・プールを用いて実施することが可能である。すなわち、本願の開示は、汎用コンピュータ以外のコンピュータや、通常のオペレーティング・システムなしのコンピュータを想定している。本明細書及び特許請求の範囲の原文において用いているように、「computer」の語は、パーソナル・コンピュータ、ワークステーション、ネットワーク・コンピュータ、又は何れかの別の適切な処理装置に及ぶことが意図されている。HPCサーバ102又はコンポーネント・ノード115は、リナックス（Linux）、ユニックス（Unix）、ウィンドウズ（登録商標）・サーバ、又は何れかの別の適切なオペレーティング・システムを含む何れかのオペレーティング・システムを実行するよう形成し得る。一実施例によれば、HPCサーバ102は、遠隔ウェブ・サーバを含んでいてもよく、遠隔ウェブ・サーバに通信可能に結合されていてもよい。したがって、サーバ102は、ノード115を動的に割り当ててHPCジョブ150を処理するのに適した何れかの組み合わせでソフトウェア及び／ハードウェアを備えている何れかのコンピュータを備え得る。

【0012】

概略を述べれば、HPCサーバ102は、管理ノード105、複数のノード115を備えるグリッド110、及びクラスタ管理エンジン130を含む。特に、サーバ102は、i) デュアル・プロセッサと、ii) 大容量の、高帯域幅のメモリと、iii) デュアル・ホスト・チャネル・アダプタ（HCA）と、iv) 統合ファブリック・スイッチングと、v) FPGAサポートと、vi) 冗長な電源入力すなわちN+1の電源との構成部分の一部又は全部を備えている複数のブレード（ノード115）を含む標準の19インチのラックであり得る。これらの種々の構成部分によって、障害がノード・レベルに限定されることが可能になる。しかし、HPCサーバ102及びノード115がこれらの構成部分を全部含まない場合があるということが分かるものである。

【0013】

管理ノード105は実質的に専ら、管理者を管理するか支援する少なくとも1つのブレードを備える。例えば、管理ノード105は2つのブレードを備える場合があり、2つのブレードのうちの1つは（アクティブ構成／パッシブ構成などの）冗長性を有する。一実施例では、管理ノード105は、HPCノード115と同じ種類のブレード又は計算装置であり得る。しかし、管理ノード105は、少なくとも部分的にグリッド110を管理するよう動作可能な状態にとどまっている限り、何れかの数の回路を含み、何れかの適切な方法で構成される何れかのノードであり得る。多くの場合、管理ノード105は、グリッド110に併せて表している複数のHPCノード115から物理的又は論理的に分離されている。図示する実施例では、管理ノード105は、グリッド110にリンク108を介して通信可能に結合し得る。リンク108は、何れかの適切な通信プロトコルを実施する何れかの通信管路を備え得る。一実施例では、リンク108は管理ノード105とグリッド110との間の、ギガビット又は10ギガビットのイーサネット（登録商標）通信を備える。

【0014】

グリッド110は、処理能力を向上させるよう相互接続されるノード群115である。

グリッドは、通常、3次元トラスであるが、本願の開示の範囲から逸脱することなく、メッシュ、超立方体、又は何れかの別の形状若しくは構成であり得る。グリッド110におけるノード115間のリンクは、例えばファイバ又は銅などの、電気信号又は電磁気信号を伝達することが可能な、直列又は並列の、アナログ・リンク、デジタル・リンク、又は何れかの別の種類のリンクを備え得る。各ノード115は統合スイッチによって構成される。このことによって、ノード115が3次元トラスの基本的な構成体により容易になることを可能にし、別のノード115間のXYZ距離を最小にすることに寄与する。更に、このことは、最大でギガビット・レベルの速度での大容量システムで銅線を機能させる場合があり、一部の実施例では、最長のケーブルは5メートル未満である。要約すれば、ノード115は、一般的に、最短距離の通信とI/O帯域幅の増加とに最適化される。

10

【0015】

各ノード115は、クラスタ管理エンジン130と通信可能に結合されるクラスタ・エージェント132を含み得る。一般的には、エージェント132は要求又はコマンドを管理ノード105及び／又はクラスタ管理エンジン130から受信する。エージェント132は、ノード115の物理ステータスを判定し、処理データを「ハートビート」などによって管理ノード105に通信するよう動作可能な何れかのハードウェア、ソフトウェア、ファームウェア、又はそれらの組み合わせを含み得る。別の実施例では、管理ノード105は、エージェント132を周期的にポーリングして関連ノード115のステータスを判定し得る。エージェント132は、クラスタ管理エンジン130の少なくとも部分と互換である状態のままである限り、例えば、C、C++、アセンブラ、ジャバ（Java）、ビジュアル・ベーシック（Visual Basic）及び他の言語又はそれらの組み合わせなどの何れかの適切なコンピュータ言語で書かれる場合がある、すなわち記述される場合がある。

20

【0016】

クラスタ管理エンジン130は、ノード115を動的に、割り当て、管理し、ノード115を用いてジョブ150を実行するよう動作可能な、何れかのハードウェア、ソフトウェア、ファームウェア、又はそれらの組み合わせを含み得る。例えば、クラスタ管理エンジン130は、C、C++、ジャバ、ビジュアル・ベーシック、アセンブラ、4GLの何れかの適切なバージョン、及びその他の言語又はそれらの何れかの組み合わせにおいて作成すなわち記述される場合がある。クラスタ管理エンジン130を図1に単一のマルチタスク・モジュールとして示しているが、このエンジンによって行われる特徴と機能は、（図7に更に詳細に表すように）例えば、物理層モジュール、仮想層モジュール、ジョブ・スケジューラ、及びプレゼンテーション・エンジンなどの、複数のモジュールによって行い得る。更に、管理ノード105の外部に示すが、管理ノード105は通常、クラスタ管理エンジン130と関連する1つ又は複数の処理を実行し、クラスタ管理エンジン130を記憶させ得る。更に、クラスタ管理エンジン130は、本願の開示の範囲から逸脱することなく、別のソフトウェア・モジュールの子モジュールすなわちサブモジュールであり得る。したがって、クラスタ管理エンジン130は、ノード115とジョブ150とをインテリジェントに管理するよう動作可能な1つ又は複数のソフトウェア・モジュールを備える。特定の実施例では、クラスタ管理エンジンは、以下に説明するように、ジョブ150にノード115を割り当てるスケジューラ515を含む。スケジューラ515は、以下に更に説明するように、ジョブ150にノード115を割り当てるスケジューリング・アルゴリズムを用い得る。

30

40

【0017】

サーバ102は、クライアント・サーバ環境又は別の分散環境においてネットワーク106を介してクライアント120などの別のコンピュータ・システムと通信するインタフェース104を含み得る。特定の実施例では、サーバ102はジョブ150又はジョブ・ポリシーをネットワーク106から受信してディスク・ファーム140に記憶させる。ディスク・ファーム140は、ノードを相互接続するものと同じ広帯域インタフェースを用

50

いて計算アレイに直接接続される場合もある。一般的には、インタフェース104は、適切な組み合わせにおける、ネットワーク106と通信するよう動作可能な、ソフトウェア及び／又はハードウェアにコード化される論理を備える。特に、インタフェース104は、物理的信号を通信するよう動作可能な通信ネットワーク106又はハードウェアに関連した1つ又は複数の通信プロトコルをサポートするソフトウェアを備え得る。

【0018】

ネットワーク106は、コンピュータ・サーバ102と、クライアント120などの何れかの別のコンピュータとの間の無線通信又は有線通信を容易にする。実際に、サーバ102とクライアント120との間に存在するものとして図示しているが、ネットワーク106は、本願の開示の範囲から逸脱することなく、種々のノード115間に存在する場合もある。すなわち、ネットワーク106は、種々の計算構成部分間での通信を容易にするよう動作可能な何れかのネットワーク又はサブネットワークに及ぶ。ネットワーク106は、例えば、インターネット・プロトコル（IP）パケット、フレーム・リレイ・フレーム、非同期転送モード（ATM）・セル、音声、データ、及び別の適切な情報をネットワーク・アドレス間で通信し得る。ネットワーク106は、1つ又は複数の場所での、1つ又は複数の、ローカル・エリア・ネットワーク（LAN）、無線アクセス・ネットワーク（RAN）、メトロポリタン・エリア・ネットワーク（MAN）、ワイド・エリア・ネットワーク（WAN）、インターネットとして知られるグローバル・コンピュータ・ネットワークの全部若しくは一部、及び／又はいずれかの別の通信システムを含み得る。

【0019】

一般的には、ディスク・ファーム140は、ジョブ150、プロファイル、ブート画像、又は別のHPC情報を記憶させる何れかのメモリ、データベース又はストレージ・エリア・ネットワーク（SAN）である。図示する実施例によれば、ディスク・ファーム140は、1つ又は複数のストレージ・クライアント142を含む。ディスク・ファーム140は、いくつかの通信プロトコル、例えば、インフィニバンド（InfiniBand）（IB）、ギガビット（Gigabit）・イーサネット（Ethernet）（登録商標）（GE）、又はファイバチャネル（FibreChannel）（FC）のうちの何れかによってデータ・パケットを処理し、ルーティングする場合がある。データ・パケットは通常、ディスク・ファーム140内のデータを伝送するのに用いられる。データ・パケットは、発信元識別子と宛て先識別子とを有するヘッダを含み得る。発信元識別子、例えば、発信元アドレスは情報の発信元を識別し、宛て先識別子、例えば宛て先アドレスは情報の受信先を識別する。

【0020】

クライアント120は、ジョブ投入画面又はアドミニストレーション（管理）画面がユーザにグラフィカル・ユーザ・インタフェース（GUI）126を介して提示されるよう動作可能な何れかの装置である。概略を述べれば、図示するクライアント120は、少なくともGUI126を含み、システム100に関連した何れかの適切なデータを受信し、送信し、処理し、記憶するよう動作可能な電子計算機装置を備える。通信可能にサーバ102に結合される何れかの数のクライアント120が存在する場合があるということが分かるものである。更に、「client 120」及び「user of client 120」は、本願の開示の範囲から逸脱することなく、適宜、同義で用い得る。更に、図示を容易にするよう、各クライアントは1つのユーザによって用いられるという点で表している。しかし、多くのユーザが1つのコンピュータを用いて、ジョブ150を同じGUI126を用いて通信し得る。

【0021】

本願の開示に記載のように、クライアント120は、パーソナル・コンピュータ、タッチ画面端末、ワークステーション、ネットワーク・コンピュータ、キオスク、無線データ・ポート、携帯電話、携帯情報端末（PDA）、これら若しくは別の装置の内部の1つか複数のプロセッサ、又は何れかの別の適切な処理装置に及ぶことが意図されている。例えば、クライアント120は、情報を受け入れることが可能な、キーパッド、タッチ画面、

マウス、又は別の装置などの入力装置、ディジタル・データ、視覚情報を含む、サーバ102若しくはクライアント120の処理に関連した情報を伝える出力装置、又はGUI126を含むコンピュータを備え得る。入力装置も出力装置も、磁気コンピュータ・ディスクなどの固定記憶媒体又は取り外し可能記憶媒体、CD-ROM、又は別の適切な媒体を含んで、アドミニストレーション画面及びジョブ投入画面、すなわちGUI126を介してクライアント120のユーザからの入力を受信し、出力をそのユーザに備え得る。

【0022】

GUI126は、i) クライアント120のユーザがシステム100とインタフェースをとって1つ又は複数のジョブ150を投入する工程；及び／又はii) システム管理者（又はネットワーク管理者）がクライアント120を用いて、何れかの適切な監視目的でシステム100とインタフェースをとる工程；を可能にするよう動作可能なグラフィカル・ユーザ・インタフェースを備える。一般的に、GUI126は、HPCシステム100によって備えられるデータの効率的でかつユーザ・フレンドリな提示をクライアント120のユーザに備える。GUI126は、ユーザによって操作される双方向フィールド、プルダウン・リスト、及びボタンを有する複数のカスタム化可能なフレームすなわち表示を備え得る。一実施例では、GUI126は、種々のジョブ・パラメータ・フィールドを表示し、クライアント120のユーザからのコマンドを入力装置のうちの1つを介して受信するジョブ投入ディスプレイ（表示）を表示させる。代替的に、又は組み合わせによって、GUI126は、ノード115の物理ステータス及び論理ステータスを図6A乃至6Bに示すようにシステム管理者に提示し、種々のコマンドを管理者から受信する場合がある。管理者コマンドは、ノードを利用可能（不能）なものとして印しを付けるコマンド、保守するようノードをシャットダウンさせるコマンド、ノードを再ブートするコマンド、又は何れかの別の適切なコマンドを含み得る。更に、グラフィカル・ユーザ・インタフェース（*graphical user interface*）の語を単数形か複数形で用いて、1つ又は複数のグラフィカル・ユーザ・インタフェースと、特定のグラフィカル・ユーザ・インタフェースの表示の各々を表し得るということが分かるものである。したがって、GUI126は、情報をシステム100において処理し、結果をユーザに効率的に提示する、一般的なウェブ・ブラウザなどの何れかのグラフィカル・ユーザ・インタフェースを想定している。サーバ102は、クライアント120からのデータをウェブ・ブラウザ（例えば、マイクロソフト（*Microsoft*）社のインターネット・エクスプローラ（*Internet Explorer*）又はネットスケープ（*Netscape*）社のナビゲータ（*Navigator*））を介して受け入れ、適切なHTML応答又はXML応答をネットワーク106を用いて戻すことが可能である。

【0023】

処理の一特徴では、HPCサーバ102はまず、初期化すなわちブートされる。この処理中に、クラスタ管理エンジン130は、グリッド110におけるノード115の存在、状態、位置、及び／又は別の特性を判定する。上記のように、このことは、各ノードが初期化されるか、管理ノード105によってほぼ即時にポーリングされると、通信される「ハートビート」に基づく場合がある。次に、クラスタ管理エンジン130は、グリッド110の種々の部分を1つ又は複数の仮想クラスタ220に、例えば、所定のポリシーに基づいて動的に割り当て得る。一実施例では、クラスタ管理エンジン130は、考えられる障害についてノード115を継続して監視し、ノード115のうちの1つに障害が起こったということを判定すると、種々の回復手法のうちの何れかを用いて障害を効果的に管理する。クラスタ管理エンジン130が一意の実行環境を仮想クラスタ220の割り当てノード毎に管理し、備える場合もある。実行環境は、ホスト名、IPアドレス、オペレーティング・システム、構成サービス、局所ファイル・システム並びに共有ファイル・システム、及びインストールされているアプリケーション群並びにデータ群を含み得る。クラスタ管理エンジン130は、関連ポリシーによって、かつ優先度などのクラスタ間ポリシーによってノードを仮想クラスタ220から動的に加算するか減算する場合がある。

【0024】

ユーザは、クライアント120にログオンする場合、GUI126を介してジョブ投入画面が提示される場合がある。ユーザがジョブ・パラメータを入力し、ジョブ150を投入すると、クラスタ管理エンジン130はジョブ投入、関連パラメータ、及び、ジョブ150、ユーザ又はユーザ群と関連した何れかの所定のポリシーを処理する。クラスタ管理エンジン130は更に、適切な仮想クラスタ220を、少なくとも部分的にはこの情報に基づいて判定する。エンジン130は更に、ジョブ空間230を仮想クラスタ220内に割り当て、ジョブ150を割り当てノード115にわたってHPC手法を用いて実行する。少なくとも部分的にはこの向上されたI/O性能に基づいて、HPCサーバ102はジョブ150の処理をよりすばやく完了し得る。完了すると、クラスタ管理エンジンは結果160をユーザに通信する。

10

【0025】

図2はノード（又はブレード）115の例を示す。ノード115は、1つ又は複数のジョブ150の、スレッド又は処理などの全部又は一部を処理するよう、何れかの向きでの何れかの計算装置を含む。制約ではなく例として、ノード115は、ゼオン（XEON）マザーボード、オプテロン（OPTERON）マザーボード、又は別の計算装置を含み得る。ノード115は、グリッド110におけるノード115にわたってスイッチング機能を分散させることを可能にする統合ファブリックを備えるアーキテクチャを有する。特定の実施例では、グリッド110におけるノード115にわたってそのような機能を分散させることはグリッド110における集中スイッチングを妨げる場合があり、そのことは同様に、グリッド110におけるフォールト・トレランスを向上させ、グリッド110にお

20

【0026】

ノード115は、2つのCPU164と、スイッチ（又はファブリック）166を含む。ノード115への言及は、適宜、2つのCPU164と、スイッチ166とに及び得る。ノード115への言及は、適宜、CPU164のみに及び得る。スイッチ166は、統合スイッチであり得る。特定の実施例では、スイッチ166は24個のポートを有する。スイッチ166上の2つのポートは、ノード115との間の入出力を行うよう、ノード115を管理ノード105に結合し得る。更に、グリッド110における3次元メッシュ、3次元トラス、又は別のトポロジの実施を容易にするよう、スイッチ166上の2つのポートは各々、グリッド110のx軸に沿ってノード115を別のノード115に結合する
30
場合があり、スイッチ166上の2つのポートは各々、グリッド110のy軸に沿ってノード115を別のノード115に結合する場合があり、スイッチ166上の2つのポートは各々、グリッド110のz軸に沿ってノード115を別のノード115に結合する場合がある。更に、スイッチ166上の別のポートはノード115を別のノード115にグリッド110において結合して、グリッド110における（4次元トラス又は4つ以上の次元を含む別の非伝統的なトポロジなどの）多次元トポロジの実施を容易にする場合がある。特定の実施例では、スイッチ166上の1つ又は複数のポートは、ノード115を1つ又は複数の別のノード115にグリッド110の1つ又は複数の対角線軸に沿って結合する
40
場合があり、そのことは、ノード115と、ノード115から比較的遠いところにある1つ又は複数の別のノード115との間の通信ジャンプ又は通信ホップを削減し得る。制約ではなく例として、スイッチ166上のポートは、ノード115からいくつかの3次元ジャンプ離れたところにある、グリッド110の北東軸に沿って存在する別のノード155に結合し得る。特定の実施例では、スイッチ166はインフィニバンド（InfiniBand）・スイッチである。特定のスイッチ166を図示し、説明したが、本発明は何れかの適切なスイッチ166を想定している。

【0027】

リンク168aはCPU164aをスイッチ166に結合する。リンク168bはCPU164aを以下に説明するように、別のノード115における別のスイッチ166に結合する。リンク168cはCPU164bをスイッチ166に結合する。リンク168dはCPU164bを、以下に説明するように別のスイッチ166に結合する。リンク16

50

8 eと168 fは、スイッチ166を、以下に更に説明するように、別のノード115における2つの別のCPU164に結合する。特定の実施例では、リンク168は、各方向に毎秒約1ギガバイトを通信することができるインフィニバンド4Xリンクを含む。特定のリンク168を示し、説明したが、本発明は何れかの適切なリンク168を想定している。リンク170は、ノード115へのI/Oリンクである。リンク170は、各方向に毎秒約1ギガバイトを通信することができるインフィニバンド4Xリンクを含む。特定のリンク170を示し、説明したが、本発明は何れかの適切なリンク170を想定している。リンク172は、スイッチ166を、以下に説明するように、別のノード115における別のスイッチ166にスイッチ166を結合する。特定の実施例では、リンク172は、各方向に毎秒約3ギガバイトを通信することができるインフィニバンド12Xリンクを含む。特定のリンク172を示し、説明したが、本発明は何れかの適切なリンク172を想定している。

10

【0028】

図3は、ノード115におけるCPU164の例を示す。CPU164の例を示し、説明したが、本発明は何れかの適切なCPU164を想定している。CPU164は、プロセッサ174、メモリ・コントローラ・ハブ(MCH)176、メモリ装置178、及びホスト・チャネル・アダプタ(HCA)180を含む。プロセッサ174は、ハードウェア構成部分、ソフトウェア構成部分、組み込み論理構成部分又は2つ以上のそのような構成部分の組み合わせを含む。特定の実施例では、プロセッサ174は、インテル(INTERTEL)社によるNOCONA(ノコナ)XEON(ゼオン)プロセッサ174である。特定の実施例では、プロセッサ174は、約1メガバイトのキャッシュを有し、毎秒約7.2ギガフロップが可能な約3.6ギガヘルツのプロセッサである。特定の実施例では、プロセッサ174は、ハイパスレディングを備える。特定の実施例では、プロセッサ174は、メモリ帯域幅の効率的な利用を備えるメモリ・コントローラを含む。特定のプロセッサ174を示し、表したが、本発明は何れかの適切なプロセッサ174を想定している。

20

【0029】

バス182はプロセッサ174とMCH176とをお互いに結合する。特定の実施例では、バス182は、毎秒約6.4ギガバイトを通信することができる約800MHzのフロント・サイド・バス(FSB)である。特定のバス182を示し、説明したが、本発明は何れかの適切なバス182を想定している。MCH176は、プロセッサ174と、メモリ装置178などの、HPCシステム100の1つ又は複数の別の構成部分との間の通信を容易にする、ハードウェア構成部分、ソフトウェア構成部分、若しくは組み込み論理構成部分、又は2つ以上のそのような構成部分の組み合わせを含む。特定の実施例では、MCH176は、プロセッサ174と、メモリ装置178、バス182、レベル2(L2)キャッシュ及び、CPU164の1つ又は複数の別の構成部分のうちの1つ又は複数のものとの間の通信を制御するCPU164用のノースブリッジである。特定の実施例では、MCH176は、リンデンハースト(LINDENHURST)E7520MCH176である。特定の実施例では、メモリ装置178はMCH176に別個に結合される2つのダブル・データ・レート(DDR)のメモリ装置を含む。制約ではなく例として、メモリ装置178は、その各々が、チャネル毎に毎秒約3.2ギガバイトが可能なその2つのDDR2-400メモリを含み得る。特定のメモリ装置178を示し、説明したが、本発明は何れかの適切なメモリ装置178を想定している。

30

40

【0030】

特定の実施例では、リンクは、プロセッサ174と、ICHに結合される基本I/Oシステム(BIOS)と、ICHに結合される、ギガビット(Gigabit)・イーサネット(Ethernet)(登録商標)(GbE)コントローラ又は別のイーサネット(Ethernet)(登録商標)・インタフェースとの何れか又は両方などの、HPCシステム100の1つ又は複数の別の構成部分との間のI/Oを容易にする1つ又は複数のハードウェア構成部分、ソフトウェア構成部分、又は組み込み論理構成部分を含むI/O

50

コントローラ・ハブ（ICH）にMCH176を結合する。特定の実施例では、ICHは、CPU164のI/O機能を制御するCPU164用のサウスブリッジである。ICHに結合されるイーサネット（登録商標）・インタフェースは、ICHと、イーサネット（登録商標）・インタフェースに結合されるベースボード管理コントローラ（BMC）との間の通信を容易にし得る。特定の実施例では、HPCシステム100の管理ノード105又は別の構成部分は、1つ又は複数のそのようなBMCを含む。特定の実施例では、リンクはイーサネット（登録商標）・インタフェースをスイッチに結合し、それによって1つ又は複数のGbE管理ポートへのアクセスを備える。

【0031】

バス184は、MCH176とHCA180とをお互いに結合する。特定の実施例では、バス184は、毎秒約4ギガバイトを通信することが可能なPCI-Express（エクスプレス）8Xバス184などの、端末構成部分相互接続（PCI）バス184である。特定の実施例では、バス184を図示し、説明したが、本発明はどれかの適切なバス184を想定している。HCA180は、CPU164にチャンネル・ベースのI/Oを備える、ハードウェア構成部分、ソフトウェア構成部分、若しくは組み込み論理構成部分、又は2つ以上のそのような構成部分の組み合わせを含む。特定の実施例では、HCA180はメラノックス（MELLANOX）社のインフィニバンドHCA180である。特定の実施例では、HCA180は、ノード115におけるスイッチ166に対して処理構成要素（PE）毎に約1.85ギガバイトを可能にし、基本I/Oシステム（BIOS）、イーサネット（登録商標）・インタフェース又は別のI/OなどのI/OにPE毎に対して約800メガバイトを可能にする場合がある、毎秒約2.65ギガバイトの帯域幅を備える。特定の実施例では、HCA180は、スイッチ166での帯域幅が毎秒約3.7ギガバイトに達して、ピーク時で毎秒約13.6ギガフロップとなることと、スイッチ166でのI/O速度がギガフロップ毎約50メガバイトに達してフロップ毎約0.27バイトとなることとの何れか又は両方を可能にする。特定の実施例では、HCA180を示し、説明したが、本発明はどれかの適切なHCA180を想定している。各リンク168は、HCA180をスイッチ166に結合する。リンク168aは、以下に説明するように、HCA180に対して1次のスイッチ166である第1スイッチ166にHCA180を結合する。特定の実施例では、HCA180を含むノード115は、第1スイッチ166を含む。リンク168bは、以下に説明するように、HCA180に対して2次のスイッチである第2スイッチ166にHCA180を結合する。特定の実施例では、HCA180を含まないノード115は、以下に説明するように、第2スイッチ166を含む。

【0032】

図4は、2つのスイッチ166と4つのプロセッサ174とを含むノード対186の例を示す。ノード対186におけるスイッチ166は、お互いに対して冗長であり、そのことはノード対186でのフォールト・トレランスを向上させ得る。ノード対186における第1スイッチ166が適切に機能していない場合、ノード対186における第2スイッチ166がノード対186における4つのCPU全てにスイッチングを備え得る。ノード対186では、スイッチ166aは、CPU164aと164bとに対しては1次スイッチ166であり、CPU164cと164dとに対しては2次スイッチ166である。スイッチ166bは、CPU164cと164dとに対して1次スイッチ166であり、CPU164aと164bとに対して2次スイッチ166である。両方のスイッチ166aと166bが適切に機能している場合、スイッチ166aはCPU164aと164bとにスイッチングを備える場合があり、スイッチ166bはCPU164cと164dとにスイッチングを備え得る。スイッチ166aが適切に機能しているが、スイッチ166bが適切に機能していない場合、スイッチ166aは、CPU164a、164b、164c及び164dにスイッチングを備え得る。スイッチ166bが適切に機能しているが、スイッチ166aが適切に機能していない場合、スイッチ166bがCPU164a、164b、164c、及び164dにスイッチングを備え得る。

【0033】

リンク172は、グリッド110におけるノード対186の外部の6つのノード115にノード対186における各ノード115を結合する。制約ではなく例として、スイッチ166aでのリンク172aは、グリッド110におけるノード115aのノース（北）方向にあるノード対186の外部の第1ノード115にノード115aを結合し、スイッチ166aでのリンク172bは、グリッド110におけるノード115aのサウス（南）方向にあるノード対186の外部の第2ノード115にノード115aを結合し、スイッチ166aでのリンク172cは、グリッド110におけるノード115aのイースト（東）方向にあるノード対186の外部の第3ノード115にノード115aを結合し、スイッチ166aでのリンク172dは、グリッド110におけるノード115aのウエスト（西）方向にあるノード対186の外部の第4ノード115にノード115aを結合し、スイッチ166aでのリンク172eは、グリッド110におけるノード115aの上にあるノード対186の外部の第5ノード115にノード115aを結合し、スイッチ166aでのリンク172fは、グリッド110におけるノード115aの下にあるノード対186の外部の第6ノード115にノード115aを結合する。特定の実施例では、リンク172は、お互いに異なるノード対186の外部のノード115群にノード対186におけるノード115a及び115bを結合する。制約ではなく例として、スイッチ166aでのリンク172は、ノード対186の外部の第1ノード115、ノード対186の外部の第2ノード115、ノード対186の外部の第3ノード115、ノード対186の外部の第4ノード115、ノード対186の外部の第5ノード115、及びノード対186の外部の第6ノード115を含む、ノード対186の外部の、第1の6つのノード115の群をノード115aに結合し得る。スイッチ166bでのリンク172は、ノード対186の外部の第7ノード115、ノード対186の外部の第8ノード115、ノード対186の外部の第9ノード115、ノード対186の外部の第10ノード115、ノード対186の外部の第11ノード115、及びノード対186の外部の第12ノード115を含む、ノード対186の外部の、第2の6つのノード115の群をノード115bに結合し得る。

【0034】

特定の実施例では、リンク172は、第1端の反対側の、グリッド110の第2端に隣接する第2ノード115にグリッド110の第1端に隣接する第1ノード115を結合し得る。制約ではなく例として、グリッド110の左端に隣接する第1ノード115と、グリッド110の左端の反対側にある、グリッド110の右端に隣接する第2ノード115を検討する。リンク172は、グリッド110における、第2ノード115の位置に対する第1ノード115の位置にもかかわらず、第1ノード115が第2ノード115のイースト方向にあり、第2ノード115が第1ノード115のウエスト方向にあるように、第1ノード115と第2ノード115とをお互いに結合し得る。別の例として、グリッド110の前端に隣接する第1ノード115と、グリッド110の前端の反対側にある、グリッド110の後端に隣接する第2ノード115を検討する。リンク172は、グリッド110における、第2ノード115の位置に対する第1ノード115の位置にもかかわらず、第1ノード115が第2ノード115のサウス方向にあり、第2ノード115が第1ノード115のノース方向にあるように、第1ノード115と第2ノード115とをお互いに結合し得る。更に別の例として、グリッド110の上端に隣接する第1ノード115と、グリッド110の上端の反対側にある、グリッド110の下端に隣接する第2ノード115を検討する。リンク172は、グリッド110における、第2ノード115の位置に対する第1ノード115の位置にもかかわらず、第1ノード115が第2ノード115の下にあり、第2ノード115が第1ノード115の上にあるように、第1ノード115と第2ノード115とをお互いに結合し得る。

【0035】

図5A乃至図5Dは、システム100におけるグリッド110とその利用又はトポロジの種々の実施例を示す。図5Aは、複数のノード種類を用いたグリッド110の一構成、すなわち3次元トラスを示す。例えば、図示するノード種類は外部I/Oノード、ファ

10

20

30

40

50

イル・システム（F S）サーバ、F Sメタデータ・サーバ、データベース・サーバ、及び計算ノードである。図5 Bは、グリッド1 1 0の「フォールディング（畳むこと）」の例を示す。フォールディングは一般的に、グリッド1 1 0の1つの物理的な先端が、相当する軸方向の先端と接続し、それによってより頑健なトポロジすなわちエッジレス・トポロジを備えることを可能にする。この実施例では、ノード1 1 5は、ラップアラウンドされて、ノード・ライン2 1 6によるほぼシームレスなトポロジ接続を備える。ノード・ライン2 1 6は、2つ以上のノード1 1 5を相互接続する何れかの通信プロトコルを実施する何れかの適切なハードウェアであり得る。例えば、ノード・ライン2 1 6はギガビット・イーサネット（登録商標）を実施する銅線ケーブル又は光ファイバ・ケーブルであり得る。特定の実施例では、上記のように、ノード・ライン2 1 6は、1つ又は複数のリンク1 7 2を含む。

10

【0 0 3 6】

図5 Cは、そのグリッド1 1 0内に割り当てられる1つの仮想クラスタ2 2 0を備えているそのグリッド1 1 0を示す。1つの仮想クラスタ2 2 0しか示していないが、本願の開示の範囲から逸脱することなく、グリッド1 1 0における仮想クラスタ2 2 0の（ゼロを含む）何れかの数であり得る。仮想クラスタ2 2 0は、関連ジョブ1 5 0を処理する論理的なノード群1 1 5である。例えば、仮想クラスタ2 2 0は、同様なジョブ1 5 0を投入すると思われる、1つの研究グループ、部署、研究所、又は何れかの別のユーザ群と関連し得る。仮想クラスタ2 2 0は、何れかの形状であり得るものであり、グリッド1 1 0内に何れかの数のノード1 1 5を含み得る。実際に、図示する仮想クラスタ2 2 0が複数の物理的に隣接しているノード1 1 5を含んでいるが、クラスタ2 2 0は、ジョブ1 5 0を処理するよう動作可能な論理的に関連したノード1 1 5の分散クラスタであり得る。

20

【0 0 3 7】

仮想クラスタ2 2 0は、何れかの適切な時点で割り当て得る。例えば、クラスタ2 2 0は、システム1 0 0が初期化されると、例えば起動パラメータに基づいて割り当ててもよく、例えば、変動する、サーバ1 0 2のニーズに基づいて動的に割り当ててもよい。更に、仮想クラスタ2 2 0は経時的にその形状とサイズとを変えて、変動するリクエスト、デマンド、及び状況にすばやく対応し得る。例えば、仮想クラスタ2 2 0は、動的に変動させて、先行時点ではクラスタ2 2 0の部分であった第2ノード1 1 5の障害に応じて自動的に割り当てられる第1ノード1 1 5を含み得る。特定の実施例では、クラスタ2 2 0は、ノード1 1 5を処理の要求に応じて共有し得る。特定の実施例では、スケジューラ5 1 5が、以下に説明するように、スケジューリング・アルゴリズムによって1つ又は複数のジョブ1 5 0に1つ又は複数の仮想クラスタ2 2 0を割り当て得る。

30

【0 0 3 8】

図5 Dは、仮想クラスタ2 2 0例の内部に割り当てられる種々のジョブ空間2 3 0 a及び2 3 0 b各々を示す。一般的には、ジョブ空間2 3 0は、受信ジョブ1 5 0を完了するよう動的に割り当てられる仮想クラスタ2 2 0内のノード1 1 5の群である。通常、実行ジョブ1 5 0毎に1つのジョブ空間2 3 0が存在し、逆もあてはまるが、本願の開示の範囲から逸脱することなく、ジョブ空間2 3 0はノード1 1 5を共有し得る。ジョブ空間2 3 0のディメンションは、ユーザ又は管理者によって手作業で入力してもよく、ジョブ・パラメータ、ポリシー、及び／又は何れかの別の適切な特性に基づいて動的に判定してもよい。特定の実施例では、スケジューラ5 1 5は、以下に説明するように、スケジューリング・アルゴリズムによってジョブ空間2 3 0の1つ又は複数のディメンションを判定し得る。

40

【0 0 3 9】

図6 A及び図6 Bは、システム1 0 0による、管理グラフィカル・ユーザ・インタフェース4 0 0の種々の実施例を示す。多くの場合、管理G U I 4 0 0は、クライアント1 2 0にG U I 1 2 6を用いて提示される。一般的に、管理G U I 4 0 0は、種々の管理用対話画面若しくは管理用対話ディスプレイをシステム管理者に提示するもの及び／又は種々のジョブ投入画面若しくはジョブ・プロファイル画面をユーザに提示するものである。こ

50

これらの画面又はディスプレイは、種々の収集情報表示に組み立てられるグラフィカル構成要素から成る。例えば、G U I 4 0 0 は、（図 6 A に示す）グリッド 1 1 0 の物理的状态の表示、又は（図 6 B に示す）グリッド 1 1 0 におけるノード 1 1 5 の論理的な、割り当てすなわちトポロジの表示を提示し得る。

【0040】

図 6 A は例示的ディスプレイ 4 0 0 a を示す。ディスプレイ 4 0 0 a は、ノード 1 1 5 を効果的に管理するよう、管理者に提示される情報を含み得る。例示の実施例は、グリッド 1 1 0 の論理的な、「ピクチャ」すなわちスクリーンショットを備えている標準的なウェブ・ブラウザを含む。例えば、このピクチャはグリッド 1 1 0 と構成ノード 1 1 5 との物理的な状態を備え得る。各ノード 1 1 5 が何れかの数の色のうちの 1 つであり、各色が種々の状態を表す場合がある。例えば、障害ノード 1 1 5 が赤色であり、利用ノード又は割り当てノード 1 1 5 が黒色であり、非割り当てノード 1 1 5 がシェード・カラーである場合がある。更に、ディスプレイ 4 0 0 a は、管理者が、ノード 1 1 5 のうちの 1 つの上にポインタを移動させ、その種々の物理的属性を視ることを可能にし得る。例えば、管理者には、「ノード」、「利用可能性」、「プロセッサ利用率」、「メモリ利用率」、「温度」、「物理位置」、及び「アドレス」を含む情報が備えられる場合がある。当然、これらは例示的なデータ・フィールドに過ぎず、何れかの適切な物理ノード情報又は論理ノード情報が管理者用ディスプレイであり得る。ディスプレイ 4 0 0 a は、管理者が、グリッド 1 1 0 の表示を回転させるか何れかの別の適切な関数を行うことを可能にする場合もある。

【0041】

図 6 B は例示的なディスプレイ 4 0 0 b を示す。ディスプレイ 4 0 0 b は、グリッド 1 0 0 の論理的な状態の表示すなわちピクチャを提示する。図示する実施例は、グリッド 1 1 0 内に割り当てられる仮想クラスタ 2 2 0 を提示する。ディスプレイ 4 0 0 b は更に、1 つ又は複数のジョブ 1 5 0 を実行するよう、クラスタ 2 2 0 内に割り当てられる 2 つの例示的なジョブ空間 2 3 0 を表示する。ディスプレイ 4 0 0 b は、管理者に、（割り当て又は非割り当てなどの）種々の状態によってグループ化されるノード 1 1 5 の数を視るよう、ポインタをグラフィカル仮想クラスタ 2 2 0 上に移動させることを可能にし得る。更に、管理者は、適切なジョブ情報が提示されるようにジョブ空間 2 3 0 のうちの 1 つの上にポインタを移動させ得る。例えば、管理者は、ジョブ名、開始時間、ノード数、推定終了時間、プロセッサ利用度、I / O 利用度他を視ることができる場合がある。

【0042】

（例示的ディスプレイ 4 0 0 a 及び 4 0 0 b 各々によって上記に表す）管理 G U I 1 2 6 が例示的な目的のためのものにすぎず、図示するグラフィカル構成要素、更には図示しない別の管理構成要素を何ら含まない場合があり、それらの構成要素の一部又は全部を含む場合もあるということが分かるものである。

【0043】

図 7 は、システム 1 0 0 による、クラスタ管理エンジン 1 3 0 の一実施例を示す。この実施例では、クラスタ管理エンジン 1 3 0 は、複数のサブモジュール又は構成部分、すなわち、物理マネージャ 5 0 5、仮想マネージャ 5 1 0、ジョブ・スケジューラ 5 1 5、及び局所のメモリ又は変数 5 2 0 を含む。

【0044】

物理マネージャ 5 0 5 は、種々のノード 1 1 5 の物理的状态を判定し、この判定状態に基づいて効果的にノード 1 1 5 を管理するよう動作可能な何れかのソフトウェア、論理、ファームウェア、又は別のモジュールである。物理マネージャはこのデータを用いて、ノード 1 1 5 の障害を効率的に判定し、この障害に効率的に応答する場合がある。一実施例では、物理マネージャ 5 0 5 は複数のエージェント 1 3 2 に通信可能に結合され、各エージェントは 1 つのノード 1 1 5 上に存在する。上記のように、エージェント 1 3 2 は、少なくとも物理情報を収集し、マネージャ 5 0 5 に通信する。物理マネージャ 5 0 5 は更に、クライアント 1 2 0 のある場所にいるシステム管理者に警告を、ネットワーク 1 0 6 を

介して通信するよう動作される。

【0045】

仮想マネージャ510は、仮想クラスタ220と、ノード115の論理的状態とを管理するよう動作可能な何れかのソフトウェア、論理、ファームウェア、又は別のモジュールである。一般的に、仮想マネージャ510は、ノード115の論理的表現をノード115の物理的状态とリンクする。これらのリンクに基づいて、仮想マネージャ510は、ノード障害又は、HPC処理を増加させる（システム若しくはユーザの）要求に応じる場合などに、仮想クラスタを生成し、これらのクラスタ220に対する種々の変更を処理し得る。仮想マネージャ510は、非割り当てノード115などの仮想クラスタ220の状態をスケジューラ515に通信して、非実行状態又は待ち行列に入っている状態の、HPC処理及びHPCジョブ150の動的バックフィルを可能にする場合もある。仮想マネージャ510は更に、ジョブ150の特定ノード115との互換性を判定し、この情報をスケジューラ515に通信する場合がある。特定の実施例では、仮想マネージャ510は、個々の仮想クラスタ220を表すオブジェクトであり得る。

10

【0046】

特定の実施例では、クラスタ管理エンジン130は、スケジューラ515を含む。スケジューラ515は、スケジューリング・アルゴリズムによってジョブ150にノード115を割り当てる、ハードウェア構成部分、ソフトウェア構成部分、若しくは組み込み論理構成部分又は、1つか複数のそのような構成部分を含む。特定の実施例では、スケジューラ515はプラグインである。特定の実施例では、クラスタ管理エンジン150がジョブ150を受信することに応じて、クラスタ管理エンジン130は、ジョブ150に1つ又は複数のノード515を割り当てるようスケジューラ515をコールする。特定の実施例では、ジョブ150に1つ又は複数のノード515を割り当てるよう、クラスタ管理エンジン130がスケジューラ515をコールすると、クラスタ管理エンジン130は、ジョブ150に割り当てののに利用可能な、グリッド110におけるノード115をスケジューラ515に対して識別する。制約ではなく例として、クラスタ管理エンジン130が、ジョブ150に1つ又は複数のノード115を割り当てるようスケジューラ515をコールすると、クラスタ管理エンジン130は、ジョブ150への割り当てに利用可能な、グリッド110におけるノード115の全てのリストをスケジューラ515に通信し得る。特定の実施例では、クラスタ管理エンジン130は、ジョブ150に対する割り当てに利用可能なノード115の数がジョブ150に要求されるノード115の数以上である場合にのみ、ジョブ150に1つ又は複数のノード115を割り当てるよう、スケジューラ515をコールする。

20

30

【0047】

上記のように、特定の実施例では、グリッド110は、その各々が4つのCPU164に結合されるその3次元の、スイッチ166のトラスである。スケジューラ515は、グリッド110を、ノード115のトラスとして論理的に構成する。サイズが $[x, y, z]$ の、スイッチ166のトラスは、 $[4x, y, z]$ 、 $[x, 4y, z]$ 、 $[x, y, 4z]$ 、 $[2x, 2y, z]$ 、 $[2x, y, 2z]$ 、及び $[x, 2y, 2z]$ の6つの考えられる論理的な構成を備える。スケジューラ515が、ジョブ150に1つ又は複数のノード115を割り当てる場合、スケジューラ515はジョブ150に最も適した論理的な構成を選択し得る。

40

【0048】

メッセージ・パッシング・インタフェース(MPI)は、ジョブ150における処理間の通信の標準である。特定の実施例では、スケジューラ515はジョブ150に割り当てられる各ノード115にMPIランクを割り当てる。N個の処理を含むジョブ150について、スケジューラ515は各処理に0とN-1との間の一意の整数のランクを割り当てる。ジョブ150における第1処理にメッセージを通信するよう、ジョブ150における第2処理は、第1処理のランクを規定し得る。同様に、ジョブ150における第1処理からメッセージを受信するよう、ジョブ150における第2処理は、第1処理のランクを規

50

定し得る。同様に、ジョブ150における第1処理からメッセージを受信するよう、ジョブ150における第2処理は第1処理のランクを規定し得る。スケジューラ150は、その各々がその各々のブロードキャスト・グループにおける処理からその各々のブロードキャスト・グループにおける全ての別の処理までのメッセージの通信を容易にする、その1つ又は複数のブロードキャスト・グループを規定する場合もある。ブロードキャスト・グループにおける第1処理からメッセージを受信するよう、ブロードキャスト・グループにおける第2処理はブロードキャスト・グループを規定し得る。

【0049】

特定の実施例では、スケジューラ515は、「spatial」と「compact」と「any」との3つの種類の要求を処理する。「request」への言及は、適宜、ジョブ150に及び、適宜、その逆もあてはまる。ユーザがジョブ150をHPCサーバ102に投入すると、ユーザは要求の種類を規定し得る。「spatial」の要求は、空間的に記述されるジョブ150に及ぶ。既存のMPIアプリケーションのうちの1つのクラスは、ジョブ150における処理間の空間的な関係を呈する。例として、気象モデルがある。気象モデルを含むジョブ150を処理するよう、HPCサーバ102は、緯度経度（又は同様な座標システム）に及ぶ2次元グリッドを用いて、地球の表面を分割し、時間を離散的な時間ステップに分割する場合がある。ジョブ150の各処理は特定の領域の気象をモデル化する。各時間ステップの始めに、処理は、処理に隣接する4つの別の処理の各々と境界値を交換し、更に、特定の領域の気象を計算する。気象モデルを含むジョブ150を処理するよう、HPCサーバ102は、2次元のグリッドではなく、緯度経度と高度（又は同様な座標システム）に及ぶ3次元のグリッドを用いて地球の表面を分割し得る。

【0050】

ジョブ150における処理間で空間的な関係を呈するMPIアプリケーションについて、ユーザはジョブ150にノード115のトリプレット

【0051】

【数1】

$$\{S_x, S_y, S_z\}$$

を要求し得る。次元Sの全てが1よりも大きい場合、要求は3次元の要求である。次元Sのうちの1つが1に等しい場合、要求は2次元の要求である。次元Sのうちの2つが1に等しい場合、要求は1次元の要求である。要求をノード115に割り当てるよう、スケジューラ150は空間的な座標をMPIランクに：

【0052】

【数2】

$$[x, y, z] \rightarrow x \times S_x \times S_z + y \times S_z + z ;$$

としてマッピングし得る。S_x、S_y、及びS_zは要求のサイズを示し、xはゼロとS_xとの間であり、yはゼロとS_yとの間であり、zはゼロとS_zとの間である。ノード115を2次元の要求に割り当てるよう、スケジューラ150は、空間座標をMPIランクに：

【0053】

【数3】

$$[x, y] \rightarrow x \times S_y + y ;$$

としてマッピングし得る。特定の実施例では、空間座標をMPIランクにマッピングするよう、スケジューラ515はまず、グリッド110のz軸に沿って増加させ、次にグリッド110のy軸に沿って増加させ、更にグリッド110のx軸に沿って増加させる。スケジューラ515がMPIランクに空間座標をマッピングすることに関して正しくない前提

、例えば、まず、グリッド 1 1 0 の x 軸に沿って増加させ、次にグリッド 1 1 0 の y 軸に沿って増加させ、更にグリッド 1 1 0 の z 軸に沿って増加させるという前提に対処するよう、クラスタ管理エンジン 3 0 は要求ジョブ 1 5 0 を、例えば、

【0 0 5 4】

【数 4】

$$\{S_x, S_y, S_z\}$$

としてスケジューラ 5 1 5 に提示し得る。

【0 0 5 5】

10

「compact」要求は、空間的に記述されていないジョブ 1 5 0 に及ぶ。スケジューラ 5 1 5 は、コンパクト要求に割り当てられる、ノード 1 1 5 の対の各々の間の最大通信距離（又は最大ホップ・カウント）を最小にするよう、ノード 1 1 5 をコンパクト要求に割り当て得る。「any」要求は、処理間通信をほとんど必要としないか全く必要としないジョブ 1 5 0 に及ぶ。スケジューラ 1 5 0 は何れかのノード 1 1 5 の群を割り当てて何れかの要求を満たし得る。そのようなジョブ 1 5 0 は、グリッド 1 1 0 における断片化から生じるホールを充填する機会をスケジューラ 1 5 0 に備える。

【0 0 5 6】

ユーザは、ジョブ 1 5 0 を H P C サーバ 1 0 2 に投入する場合、ジョブ 1 5 0 にアグレシブ・フラグを規定する場合もある。特定の実施例では、アグレシブ・フラグは、ノード 1 1 5 をジョブ 1 5 0 に割り当てる目的でスケジューラ 5 1 5 に割り当てられる自由度を示す、ゼロと 1 との間の浮動小数である。高い数は低い数よりも自由度をスケジューラ 5 1 5 に与える。ユーザが空間的な要求を H P C サーバ 1 0 2 に投入し、空間的な要求上のアグレシブ・フラグをゼロに設定する場合、空間的な要求に対応するようノード 1 1 5 が利用可能である場合のみジョブ 1 5 0 をスケジューリングする。特定の実施例では、ユーザが空間的な要求を H P C サーバ 1 0 2 に投入し、空間的な要求上のアグレシブ・フラグをゼロより大きな数に設定する場合、スケジューラ 5 1 5 は空間的な要求に対応しようとするが、スケジューラ 5 1 5 は、空間的な要求に対応することが可能でない場合、ジョブ 1 5 0 をコンパクトな要求としてスケジューリングする。特定の実施例では、コンパクトな要求は、コンパクトな要求に割り当てられるノード 1 1 5 の対の間のホップ・カウントが無限であることを可能にし得る。クラスタ管理エンジン 1 3 0 は、割り当てに利用可能なノード 1 1 5 の数が要求ノード 1 1 5 数以上である場合のみ、スケジューラ 5 1 5 をコールするので、スケジューラ 1 5 0 は、そのような要求に常に対応することが可能である。特定の実施例では、コンパクトな要求のアグレシブ・フラグは、コンパクトな要求に割り当てられるノード 1 1 5 の対の間のホップ・カウントに対する限界を示す。そのような実施例では、ホップ・カウントに対する限界は

20

30

【0 0 5 7】

【数 5】

$$\frac{1}{1-a}$$

40

に等しい場合があり、a はアグレシブ・フラグである。

【0 0 5 8】

特定の実施例では、クラスタ管理エンジン 1 3 0 がスケジューラ 5 1 5 をコールして 1 つ又は複数のノード 1 1 5 をジョブ 1 5 0 に割り当てる場合、クラスタ管理エンジン 1 3 0 は：要求される、ノード 1 1 5 の数；要求種類；ジョブ 1 5 0 のサイズ；ジョブ 1 5 0 上のアグレシブ・フラグ；（スケジューラ 5 1 5 が後に調節してグリッド 1 1 0 のノード・ベースのサイズを判定する）グリッド 1 1 0 のスイッチ・ベースのサイズ；（特定の実施例では、4 に等しい、）スイッチ 1 6 6 毎のノード 1 1 5 の数；ジョブ 1 5 0 への割り

50

当てに利用可能なノード115の数；及び（例えば、ジョブ150への割り当てに利用可能なノード115全てのリストなどの）ジョブ150への割り当てに利用可能な1つ又は複数のノード115を識別するもの；の入力をスケジューラ515に備える。特定の実施例では、RequestedNodesは要求されるノード115の数を示し、RequestTypeは要求種類を示し、（アレイを含む）RequestedSizeはジョブ150のサイズを示し、AggressiveFlagはジョブ150上のアグレッシブ・フラグを示し、（アレイを含む）TorusSizeはグリッド110のスイッチ・ベースのサイズを示し、NodesPerSwitchはスイッチ166毎のノード115の数を示し、NumFreeNodesはジョブ150への割り当てに利用可能なノード115の数を示し、（アレイを含む）FreeNodeListはジョブ150への割り

10

【0059】

特定の実施例では、スケジューラ515がジョブ150をスケジューリングする（かスケジューリングしようとする）場合、スケジューラ515は、（ジョブ150に割り当てられるノード115のリストなどの）ジョブ150に割り当てられるノード115を識別するもの；ジョブ150に割り当てられる各ノードのMPIランク；及び（1）スケジューラ515がジョブ150をスケジューリングしたこと；（2）スケジューラ515はジョブ150をスケジューリングしなかったこと；又は（3）スケジューラ515がジョブ150をスケジューリングすることが全く可能でないということを示す戻り値；の出力を

20

【0060】

特定の実施例では、ノード115をジョブ150に割り当てよう、スケジューラ515はまず、ジョブ150をスケジューリングするよう変数を初期化し、次に変数によってジョブ150をスケジューリングし、クラスタ管理エンジン130で処理するようスケジュール（又は結果）を変換する。SpatialAllowed、CompactAllowed、及びAnyAllowedの3つの変数は、スケジューリングが可能な種類を示す。スケジューラ515は、SpatialAllowed、CompactAllowed、及びAnyAllowedを初期化するよう以下のロジックの例を用い得る：

【0061】

【表1】

30

- If the NodesRequested = 1
 - SpatialAllowed = False
 - CompactAllowed = False
 - AnyAllowed = True
- Else If RequestedType = SPATIAL
 - SpatialAllowed = True
 - AnyAllowed = False
 - If AggressiveFlag > 0
 - CompactAllowed = True
 - Else
 - CompactAllowed = False
- Else If RequestedType = Compact
 - SpatialAllowed = False
 - CompactAllowed = True
 - AnyAllowed = False
- Else If RequestedType = Any
 - SpatialAllowed = False
 - CompactAllowed = False
 - AnyAllowed = True

40

特定の実施例では、スケジューラ515は、グリッド110のより小さい次元の前にグ

50

リッド110のより大きな次元を示すよう、グリッド110のスイッチ・ベースのサイズを形成させる。(アレイを含む) TorusMapは、グリッド110のより小さい次元の前にグリッド110のより大きな次元を示すよう、グリッド110のスイッチ・ベースのサイズを形成させたグリッド110のスイッチ・ベースのサイズを示す。スケジューラ515は、TorusMapをFreeNodeListに識別されているノード115の全てに適用する。(アレイを含む) InverseTorusMapはTorusMapの逆数であり、スケジューラ515はInverseTorusMapを、処理するようクラスタ管理エンジン130にリストを戻す前にジョブ150に割り当てられるノード115のリストに適用する。制約ではなく例として、クラスタ管理エンジン130が14×16×15のスイッチ・ベースのトーラス・サイズをスケジューラ515に通信する場合、スケジューラ515はTorusMapを{2, 0, 1}に設定する。その場合、スイッチ・ベースのトーラス・サイズは16×15×14となり、{x, y, z}の指数を有するFreeNodeListにおけるノード155については、スケジューラ515がTorusMapを適用した後のノード155の指数は{y, z, x}である。上記例のInverseTorusMapは{1, 2, 0}である。

10

【0062】

特定の実施例では、NumMapDimensionsは、スイッチ・ベースのトーラスをノード・ベースのトーラスに変換する場合に修正する次元の数を示す。MapDimensions[2]及びMapMod[2]は、修正する次元の指数と、修正する次元の各々の乗数を備える。スケジューラ515は、修正する次元のうちの1つを4で掛けてもよく、修正する次元のうちの2つの各々を2で掛けてもよい。スケジューラ515は、どちらの乗算を適用するかを判定し、次に、適宜、当初はスイッチによって記述された、トーラスのサイズを修正する。スケジューラ515は、RequestTypeによって、どちらの乗算を適用するかを判定する。

20

【0063】

特定の実施例では、スケジューラ515は要求を満たすメッシュのリストを生成する要求に1つ又は複数の座標変換を適用する。メッシュは、グリッド110に埋め込まれるボックスを含む。

【0064】

【数6】

30

$$[S_x, S_y, S_z]$$

の始点と

【0065】

【数7】

$$[E_x, E_y, E_z]$$

の終点がメッシュを規定する。メッシュは、1つ又は複数の次元における終点よりも大きな始点を有する場合、1つ又は複数の次元において「ラッピング」する。制約ではなく例として、[3, 7, 5]での始点と[2, 9, 4]での始点とを備えているメッシュはx次元とy次元でラッピングする。グリッド110における点[x, y, z]は、

40

【0066】

【数8】

$$[S_x \leq x \leq E_x], [S_y \leq y \leq E_y], \text{及び} [S_z \leq z \leq E_z]$$

である場合、非ラッピング・メッシュに存在する。スケジューラ515が要求を満たすメッシュ・リストを生成した後、スケジューラ515は、要求への割り当てに利用可能なノード155の群に対してスケジューリング可能なメッシュをスケジューラ515が識別す

50

るまでスケジューラ 515 はリストをループする。一般的に、3次元の要求は、要求を満たす6つのメッシュをもたらす傾向にあり、2次元の要求は、要求を満たす数百のメッシュをもたらす傾向にあり、1次元の要求は、要求を満たす数十のメッシュをもたらす傾向にある。特定の実施例では、スケジューラ 515 は、要求を満たすメッシュ数を最大にするよう、2次元又は3次元の要求に対するノード・ベースのトラスを設定する。

【0067】

1次元の要求をスケジューリングする（か、1次元の要求に1つ又は複数のノード 115 を割り当てる）よう、変数を初期化するのに、スケジューラ 515 は、グリッド 110 におけるスイッチ 166 の y 軸と z 軸をノード 115 の 2×2 の構成に設定する。スケジューラ 515 は、グリッド 110 におけるスイッチ 166 の z 軸が未使用の次元であるよう

10

沿って y 軸にフォールディングする。したがって、特定の実施例では、1次元の要求には

：

【0068】

【表 2】

NumMapDimensions = 2

MapDimension[0] = 1

MapDimension[1] = 2

MapMod[0] = 2

MapMod[1] = 2

20

が適用され、 $[n]$ は 0 から $1 - n$ までの範囲の指数を有する 1次元のアレイを適宜、示す。制約ではなく例として、 $a = \{4, 6, 2\}$ は適宜、

【0069】

【数 9】

$a[0] = 4$ 、 $a[1] = 6$ 、及び $a[2] = 2$

30

に相当する。

【0070】

特定の実施例では、スケジューラ 515 は、2次元の要求をスケジューリングするよう、グリッド 110 におけるスイッチ 166 の y 軸と z 軸とをノード 115 の 2×2 の構成に設定して変数を初期化する場合もある。特定の実施例では、スケジューラ 515 は、スケジューリングするよう、2次元要求を第3の、未使用の次元にフォールディングして、よりコンパクトな形状を生成する。多くのそのようなフォールディングが考えられる場合があるので、スケジューラ 515 はそのようなフォールディングの最大数をもたらす（ノード 115 の 2×2 の構成とは異なり得る）構成を選択し得る。スケジューラ 515 は、2次元の要求について考えられる6つの構成の各々を検査し、6つの考えられる構成毎に考えられるフォールディングの数を計算する場合がある。特定の実施例では、スケジューラ 515 は最大の考えられるフォールディング数を可能にする構成を選択する。特定の実施例では、2つの 1×4 の構成が互角の場合、スケジューラ 515 はまず、z 軸を修正する 1×4 の構成を選択し、次に y 軸を修正する 1×4 の構成を選択する。特定の実施例では、 1×4 の構成と 2×2 の構成が互角の場合、スケジューラ 515 は 2×2 の構成を選択する。特定の実施例では、2つ以上の 2×2 の構成が互角の場合、スケジューラ 515 はまず、y 軸と z 軸とを修正する 2×2 の構成を選択し、次に x 軸と z 軸とを修正する 2×2 の構成を選択し、更に、x 軸と y 軸とを修正する 2×2 の構成を選択する。特定の実施例では、3次元の要求が、フォールディングを可能にする代わりにお互いに対してその

40

50

各々が一意のその6つの向き（又は回転）を可能にする以外は、2次元の要求をスケジューリングするよう、スケジューラ515が変数を初期化することになるので、スケジューラ515は3次元の要求をスケジューリングするよう変数を初期化する。

【0071】

特定の実施例では、コンパクトな要求をスケジューリングするよう変数を初期化するのに、スケジューラ515は、コンパクトな要求のz軸を4で掛けて1×4の構成を生成する。1×4の構成を用いて、コンパクトな要求を処理することは、コンパクトな要求に割り当てられるスイッチ166に結合されるノード115の全てを用いることを容易にし、このことは同様に、グリッド110におけるスイッチ・ポイントでの断片化を削減する。特定の実施例では、スケジューラ515は同様に、何れかの要求をスケジューリングするよう変数を初期化する。

10

【0072】

パーティションは、スケジューリングするのに利用可能な、グリッド110におけるノード115の全てを含む最小メッシュである。PartStart[3]はパーティションの始点座標を示し、PartEnd[3]はパーティションの終点座標を示し、PartSize[3]はパーティションのサイズを示し、PartWraps[3]はパーティションがラッピングするか否かを示す。スケジューラ515は要求を満たすノード115をサーチする長さを削減するようパーティションを構成し得る。パーティションは、グリッド110よりもずっと小さい場合がある。i=0、1、及び2の場合、PartStart[i]は、(アレイを含む)FreeMeshにおいて考えられるiの座標の全てのうちの最小の値を含む。PartSize[i]=PartEnd[i]-PartStart[i]+1である。PartSize[i]がTorusSize[i]に等しい場合、PartWraps[i]はTrue(真)である。スケジューラ515は(アレイを含む)NodeInUseをFreeMeshにおける全てのノードについてNODE__NOT__IN__USEに設定し、他のノードの全てについてNODE__IN__USEに設定する。

20

【0073】

特定の実施例では、FreeY[i, j, k]は、

【0074】

【数10】

30

$$\{i, \text{TorusSize}[1]-1, k\}$$

までの線 {i, j, k} に沿ったフリー・ノード155の数を含む。FreeX[i, j, k]は、

【0075】

【数11】

$$\{\text{TorusSize}[0]-1, j, k\}$$

までの線 {i, j, k} に沿ったフリー・ノード155の数を含む。スケジューラ515は、FreeY[i, j, k]とFreeX[i, j, k]とを用いて、以下に説明するようにスキャン・アルゴリズムを実行する。特定の実施例では、SpatialAllowed又はCompactAllowedがTrueである場合のみ、FreeY[i, j, k]とFreeX[i, j, k]とを構成する。

40

【0076】

SpatialAllowedがTrueである場合、スケジューラ515は、要求をスケジューリングするよう種々の構造を試みる。

【0077】

【数 1 2】

サイズ $S = \{S_x, S_y, S_z\}$ である空間的なジョブは、 $\{S_x, S_y, S_z\}$ 、 $\{S_x, S_z, S_y\}$ 、 $\{S_y, S_x, S_z\}$ 、 $\{S_y, S_z, S_x\}$ 、 $\{S_z, S_x, S_y\}$ 、及び $\{S_z, S_y, S_x\}$

の最大 6 つの一意の向きを有する。これらの 6 つの向きは、スケジューラ 515 がメッシュに適用し得る、4 つの一意の 90° の回転と 2 つの一意の 180° の回転とに相当する。いずれかの 2 つの次元がお互いに等しい場合、3 つの一意の向きのみが利用可能である。スケジューラ 515 はメッシュをスケジューリングする場合、考えられる向きを全て考慮する。ジョブ 150 が 2 次元である、すなわちジョブ 150 の 1 つの次元が 1 に等しい場合、スケジューラ 515 は、ジョブ 150 の 2 つの、使用次元すなわち、ジョブ 150 の 1 より大きい次元、の何れかを、ジョブ 150 の未使用次元、すなわちジョブ 150 の 1 に等しい次元にアコーディオンに似たかたちでフォールディングして、よりコンパクトな 3 次元のメッシュを生成し得る。スケジューラ 515 がフォールディングの長さの整数の倍数でない次元をフォールディングする場合、最後のフォールディングは先行するフォールディングの全てよりも短くなり、それによって 2 次元のメッシュが 3 次元メッシュ上に連結されることになる。ジョブ 150 が 1 次元の場合、スケジューラ 515 はジョブ 150 を 2 つの未使用次元の何れかにフォールディングし得る。スケジューラ 515 は更に、2 つの結果次元の何れかを残りの未使用次元にフォールディングし得る。メッシュの結果形状は、一般的に言えば、4 つのメッシュを連結したものとなる。

【0078】

図 8 は、y 次元にフォールディングされる 1 次元要求の例を示す。図 8 では、スケジューラ 515 は、2 次元メッシュ $\{1, 2, 4\}$ と、この 2 次元メッシュに連結される $\{1, 1, 3\}$ を生成するよう、 $\{1, 1, 11\}$ の 1 次元要求を 4 のフォールディング長を用いて y 次元にフォールディングしている。スケジューラ 515 は第 1 フォールディングにゼロの番号を付し、第 2 フォールディングに 1 の番号を付し、第 3 の短いフォールディングに 2 の番号を付す場合がある。スケジューラ 515 がフォールディングに沿ったノード 115 に MPI ランクを割り当てる場合、MPI ランクは、偶数番号のフォールディングに沿って z 値が増加するにつれ、かつ、奇数番号のフォールディングに沿って z 値が減少するにつれて増やされる。制約ではなく例として、 $[0, 0]$ でのノード 115 の MPI ランクはゼロである場合があり、 $[0, 1]$ でのノード 115 の MPI ランクは 1 である場合があり、 $[0, 2]$ でのノード 115 の MPI ランクは 2 である場合があり、 $[0, 3]$ でのノード 115 の MPI ランクは 3 である場合がある。 $[1, 3]$ でのノード 115 の MPI ランクは 4 である場合があり、 $[1, 2]$ でのノード 115 の MPI ランクは 5 である場合がある、などである。連結は $z = 0$ で開始するが、これはフォールディングを偶数を有するからである。スケジューラ 515 が奇数の完全なフォールディングを用いて要求をフォールディングした場合、連結はむしろ $z = 3$ で開始し、 $x = 0$ に向けて内側に進むことになる。特定の実施例では、スケジューラ 515 はアコーディオンに似たフォールディングのみを考慮する。別の種類のフォールディングも存在する。制約ではなく例として、フォールディングは階段形状を形成する場合がある。スケジューラ 515 は、1 次元のジョブ 150 に対する特定のフォールディングを禁止する場合がある。上記のように、特定の実施例では、スケジューラ 515 は 1 次元のジョブ 150 を 2 度フォールディングする。第 2 フォールディングは、スケジューラ 515 が最初にフォールディングした次元をフォールディングするか、スケジューラ 515 がフォールディングした対象の最初の次元をフォールディングする。図 8 では、スケジューラ 515 は、z 次元をフォールディングしており、y 次元にフォールディングしている。スケジューラ 515 が最初にフォールディングした次元を第 2 フォールディングがフォールディングした場合、スケジューラ 515 は、最大 3 つの連結を生成し、メッシュの合計を 4 つとする場合がある。特定の実施例では、スケジューラ 515 は、2 つの連結までしか可能にしない。その結果、スケジューラ 515 が 1 次元のジョブ 150 をスケジューリングする場合、第 1 フォールディングが連結をもたらさなかった場合でなければ、第 2 フォールディングは

、スケジューラ 515 が最初にフォールディングした対象の次元をフォールディングすることに制限される。ジョブ 150 のサイズがフォールディング長の整数の倍数であれば、連結は何ら生じない。特定の実施例では、そのような制限によって、スケジューラ 515 が 2 つの連結までしか可能にしないものとなる。特定の実施例では、スケジューラ 515 は当初、要求を満たす、考えられるメッシュ全てを構成する。要求が 1 次元か 2 次元の場合、スケジューラ 515 は、考えられる、アコーディオンに似たフォールディングの各々と、そのようなフォールディング各々の、考えられる向きの各々を構成する。要求が 3 次元の場合、スケジューラ 515 は、要求の、考えられる向きの各々を構成する。特定の実施例では、スケジューラ 515 は、以下に説明するように、Try Structure のリストを用いてそのような構成の各々を記録する。

10

【0079】

Compact Allowed が True である場合、スケジューラ 515 は、要求される、ノード 115 の数を含むコンパクトなメッシュを構成する。スケジューラ 515 はメッシュを最善のフィットと指定し、メッシュを（アレイを含む）Best Fit に記憶する。制約ではなく例として、N を、要求される、ノード 115 の数とし、Q を、N の立方根を整数に丸めたものとする。スケジューラは当初、Best Fit を {Q, Q, Q} に設定する。N = Q³ の場合、スケジューラ 515 は終了する。さもなければ、スケジューラ 515 は、以下に説明するように、Build Compact Fits 関数によって Best Fit の 1 つ又は複数の次元を増やすことになる。スケジューラ 515 は更に、Best Fit の次元以上でかつグリッド 110 の次元以下である次元を有するメッシュ全てを構成して、（アレイを含む）Fit を用いてメッシュを記録する。

20

【0080】

スケジューラ 515 は更に、望ましくないメッシュを Fit から取り除く。上記のように、特定の実施例では、グリッド 110 は、その各々が 4 つの CPU 164 に結合されるそのスイッチ 166 の 3 次元トラスである。スイッチ 166 毎に 4 つの CPU 164 を含むグリッド 110 を補正するよう、1 次元における 4 倍又は 2 次元における 2 倍によってトラスを修正する。スイッチ 166 での 1 つの CPU 164 が処理を実行する場合にスイッチ 166 での CPU 164 の全てが処理を実行するようにスケジューラ 515 が要求を満たす可能性を増加させるよう、増倍率の整数の倍数である 1 つ又は複数の修正次元におけるサイズを有するメッシュのみを保持する。制約ではなく例として、スケジューラ 515 が、y 次元におけるスイッチ 166 のトラスを 2 によって乗算し、z 次元におけるスイッチ 166 のトラスを 2 によって乗算する場合、スケジューラ 515 は、偶数の y 次元と z 次元とを有する Fit におけるメッシュのみを保持することになる。

30

【0081】

スケジューラ 515 は更に、残りのメッシュにおける最大ホップ・カウントによって Fit における残りのメッシュをソートする。サイズが

【0082】

【数 13】

$$\{S_x, S_y, S_z\}$$

40

のメッシュにおける何れかの 2 つのノード間の最大距離は、

【0083】

【数 14】

$$(S_x + 1) + (S_y - 1) + (S_z - 1)$$

である。2 つのメッシュが、お互いに同一の最大ホップ・カウントを有する場合、スケジューラ 515 は、メッシュを別のメッシュの前に近づけて立方体になるようにする。制約ではなく例として、

50

【0084】

【数15】

$$M_1 = \{4, 6, 16\} \text{ と } M_2 = \{8, 9, 9\}$$

は同じ最大距離を有するが、スケジューラ515は、 M_2 を M_1 の前に配置する。

【0085】

スケジューラ515は、望ましくないメッシュをFitから取り除かなかった場合でも、少なくともN個のノード115を含むメッシュ全てを生成する訳ではないことになる。

制約ではなく例として、Nが27に等しいものであり、BestFitが{3, 3, 3}に等しかった場合、Fitはメッシュ{1, 1, 27}を含むものでないことになる。{1, 1, 27}のメッシュは合理的なメッシュ数をもたらし、要求を満たす少なくとも1つのメッシュを常にもたらし、これは、Fitがグリッド110に等しいメッシュを含むことになり、Nがグリッド110におけるノード115の数以下である場合にのみクラスタ管理エンジン130がスケジューラ515をコールするからである。

10

【0086】

AnyAllowedが真である場合、1つ又は複数のフリー・メッシュを構成するよう、スケジューラ515は、フリー・ノード115を識別するまで、x軸を外部ループとし、y軸を次のループとし、z軸を内部ループとして、NodeInUseをループする。フリー・メッシュは、フリー・ノード115のみを含むメッシュを含み、フリー・ノード115はジョブ150に割り当て可能なノード115を含む。スケジューラ515は、NumFreeMeshesとFreeMesh[NumFreeMeshes]とを構成する。NumFreeMeshesはグリッド110におけるフリー・メッシュの数を示し、FreeMeshは、グリッド110における1つ又は複数のフリー・メッシュ構造を、グリッド110におけるフリー・メッシュ毎に識別するリストである。制約ではなく例として、ノード115の指数は

20

【0087】

【数16】

$$\{i1, j1, k1\}$$

30

であり得る。

スケジューラ515が、例えば

【0088】

【数17】

$$\{i1, j1, k2\}$$

などの非フリー・ノード115を識別するまでz軸を増やし得る。スケジューラ515はFreeMesh.start[2]をk1に設定する場合があります、FreeMesh.end[2]をk2-1に設定する場合があります。FreeMesh.start[2]はz軸に沿ったフリー・メッシュの開始値に相当し、FreeMesh.end[2]はフリー・メッシュの終了値に相当する。スケジューラ515は、

40

【0089】

【数18】

$$\{i1, j2, k1\} \text{ から } \{i1, j1, k2-1\}$$

までの線が少なくとも1つの非フリー・ノードを含むように、更に、y軸をj1から始め

50

て第1値 j_2 を識別し得る。スケジューラ 515 は更に、FreeMesh.start [1] を j_1 に設定し、FreeMesh.end [2] を $j_2 - 1$ に設定する。スケジューラ 515 は更に、

【0090】

【数19】

$$\{i_2, j_1, k_1\} \text{ から } \{i_2, j_2 - 1, k_2 - 1\}$$

までの線が少なくとも1つの非フリー・ノードを含むように、x軸を i_1 から始めて第1値 i_2 を識別する。スケジューラは更に、FreeMesh.start [0] を i_1 に設定し、FreeMesh.end [0] を

【0091】

【数20】

$$i_2 - 1$$

に設定する。スケジューラ 515 は、グリッド 110 における全てのノード 115 を対象とするよう、上記処理を繰り返す。上記処理は一意のフリー・メッシュ群をもたらすものでない。別の順序でループすることは、異なるフリー・メッシュ群を生成する傾向にあるが、それは2つ以上のフリー・メッシュがお互いに境界を共有する場合のみである。ノード 115 において全体が取り囲まれるフリー・メッシュは常に一意である。図9及び図10は、2次元の場合に、y軸を内部ループとして用いる場合とx軸を内部ループとして用いる場合との違いを示す。図9は、y軸を内部ループとして用いて構成される2つのフリー・メッシュを示し、図10は、x軸を内部ループとして用いて構成される2つのフリー・メッシュを示す。図9では、領域 530 は使用中のノード 115 を含み、領域 532 a は第1のフリー・メッシュであり、領域 532 b は第2のフリー・メッシュである。同様に、図10では、領域 530 は使用中のノード 115 を含み、領域 532 a は第1のフリー・メッシュであり、領域 532 b は第2のフリー・メッシュである。

【0092】

特定の実施例では、スケジューラ 515 は第1スケジューリング・アルゴリズムを用いて空間的な要求をスケジューリングし、第2スケジューリング・アルゴリズムを用いてコンパクトな要求をスケジューリングし、第3スケジューリング・アルゴリズムを用いて何れかの要求をスケジューリングする。第1スケジューリング・アルゴリズムと第2スケジューリング・アルゴリズムはお互いに同様であるが、お互いに比較的異なるスキャン・アルゴリズムを用いる。スケジューラ 515 がジョブ 150 をスケジューリングする場合、スケジューラ 515 はMPIランクによってAssignedNodeListにおけるジョブ 150 に割り当てられるノード 150 を示す、すなわち、AssignedNodeList [i] はMPIランク i を有する。 $\{S_x, S_y, S_z\}$ のサイズを有する空間的な要求をスケジュール化するよう、スケジューラ 515 は、スキャン・アルゴリズムを用いて、空間的な要求についてのNodeInUseにおける始点をサーチする。以下のロジック例はスキャン・アルゴリズムの記述例を備える。PartStartはパーティションの始点であり、PartEndはパーティションの終点であり、Tx, Ty, 及びTz は各々、x次元、y次元、及びz次元におけるトラス・サイズである。

【0093】

10

20

30

40

【表 3】

```

For x = PartStart[0] to PartEnd[0]
  For y = PartStart[1] to PartEnd[1]
    For z = PartStart[2] to PartEnd[2]
      Hit = True
      For i = x to x+Sx-1
        For j = y to y+Sy-1
          For k = z to z+Sz-1
            If (NodeInUse[i mod Tx, j mod Ty, k mod Tz] = NODE_IN_USE)
              Hit = False
            End If
          End For
        End For
      End For
      If (Hit = True)
        Return True
      End If
    End For
  End For
End For
Return False

```

10

特定の実施例では、コンパクトな要求に適用可能なスキャン・アルゴリズムは、以下の
ように、最も内部にあるループにおいて増やされるカウント値によって上記 Hit フラグを
置き換える。

20

```

Count = 0
For i = x to x+Sx-1
  For j = y to y+Sy-1
    For k = z to z+Sz-1
      If (NodeInUse[i mod Tx, j mod Ty, k mod Tz] = NODE_NOT_IN_USE)
        Count = Count + 1
      End If
    End For
  End For
End For
If (Count ≥ RequestedNodes)
  Return True
End If

```

30

25

【0094】

上記ロジックは比較的非効率的であるが、それはスケジューラ 515 が最大 $S_x \times S_y \times S_z$ 回まで `NodeInUse` における各点を評価するからである。コンパクト要求の上
記スキャンでは、 z のループが例えば、 $z1$ から $z1+1$ まで増やされるにつれ、 i 内部
ループと j 内部ループは変わることなく、 k ループは終点でしか変わらない。その結果、

【0095】

【数 21】

$$\{x, y, z1\} \text{ から } \{x+S_x, y+Sy-1, z1\}$$

40

50

までの 2 次元メッシュは更なる計算から除外され、スケジューラ 5 1 5 は、
 【0 0 9 6】
 【数 2 2】

$$\{x, y, (z1+1)+Sz-1\} \text{ から } \{x+Sx-1, y+Sy-1, (z1+1)+Sz-1\}$$

までの 2 次元メッシュを更なる計算に追加する。i 内部ループ、j 内部ループ、及び k 内部ループは、

【0 0 9 7】
 【数 2 3】

10

サイズ $\{Sx, Sy, 1\}$.

の z 軸に沿った 2 次元のメッシュのシーケンスにおけるフリー・ノード 1 1 5 をカウントする。z ループは 1 つのメッシュを取り除き、別のものを追加する。y ループでは同様な作用が y 軸に沿って生じる。（その両方ともアレイを含む）その Free X 及び Free Y は処理時間を削減することを容易にする。特定の実施例では、スケジューラ 5 1 5 は以下のアルゴリズムを用いて、コンパクトな要求をスキャンする：

【0 0 9 8】

【表 4】

- 2次元のメッシュ・カウントを記憶させる $zPlane[TorusSize[2]]$ を定義する。
- 次のように x ループ、 y ループ、及び z ループの、終点を計算する：
 - For $i = 0$ to 2
 - If $PartWraps[i] = True$, $end[i] = PartEnd[i]$
 - Else $end[i] = PartEnd[i] - Size[i]$
 - この場合、 x は $PartStart[0]$ から $End[0]$ まで、などでループすることになる。
- x ループ
 - $z = PartStart[2]$ to $PartEnd[2]$ について、 $zPlane$ をメッシュ $\{x, PartStart[1], \emptyset\}$ からメッシュ $\{x+Sx-1, PartStart[1]+Sy-1, \emptyset\}$ まで再計算する。
 - 特定の実施例では、スケジューラ 515 は 3 つのループをここでは用いることになる。ここで用いられる $FreeY$ はループ数を 2 つ： x の 1 ループと z の 1 ループ；に削減する。 $FreeY[x, PartStart[1], z] - FreeY[x, PartStart[1]+Sy, 2]$ は、 $\{x, PartStart[1], \emptyset\}$ 、 $\{x, PartStart[1]+Sy-1, \emptyset\}$ を含む $\{x, PartStart[1], z\}$ から $\{x, PartStart[1]+Sy-1, \emptyset\}$ までの線 $\{x, PartStart[1], \emptyset\}$ に沿ったフリー・ノード 115 の数を備える。
 - 以下の y ループについて $NewX = True$ に設定する。
- y ループ
 - If $NewX = True$
 - 何も行わない。
 - Else
 - $zPlane$ を更新する。
 - $z = PartStart[2]$ から $PartEnd[2]$ までの各々について
 - $\{x, y-1, \emptyset\}$ から $\{x+Sx-1, y-1, \emptyset\}$ までの線区間におけるフリー・ノード 115 を $Zplane[z]$ から減算する。
 - $FreeX[x, y-1, z] - FreeX[x+Sx, y-1, z]$ を用いて x にわたってループしないようにする。
 - $\{x, y+Sy-1, \emptyset\}$ から $\{x+Sx-1, y+Sy-1, \emptyset\}$ までの線区間におけるフリー・ノード 115 を $zPlane[z]$ に加算する。
 - $FreeX[x, y+Sy-1, z] - FreeX[x+Sx, y+Sy-1, z]$ を用いて x にわたってループしないようにする。
 - 次の y の増分について $NewX = False$ に設定する。
 - 以下の z ループについて $NewY = True$ に設定する。
- z ループ
 - If $NewY = True$
 - $zPlane$ を $z = PartStart[2]$ から $z = PartEnd[2]$ まで加算し、結果を $Count$ に記憶する。
 - Else
 - $zPlane[z-1]$ を $Count$ から減算する。
 - $\{x, y, z+Sz-1\}$ から $\{x+Sx-1, y+Sy-1, z+Sz-1\}$ までの 2 次元メッシュにおけるフリー・ノード 115 の和である $zPlane[z+Sz-1]$ を計算する。上記のように、 $FreeX$ を用いてループの数を 2 つから 1 つに削減する。
 - $zPlane[z+Sz-1]$ を $Count$ に加算する
 - If $Count \geq RequestedNodes$, Return True (True を戻す)

特定の実施例では、スケジューラ 515 は次元におけるパーティション・ラッピングに対応するよう以下の修正の 1 つ又は複数のものを適用する：(1) 次元における指数がアレイ限界を超える場合、スケジューラ 515 は、何れかのアレイ参照の前に指数にモジュラス関数を適用する；(2) パーティションが x 次元又は y 次元においてラッピングする場合、例えば点 a から点 b までの線区間についてフリー・ノード 115 を計算するよう、スケジューラ 515 は、 x 次元又は y 次元における点 a からパーティションの終点までの 1 つのものと、パーティションの始点から点 b までの別のものと 2 つの線区間についてフリー・ノード 115 を計算する。

【0099】

特定の実施例では、空間的な要求に適用可能なスキャン・アルゴリズムは、コンパクトな要求に適用可能な上記スキャン・アルゴリズムと同様である。特定の実施例では、空間的な要求に適用可能なスキャン・アルゴリズムとコンパクトな要求に適用可能な上記スキャン・アルゴリズムとの違いには以下のことが含まれる。すなわち、(1) スケジューラ 515 は、特定のカウンタを有するメッシュにおける点を識別する代わりに、ノード 115 の全てがフリーであるメッシュにおける点を探し、そのことはメモリ参照を削減する傾向にあるということ；と(2) スケジューラ 515 は、最大 2 つの別のメッシュをそのベース・メッシュに連結させたそのベース・メッシュを生成するようフォールディングされる、1 次元の要求又は 2 次元の要求を処理する場合があるので、1 つ又は複数の連結メッシュを処理する必要がある場合がある。特定の実施例では、スキャン・アルゴリズムに対するそのような修正は、 $16 \times 16 \times 16$ の構成をスケジューリングするスケジューラ 515 に関連した最大実行時間の桁を、1 つ又は複数、削減する傾向にある。

10

【0100】

空間的な要求をスケジューリングするよう、スケジューラ 515 は、スケジューリング可能な `Try Structure` を識別するまで `Try` 構造におけるリストにおける各 `Try` 構造にスキャン・アルゴリズムを適用するスケジューリング・アルゴリズムを用いる。リストにおける `Try` 構造でスケジューリング可能なものが何らなく、空間的な要求上のアグレシブ・フラグがゼロである場合、スケジューラ 515 は、空間的な要求をスケジューリングすることなくクラスタ管理エンジン 130 に返答する。さもないければ、スケジューラ 515 はコンパクトなスケジューリング・アルゴリズムを用いて空間的な要求をスケジューリングしようとする。

20

【0101】

特定の実施例では、空間的なアルゴリズムによる要求をスケジューリングすることには、最大、2 つのフォールディングと 1 つの回転との 3 つの変換が関係する。スケジューラ 515 は `Try` における以下のフィールドを用いて変換を追跡する：

【0102】

【表 5】

- **Try.rMap** は回転のマッピング関数である。**Try.rMap** は、点の指数をマッピングする 3 つの要素を有するアレイである。制約ではなく例として、**Try.rMap** = {1, 0, 2} であることは、指数 0 が 1 にマッピングされ、指数 1 が 0 にマッピングされ、指数 2 が 2 にマッピングされるので、このマップの下では、{x, y, z} → {y, x, z} であるということを表す。
- **Try.irMap** は **Try.rMap** の逆数である。
- **Try.NumFoldMaps** は **Try Structure** を生成するフォールディング数を示す。
- **Try.foldLength** はフォールディング長を示すアレイである。
- **Try.foldFrom** はフォールディングされる次元の指数を示すアレイである。制約ではなく例として、**Try.foldFrom**[i] = 2 は、i フォールディングは z 軸をフォールディングしたということを示す。
- **Try.foldTo** は、フォールディングの対象の次元の指数を示すアレイである。
- **Try.foldFix** は、固定の状態のままの次元の指数を示すアレイである。

10

特定の実施例では、**Try** 構造を用いてグリッド 110 における始点でジョブ 150 がスケジューリング可能であるということをスケジューラ 515 が判定した後、スケジューラ 515 は以下のように MP I ランクを割り当てる。

20

- スケジューラ 515 は逆回転マップを始点に適用して、始点を変換前のメッシュにマッピングする。スケジューラ 515 は、逆フォールディングを適用しなくてもよいように、フォールディングを構成してメッシュの始点を固定状態のままにする。
- スケジューラ 515 は、変換前のメッシュをループして MP I ランクを生成する。上記のように、特定の実施例では、x 軸は外部ループであり、y 軸は真ん中のループであり、z 軸は内部ループである。
- スケジューラ 515 は、変換前のメッシュにスケジューラ 515 が変換を適用した順序によってループにおける各点 {x, y, z} に、変換前のメッシュに適用された変換を適用する、すなわち、0 をフォールディングし、次に 1 をフォールディングし、更に上記点を回転させて点 {x, y, z} を変換前メッシュにおいて取得する。スケジューラ 515 は更に、AssignedNodeList の終わりにノード {x, y, z} を挿入する。

30

特定の実施例では、コンパクト・スケジューリング・アルゴリズムは、機能する **Try** 機能を識別するまで、**Try** 機能のリストにおける各メッシュにスキャン・アルゴリズムを適用する。リストにおけるメッシュ数は比較的大きい場合がある。制約ではなく例として、トーラスが $16 \times 16 \times 16$ のノード 115 を含み、要求が 100 のノード 115 に対するものである場合、**BestFit** = {4, 4, 5} であり、このことは、**Try** 構造リストにおいて 2000 を超えるメッシュをもたらす。2 進サーチを **Try** 構造に適用することは望ましい場合があるが、**Try** 構造リストの 2 進サーチは特定の実施例では機能しないものである。条件 C を含む 2 進サーチは：（1）C が要素 i について真であったならば、i 以上の全ての j について C が真であったということ；及び（2）C が要素 i について偽であったならば、i 以下の全ての j について C が偽であったということ；でない限り機能しないものである。特定の実施例では、**Try** 構造の 2 進サーチは機能しないものであるが、それは、例えばメッシュ $M1 = \{4, 4, 4\}$ を用いたスキャンが、要求を満たすのに十分なノードを見つけることになる一方、例えばメッシュ $M2 = \{2, 2, 10\}$ を用いたスキャンが、**Try** 構造リストにおいて M2 が M1 を上回っているにもかかわらず、要求を満たすのに十分なノードを見つけられないことになるという可能性が存在するからである。特定の実施例では、最大距離の 2 進サーチは機能する。スケジューラ 515 は、最大距離によって **Try** 構造リストにおけるメッシュをグループ化する場合、最大距離 i を有する、リストにおけるメッシュに対するフィットを識別すれば、i 以上

40

50

の j 全てについて、最大距離 j を有する、リストにおける少なくとも 1 つのメッシュもフィットすることになる。最大距離 i を有する、リストにおけるメッシュが何らフィットしない場合、 i 以下の最大距離を有する、リストにおけるメッシュも何らフィットしないことになる。制約ではなく例として、

【0 1 0 3】

【数 2 4】

$\{x, y, z\}$

が、フィットする最大距離 i を有するメッシュであると仮定する。したがって、

【0 1 0 4】

【数 2 5】

$\{x, y, z+1\}$ は $i+1$ の最大距離を有し、 $\{x, y, z+1\}$ は $\{x, y, z\}$ に及ぶので $\{x, y, z+1\}$

も機能する。帰納的推論は i 以上の j 全てにあてはまる。最大距離 i を有する、リストにおけるメッシュが何ら機能しない場合、最大距離 $i-1$ を有する何れかのメッシュ

【0 1 0 5】

【数 2 6】

$\{x, y, z\}$ に対して、 $\{x, y, z+1\}$ は最大距離 i を有し、 $\{x, y, z+1\}$

もフィットしない。

【0 1 0 6】

【数 2 7】

$\{x, y, z\}$ もフィットしないが、それは $\{x, y, z+1\}$ が $\{x, y, z\}$

に及ぶからである。よって、スケジューラ 515 は初期化中に `MaxDistance [NumMaxDistances, 2]` を構成する。

【0 1 0 7】

特定の実施例では、Fit におけるメッシュの 2 進サーチは最善のフィットを保証するものでないが、最善のフィットに対する相当好適な上限を備える。特定の実施例では、Fit におけるメッシュの 2 進サーチは効率的である、約 1000 のメッシュに対して約 10 のスキャンを生成する。スケジューラ 515 は、最大長に対する 2 進サーチを実行するか上限から下方線形サーチを実行するよう、上限を使用し得る。特定の実施例では、下方線形サーチのほうが、効率が高い傾向にある。

【0 1 0 8】

スケジューラ 515 は Fit に対する 2 進サーチを実行し、HighFit と HighStart [3] とを戻す。HighFit は要求を満たす Fit の指数であり、HighStart はグリッド 110 におけるフィットの始点である。下方線形サーチを実行するアルゴリズムは HighFit と HighStart とから開始する。特定の実施例では、スケジューラ 515 は現行の HighFit メッシュの最大距離を減らす。スケジューラ 515 は更に、要求を満たすメッシュをスケジューラ 515 が識別するまで、最大距離を含むメッシュ全てをループする。要求を満たすメッシュをスケジューラ 515 が識別する場合、スケジューラ 515 はメッシュを HighFit に設定し、もう一度最大距離を減らし、当該処理を繰り返す。スケジューラ 515 がそのようなメッシュを何ら識別しない場合、アルゴリズムの出口処理を行い、現行の HighFit が最善のフィットとなる。スケジューラ 515 は、特定の最大距離に対するフィットを識別することが可能でない場合、より短い最大距離に対するフィットを識別することが可能でない。

10

20

30

40

50

【0109】

スケジューラ515はFitメッシュをループし、AssignedNodeListの終わりに1つ又は複数のノード115を挿入する。3つのループの順序は、ノード・ベースのトーラスにスイッチ・ベースのトーラスをスケジューラ515がマッピングする方法によって変わってくる。スケジューラは、1つの次元において、 4×1 の構成を用いてスイッチ・ベースのトーラスをマッピングする場合、1つの次元が内部ループである。スケジューラ515は、2つの次元において、 2×2 の構成を用いてスイッチ・ベースのトーラスをマッピングする場合、2つの次元が最も内部にあるループである。

【0110】

何れかの要求をスケジューリングするよう、スケジューラ515は、ノード115の要求数を何れかの要求に割り当てるまで、FreeMeshをループし、何れかの要求を満たす。

10

【0111】

スケジューラ515は、FreeMeshをループするにつれ、ノード115をAssignedNodeList漸増的に挿入する。特定の実施例では、スケジューラ515は以下のようにFreeMeshをループする：

【0112】

【表6】

- z 軸が最も内部にあるループである。スケジューラ515は、スイッチ・ベースのトーラスをノード・ベースのトーラスに変換した際に z 軸を4倍に拡張している。 z 軸を、最も内部にあるループとして用いることは、スイッチ116に結合されるCPU164の断片化がないようにする傾向がある。
- FreeMeshにおける2つの残りの次元のうちの小さいほうは真ん中のループであり、2つの残りの次元のうちの大きいほうは最も外部にあるループである。

20

スケジューラ515は、MPIランクによるAssignedNodeListにおけるノード・ベースの座標を用いた選択ノード115を示す。AssignedNodeList[i, 0]はMPIランクiのノード115のx座標であり、AssignedNodeList[i, 1]はMPIランクiのノード115のy座標であり、AssignedNodeList[i, 2]はMPIランクiのノード115のz座標である。FreeNodeListはスイッチ・ベースの座標における、スケジューラ515に転送される、利用可能ノード115のリストである。特定の実施例では、FreeNodeListにおけるmpiRankフィールドを設定するよう、スケジューラ515は以下のアルゴリズム例を用いる：

30

【0113】

【表7】

- For $i = 0$ to NumFreeNodes - 1
 - AssignedNodeList[j]をスイッチ・ベースの座標に変換し結果をTo[4]に加算する。
 - InverseTorusMapをToの最初の3つの要素に適用する。
 - For $j = 0$ to NumFreeNodes - 1
 - If To[k] = FreeNodeList[j].coordinate[k] for all $k = 0, 1, 2, 3$
 - FreeNodeList[j].mpiRank = i
 - Exit j loop

40

以下のロジック例は、スケジューラ515の特定の実施例を記述する。特定の実施例では、ジョブ150をスケジューリングするよう、クラスタ管理エンジン130がスケジューラ515をコールする場合、クラスタ管理エンジン130はスケジューラ515に、以下の入力パラメータ値を伝達する。

【0114】

50

【表 8】

RequestedNodes:	要求されるノード 115 の数を示す。
RequestType:	要求種類を示す。SPATIAL、COMPACT、又は ANY に設定される。
RequestSize:	要求サイズを示す、3つの要素を有するアレイ。SPATIAL についてのみ有効。
AggressiveFlag:	ジョブ 150 にノード 115 を割り当てる目的でスケジューラ 515 に割り当てられる自由度を示す、ゼロと 1 との間の浮動小数
TorusSize:	グリッド 110 のスイッチ・ベースのサイズを示す、3つの要素を有するアレイ
NodesPerSwitch:	グリッド 110 における各スイッチ 166 に結合される CPU164 の数
NumFreeNodes:	FreeNodeList におけるノード 115 の数
FreeNodeList:	スケジューリングに利用可能なノード 115 のスイッチ・ベースの座標を示す FreeNode 構造のリスト

10

特定の実施例では、スケジューラ 515 は、ジョブ 150 をスケジューリングしようとした後、以下のうちの 1 つを戻す：

【0 1 1 5】

【表 9】

20

PQS_ASSIGNED:	スケジューラ 515 がジョブ 150 をスケジューリングしたことを示す。
PQS_NO_ASSIGNMENT_AT_SPECIFIED_TIME:	スケジューラ 515 がジョブ 150 をスケジューリングしていないことを示す。
PQS_NO_ASSIGNMENT_FOR_JOB_CATEGORY:	グリッド 110 におけるノード 115 全てが利用可能である場合でも、スケジューラ 515 がジョブ 150 をスケジューリングすることが全く可能でないことを示す。

30

スケジューラ 515 は、ジョブ 150 をスケジューリングする場合、適宜、FreeNode 構造の mpiRank フィールドを設定する。特定の実施例では、クラスタ管理エンジン 130 とスケジューラ 515 との間のラップ機能が、クラスタ管理エンジン 130 からの入力を、スケジューラ 515 が想定する形式に変換し、スケジューラ 515 からの出力を、クラスタ管理エンジン 130 が想定する形式に変換する。

【0 1 1 6】

特定の実施例では、ジョブ 150 が理論的にスケジューリング可能であるか否かを判定する setScheduleable は以下のロジック例を包含する：

【0 1 1 7】

40

【表 1 0】

```

If setSchedulable() = False
    Return PQS_NO_ASSIGNMENT_FOR_JOB_CATEGORY
End If
If initScheduler() = False
    Return PQS_NO_ASSIGNMENT_AT_SPECIFIED_TIME
End If
If RequestedNodes > NumFreeNodes
    ret = False
Else
    ret = scheduleJob()
End If
If ret = True
    setMpiRank()
    Return PQS_ASSIGNED
Else
    Return PQS_NO_ASSIGNMENT_AT_SPECIFIED_TIME
End If

```

10

特定の実施例では、ジョブ・サイズを評価するようスケジューラ 515 がコールする Rank は以下のロジック例を包含する。Rank に対する Input は 3 つの要素を有する 1 次元アレイ In [3] を含む。Rank からの Output は、In の指数をサイズの昇順によって示す 3 つの要素を有する 1 次元アレイ Rank [3] を含む。

20

【0 1 1 8】

【数 2 8】

$$\text{In}[\text{Rank}[0]] \leq \text{In}[\text{Rank}[1]] \leq \text{In}[\text{Rank}[2]]$$

である。特定の実施例では、Rank はバブル・アルゴリズムを含む。

【0 1 1 9】

【表 1 1】

30

```

Rank[0] = 0
Rank[1] = 1
Rank[2] = 2
For I = 0 to 2
    For j = i+1 to 2
        If In[Rank[j]] < In[Rank[i]]
            K = Rank[j]
            Rank[j] = Rank[i]
            Rank[i] = K
        End If
    End For
End For

```

40

特定の実施例では、ジョブ 150 が理論的にスケジューリング可能か否かを判定する setSchedulable は以下のロジックを包含する：

【0 1 2 0】

【表 1 2】

```

For i = 0 to 2
  If TorusSize[i] ≤ 1
    Return False
  End For
If RequestedNodes > TorusSize[0] × TorusSize[1] × TorusSize[2] × NodesPerSwitch
  Return False
End If
If NodesPerSwitch not equal to four
  Return False;
End If
If RequestType = SPATIAL
  factor[0] = 2
  factor[1] = 2
  Rank(TorusSize, tRank)
  Rank(RequestedSize, jRank)
  NumJobDim = 0
  NumExceed = 0
  For i = 0 to 2
    If RequestedSize[i] > 1)
      NumJobDim = NumJobDim + 1
    Else If RequestedSize[i] < 1
      Return False
    End If
    If RequestedSize[jRank[i]] > TorusSize[tRank[i]]
      Exceed[NumExceed] = i
      NumExceed = NumExceed + 1
    End If
  End For
  If NumExceed = 0
    Return True
  Else If NumExceed = 1
    If RequestedSize[jRank[Exceed[0]] ≤ NodesPerSwitch ×
      TorusSize[tRank[Exceed[0]]
      Return True
    End If
    If NumJobDim < 3
      Return True
    End If
    Return False
  Else
    If RequestedSize[jRank[Exceed[0]] ≤ factor[0] × TorusSize[tRank[Exceed[0]] and
      RequestedSize[jRank[Exceed[1]] ≤ factor[1] ×
      TorusSize[tRank[Exceed[1]]
      Return True
    End If
    If NumJobDim < 3 and (RequestedSize[jRank[Exceed[0]] ≤ NodesPerSwitch ×
      TorusSize[tRank[Exceed[0]] or RequestedSize[jRank[Exceed[1]]
      ≤ NodesPerSwitch × TorusSize[tRank[Exceed[1]])
      Return True
    End If
    return False
  End If
  return True

```

rは以下のロジック例を包含する。ジョブ150が1つのノード115しか要求しない場合、initSchedulerは、元の要求にかかわらず、可能な種類をAnyに設定する：

【0121】

【表 1 3】

```

If RequestedNodes = 1 or RequestType = Any
    AnyAllowed = True
    SpatialAllowed = False
    CompactAllowed = False
Else If RequestType = Compact
    CompactAllowed = True
    AnyAllowed = False
    SpatialAllowed = False
Else If RequestType = Spatial
    SpatialAllowed = True
    AnyAllowed = False
    If AggressiveFlag > 0
        CompactAllowed = True
    Else
        Compact Allowed = False
    End If
End If
factor[0] = 2
factor[1] = 2
Rank(TorusSize, tRank)
TorusMap[0] = tRank[2]
TorusMap[1] = tRank[1]
TorusMap[2] = tRank[0]
InverseTorusMap[tRank[0]] = 2
InverseTorusMap[tRank[1]] = 1
InverseTorusMap[tRank[2]] = 0
If SpatialAllowed = True
    If setTorusForSpatial() = False
        Return False
    End If
Else If CompactAllowed = True
    If setTorusForCompact() = False
        Return False
    End If
Else
    If setTorusForAny() = False
        Return False
    End If
End If
For i = 0 to NumMapDimensions
    TorusSize[mapDiminsions[i]] = mapMod[i] × TorusSize[mapDiminsions[i]]
End For
SetPartition()
If SpatialAllowed = True
    buildSpatialTries()
End If
If compactAllowed = True
    buildCompactFits()
End If
If AnyAllowed = True
    buildFreeMeshes()
End If
If SpatialAllowed = True or CompactAllowed = True
    InitScan()
End If
return True

```

ースのトーラスにマッピングする `setTorusForSpatial` は以下のロジック例を包含する：

【 0 1 2 2 】

【表 1 4】

```

Rank(RequestedSize, jRank)
NumDim = 0
dNdx = 0
For i = 0 to 2
    If RequestedSize[j] > 1)
        twoD[NumDim] = i
        NumDim = NumDim + 1
    Else
        oneD[dNdx] = i
        dNdx = dNdx + 1
    End If
End For
If NumDim = 1
    Return setTorusFor1D()
Else If NumDim = 2
    Return setTorusFor2D()
Else
    Return setTorusFor3D()
End If

```

10

20

特定の実施例では、ジョブ 150 の最も大きな次元と次に大きな次元 `jRank[2]` と `jRank[1]` とにおいてグリッド 110 を 2 倍にする `setTorusFor1D` は以下のロジック例を包含する：

【 0 1 2 3 】

【表 1 5】

```

NumMapDiminsions = 2
mapDiminsions[0] = jRank[2]
mapDiminsions[1] = jRank[1]
mapMod[0] = factor[0]
mapMod[1] = factor[0]
For i = 0 to 3
    ntSize[i] = TorusSize[TorusMap[i]]
End For
For i = 0 to 3
    TorusSize[i] = ntSize[i]
End For
For i = 0 to 3
    RequestedSize[i] = OriginalSize[jRank[i]]
    JobMap[jRank[i]] = i
End For
Return True

```

30

40

特定の実施例では、`setTorusFor2D` は 6 つの方法のうちの 1 つにおいて、スイッチ・ベースのトーラスをノード・ベースのトーラスにマッピングする：

【 0 1 2 4 】

【表 1 6】

1. $\{T[0], T[1], T[2]\} \rightarrow \{T[0], 2 \times T[1], 2 \times T[2]\}$
2. $\{T[0], T[1], T[2]\} \rightarrow \{2 \times T[0], T[1], 2 \times T[2]\}$
3. $\{T[0], T[1], T[2]\} \rightarrow \{2 \times T[0], 2 \times T[1], T[2]\}$
4. $\{T[0], T[1], T[2]\} \rightarrow \{T[0], T[1], 4 \times T[2]\}$
5. $\{T[0], T[1], T[2]\} \rightarrow \{T[0], 4 \times T[1], T[2]\}$
6. $\{T[0], T[1], T[2]\} \rightarrow \{4 \times T[0], T[1], T[2]\}$

10

TはTorus Sizeである。最初の3つの構成は、スイッチ166毎のノード115をスケジューラ515が 2×2 のノード115として構成することによってもたらされる。後の3つの構成は、スイッチ166毎のノード115をスケジューラ515が 1×1 のノード115として構成することによってもたらされる。特定の実施例では、setTorusFor2Dは、スケジューラ515がマップ毎に生成することになるTry構造をカウントし、最大のTry構造数を生成することになるマップを選択する。互角の場合には、setTorusFor2Dは上記の順序によってマップを選択する。スケジューラ515はpSize[6, 4]を：

【0125】

20

【表 1 7】

pSizes[i, 0]=構成*i*の*x*次元におけるパーティションのサイズ
 pSizes[i, 1]=構成*i*の*y*次元におけるパーティションのサイズ
 pSizes[i, 2]=構成*i*の*z*次元におけるパーティションのサイズ
 pSizes[i, 3]=構成*i*について生成されることになるトライの数

を含めるよう構成する。

【0126】

特定の実施例では、setTorusFor2Dは以下のロジック例を包含する：

30

【0127】

【表 1 8】

```

max = -1
maxNdx = -1
For i = 0 to 2
  For j = i+1 to 3
    NumMapDiminsions = 2
    mapDiminsions[0] = (i+j) mod 3
    mapDiminsions[1] = (i+j+1) mod 3
    mapMod[0] = factor[0]
    mapMod[1] = factor[1]
    setTestPartSize(testPartSize)
    pSizes[i+j-1, 2] = testPartSize[2]
    pSizes[i+j-1, 1] = testPartSize[1]
    pSizes[i+j-1, 0] = testPartSize[0]
    pSizes[i+j-1][3] = cnt2DTries(testPartSize, RequestedSize)
    If pSizes[i+j-1][3] > max
      max = pSizes[i+j-1][3]
      maxNdx = i+j-1
    End If
  End For
End For
For i = 0 to 3
  NumMapDiminsions = 1
  mapDiminsions[0] = 2 - i
  mapMod[0] = NodesperGrid
  setTestPartSize(testPartSize)
  pSizes[i+3, 2] = testspSize[2]
  pSizes[i+3, 1] = testspSize[1]
  pSizes[i+3, 0] = testspSize[0]
  pSizes[i+3][3] = cnt2DTries(testPartSize, RequestedSize)
  if pSizes[i+3][3] > max
    max = pSizes[i+3][3]
    maxNdx = i+3
  End If
End For
If max ≤ 0
  if CompactAllowed = True
    SpatialAllowed = False
    Return setTorusForCompact()
  Else
    return False
  End If
Else
  For i = 0 to 2
    ntSize[i] = TorusSize[TorusMap[i]]
  End For
  For i = 0 to 2
    TorusSize[i] = ntSize[i]
  End For

```

【表 19】

```

If maxNdx < 3
  NumMapDiminsions = 2
  mapDiminsions[0] = (maxNdx+1) mod 3
  mapDiminsions[1] = (maxNdx+2) mod 3
  mapMod[0] = factor[0]
  mapMod[1] = factor[1]
  RequestedSize[mapDiminsions[0]] = OriginalSize[jRank[1]]
  RequestedSize[mapDiminsions[1]] = OriginalSize[jRank[2]]
  RequestedSize[3 - mapDiminsions[0] - mapDiminsions[1]] = OriginalSize[jRank[0]]
  JobMap[jRank[1]] = mapDiminsions[0]
  JobMap[jRank[2]] = mapDiminsions[1]
  JobMap[jRank[0]] = 3 - mapDiminsions[0] - mapDiminsions[1]
Else
  NumMod = 1
  NumMapDiminsions = 1
  mapDiminsions[0] = (5 - maxNdx) mod 3
  mapMod[0] = NodesperGrid
  If mapDiminsions[0] = 2
    i = 1
  Else
    i = 2
  End If
  RequestedSize[mapDiminsions[0]] = OriginalSize[jRank[2]]
  RequestedSize[i] = OriginalSize[jRank[1]]
  RequestedSize[3 - mapDiminsions[0] - i] = OriginalSize[jRank[0]]
  JobMap[jRank[2]] = mapDiminsions[0]
  JobMap[jRank[1]] = i
  JobMap[jRank[0]] = 3 - mapDiminsions[0] - i
End If
End If
Return True

```

特定の実施例では、setTorusFor3Dは以下のロジック例を包含する：

【0129】

30

【表 2 0】

```

max = -1
maxNdx = -1
For i = 0 to 2
  For j = i+1 to 2
    NumMapDiminsions = 2
    mapDiminsions[0] = (i+j) mod 3
    mapDiminsions[1] = (i+j+1) mod 3
    mapMod[0] = factor[0]
    mapMod[1] = factor[1]
    setTestPartSize(testPartSize)
    pSizes[i+j-1, 2] = testPartSize[2]
    pSizes[i+j-1, 1] = testPartSize[1]
    pSizes[i+j-1, 0] = testPartSize[0]
    pSizes[i+j-1, 3] = cnt2DTries(testPartSize, RequestedSize)
    If (pSizes[i+j-1, 3] > max)
      max = pSizes[i+j-1, 3]
      maxNdx = i+j-1
    End If
  End For
End For
For i = 0 to 2
  NumMapDiminsions = 1
  mapDiminsions[0] = 2 - i
  mapMod[0] = NodesperGrid;
  setTestPartSize(testPartSize)
  pSizes[i+3, 2] = testPartSize[2]
  pSizes[i+3, 1] = testPartSize[1]
  pSizes[i+3, 0] = testPartSize[0]
  pSizes[i+3, 3] = cnt2DTries(testPartSize, RequestedSize)
  If pSizes[i+3][3] > max
    max = pSizes[i+3, 3]
    maxNdx = i+3
  End If
End For
If max ≤ 0
  If CompactAllowed = True
    SpatialAllowed = False
    Return setTorusForCompact()
  Else
    return False
  End If
Else
  For i = 0 to 2
    ntSize[i] = TorusSize[TorusMap[i]]
  End For
  For i = 0 to 2
    TorusSize[i] = ntSize[i]
  End For

```

【 0 1 3 0 】

【表 2 1】

```

If maxNdx < 3
  NumMod = 2
  mod[0] = (maxNdx+1) mod 3
  mod[1] = (maxNdx+2) mod 3
  NumMapDiminsions = 2
  mapDiminsions[0] = (maxNdx+1) mod 3
  mapDiminsions[1] = (maxNdx+2) mod 3
  mapMod[0] = factor[0]
  mapMod[1] = factor[1]
  RequestedSize[mapDiminsions[0]] = OriginalSize[jRank[1]]
  RequestedSize[mapDiminsions[1]] = OriginalSize[jRank[2]]
  RequestedSize[3 - mapDiminsions[0] - mapDiminsions[1]] = OriginalSize[jRank[0]]
  JobMap[jRank[1]] = mapDiminsions[0]
  JobMap[jRank[2]] = mapDiminsions[1]
  JobMap[jRank[0]] = 3 - mapDiminsions[0] - mapDiminsions[1]
Else
  NumMod = 1
  mod[0] = 2 - (maxNdx - 3)
  NumMapDiminsions = 1
  mapDiminsions[0] = (5 - maxNdx) mod 3
  mapMod[0] = NodesperGrid
  If mapDiminsions[0] = 2
    i = 1
  Else
    i = 2
  End If
  RequestedSize[mapDiminsions[0]] = OriginalSize[jRank[2]]
  RequestedSize[i] = OriginalSize[jRank[1]]
  requestedSize[3 - mapDiminsions[0] - i] = originalSize[jRank[0]];
  JobMap[jRank[2]] = mapDiminsions[0]
  JobMap[jRank[1]] = i
  JobMap[jRank[0]] = 3 - mapDiminsions[0] - i
End If
End If
Return True

```

特定の実施例では、コンパクトな要求の z 次元を 4×1 の構成に設定する `setTorusForCompact` は以下のロジック例を包含する：

【0 1 3 1】

【表 2 2】

```

For i = 0 to 3
  ntSize[i] = TorusSize[tMap[i]]
End For
For i = 0 to 3
  TorusSize[i] = ntSize[i]
End For
NumMapDiminsions = 1
mapDiminsions[0] = 2
mapMod[0] = NodesperGrid
Return True

```

特定の実施例では、何れかの要求の z 次元を 4×1 の構成に設定する `setTorusForAny` は以下のロジック例を包含する：

【0 1 3 2】

【表 2 3】

```
For i = 0 to 3
    ntSize[i] = TorusSize[tMap[i]]
End For
For i = 0 to 3
    TorusSize[i] = ntSize[i]
End For
NumMapDiminsions = 1
mapDiminsions[0] = 2
mapMod[0] = NodesperGrid
Return True
```

10

特定の実施例では、setPartitionは以下のロジック例を包含する：

【0 1 3 3】

【表 2 4】

```

For i = 0 to TorusSize[0] - 1
  For j = 0 to TorusSize[1] - 1
    For k = 0 to TorusSize[2] - 1
      NodeInUse[i,j,k] = NODE_IN_USE
    End For
  End For
End For
For i = 0 to 2
  PartStart[i] = TorusSize[i]
  PartEnd[i] = 0
End For
For i = 0 to NumFreeNodes - 1
  To[0] = FreeNodes[i].coordinate[TorusMap[0]]
  To[1] = FreeNodes[i].coordinate[TorusMap[1]]
  To[2] = FreeNodes[i].coordinate[TorusMap[2]]
  If NumMapDimensions = 1
    To[MapDimension[0]] = To[MapDimension[0]] × MapMod[0] +
      FreeNodes[i].coordinate[3]
  Else
    To[MapDimension[0]] = To[MapDimension[0]] × MapMod[0] +
      FreeNodes[i].coordinate[3] / MapMod[1]
    To[MapDimension[1]] = To[MapDimension[1]] × MapMod[1] +
      FreeNodes[i].coordinate[3] mod MapMod[1]
  End If
  NodeInUse[To[0], To[1], To[2]] = NODE_NOT_IN_USE
  For j = 0 to 2
    If To[j] < PartStart[j]
      PartStart[j] = To[j]
    End If
    If To[j] < PartStart[j]
      PartStart[j] = To[j]
    End If
  End For
End For
End For
For i = 0 to 2
  If PartStart[i] = 0 and PartEnd[i] = TorusSize[i] - 1
    PartWraps[i] = True
  Else
    PartWraps[i] = False
  End If
  PartSize[i] = PartEnd[i] - PartStart[i] + 1
End For

```

10

20

30

40

特定の実施例では、Free YとFree Xとを構成するinitScanは以下のロジック例を包含する：

【0 1 3 4】

【表 2 5】

```

For i = 0 to TorusSize[0] - 1
  For k = 0 to TorusSize[2] - 1
    Count = 0
    For j = TorusSize[1] - 1 to 0 by -1
      If NodeInUse[i,j,k] = NODE_NOT_IN_USE
        Count = Count + 1
      End If
      FreeY[i,j,k] = Count
    End For
  End For
End For
For j = 0 to TorusSize[1] - 1
  For k = 0 to TorusStSize[2] - 1
    Count = 0
    For i = TorusSize[0] - 1 to 0 by -1
      If NodeInUse[i,j,k] = NODE_NOT_IN_USE
        Count = Count + 1
      End If
      FreeX[i,j,k] = Count
    End For
  End For
End For
End For

```

10

20

特定の実施例では、要求における次元の数を判定する `buildSpatialTrees` は以下のロジック例を包含する：

【0 1 3 5】

【表 2 6】

```

NumDim = 0
For i = 0 to 2
  If RequestedSize[i] > 1)
    NumDim = NumDim + 1
  End If
End For
If NumDim = 1
  build1Dtry()
Else If NumDim = 2
  build2Dtry()
Else
  for i = 0 to 2
    Try.baseSize[i] RequestedSize[i]
  End For
  Try.NumConcat = 0
  Try.NumFoldMaps = 0
  NumberOfTries = 0
  build3Dtry(Try, NumberOfTries)
End If

```

30

40

特定の実施例では、3次元の要求について `TryList` を構築し、1次元の要求又は2次元の要求におけるフォールディング毎に `Try` 構造を構築する `build3Dtry` は以下のロジック例を包含する：

【0 1 3 6】

【表 2 7】

```

setOrient(Try, NumOrient, orient)
if NumOrient > 0
  For (i = 0 to NumOrient - 1
    ++NumTries;
    For j = 0 to 2
      TryList[NumberOfTries].baseSize[j] = Try.baseSize[orient[i, j]]
    End For
    TryList[NumberOfTries].NumConcat = Try.NumConcat;
    For j = 0 to TryList[NumberOfTries].NumConcat - 1
      For k = 0 to 2
        TryList[NumberOfTries].concatSize[j, k] = Try.concatSize[j, orient[i, k]];
        TryList[NumberOfTries].concatStartNode[j, k] = Try.concatStartNode[j,
          orient[i, k]];
      End For
    End For
    TryList[NumberOfTries].NumFoldMaps = Try.NumFoldMaps;
    For j = 0 to TryList[NumberOfTries].NumFoldMaps
      TryList[NumberOfTries].foldLength[j] = Try.foldLength[j]
      TryList[NumberOfTries].foldFrom[j] = Try.foldFrom[j]
      TryList[NumberOfTries].foldTo[j] = Try.foldTo[j]
      TryList[NumberOfTries].foldFix[j] = Try.foldFix[j]
    End For
    For k = 0 to 2
      TryList[NumberOfTries].rMap[k] = orient[i, k]
      TryList[NumberOfTries].irMap[orient[i, k]] = ;
    End For
    NumberOfTries = NumberOfTries + 1
  
```

特定の実施例では、回転毎の `Try` 構造と指数マップとに対して、一意の回転の数 `NumOrient` を計算する `setOrient` は以下のロジック例を包含する：

【0137】

30

【表 2 8】

```

NumOrient = 0;
If try.NumberOfConcatanations > 0
  For i= 0 to 2
    size[i] = try.baseSize[i];
    For j= 0 to try.NumConcat - 1
      If try.concatStartNode[j, i] ≥ size[i]
        size[i] = Try.concatStartNode[j, i] + Try.concatSize[j, i];
      Else If Try.concatStartNode[j, i] < 0
        size[i] = size[i] - try.concatStartNode[j, i]
      End If
    End For
  End For
  If size[0] ≤ PartSize[0] and size[1] ≤ PartSize[1] and size[2] ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 1
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If size[0] ≤ PartSize[0] and size[2] ≤ PartSize[1] and size[1] ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 2
    orient[NumOrient, 2] = 1
    NumOrient = NumOrient + 1
  End If
  If size[1] ≤ PartSize[0] and size[0] ≤ PartSize[1] and size[2] ≤ PartSize[2]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 0
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If size[1] ≤ PartSize[0] and size[2] ≤ PartSize[1] and size[0] ≤ PartSize[2]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 2
    orient[NumOrient, 2] = 0
    NumOrient = NumOrient + 1
  End If
  If size[2] ≤ PartSize[0] and size[0] ≤ PartSize[1] and size[1] ≤ PartSize[2]
    orient[NumOrient, 0] = 2
    orient[NumOrient, 1] = 0
    orient[NumOrient, 2] = 1
    NumOrient = NumOrient + 1
  End If
  If size[2] ≤ PartSize[0] and size[1] ≤ PartSize[1] and size[0] ≤ PartSize[2]
    orient[NumOrient, 0] = 2
    orient[NumOrient, 1] = 1
    orient[NumOrient, 2] = 0
    NumOrient = NumOrient + 1
  End If

```

【 0 1 3 8 】

【表 2 9】

```

Else If Try.baseSize[0] = Try.baseSize[1]
  If try.baseSize[0] = try.baseSize[2]
    If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[1] and
      Try.baseSize[2] ≤ PartSize[2]
      orient[NumOrient, 0] = 0
      orient[NumOrient, 1] = 1
      orient[NumOrient, 2] = 2
      NumOrient = NumOrient + 1
    End If
  Else
    If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[1] and
      Try.baseSize[2] ≤ PartSize[2]
      orient[NumOrient, 0] = 0
      orient[NumOrient, 1] = 1
      orient[NumOrient, 2] = 2
      NumOrient = NumOrient + 1
    End If
    If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[2] ≤ PartSize[1] and
      Try.baseSize[1] ≤ PartSize[2]
      orient[NumOrient, 0] = 0
      orient[NumOrient, 1] = 2
      orient[NumOrient, 2] = 1
      NumOrient = NumOrient + 1
    End If
    If Try.baseSize[2] ≤ PartSize[0] and Try.baseSize[0] ≤ PartSize[1] and
      Try.baseSize[1] ≤ PartSize[2]
      orient[NumOrient, 0] = 2
      orient[NumOrient, 1] = 0
      orient[NumOrient, 2] = 1
      NumOrient = NumOrient + 1
    End If
  End if
Else if Try.baseSize[0] = Try.baseSize[2]
  If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 1
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[2] and Try.baseSize[1]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 2
    orient[NumOrient, 2] = 1
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[1] ≤ PartSize[0] and Try.baseSize[0] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 0
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If

```

【 0 1 3 9 】

50

【表 3 0】

```

Else If Try.baseSize[1] = Try ≥ baseSize[2])
  If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 1
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[1] ≤ PartSize[0] and Try.baseSize[0] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 0
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[1] ≤ PartSize[0] and Try.baseSize[2] ≤ PartSize[1] and Try.baseSize[0]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 2
    orient[NumOrient, 2] = 0
    NumOrient = NumOrient + 1
  End If
Else
  If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 1
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[0] ≤ PartSize[0] and Try.baseSize[2] ≤ PartSize[1] and Try.baseSize[1]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 0
    orient[NumOrient, 1] = 2
    orient[NumOrient, 2] = 1
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[1] ≤ PartSize[0] and Try.baseSize[0] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[2]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 0
    orient[NumOrient, 2] = 2
    NumOrient = NumOrient + 1
  End If
  If Try.baseSize[1] ≤ PartSize[0] and Try.baseSize[2] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[0]
    orient[NumOrient, 0] = 1
    orient[NumOrient, 1] = 2
    orient[NumOrient, 2] = 0
    NumOrient = NumOrient + 1
  End If

```

【 0 1 4 0 】

【表 3 1】

```
If Try.baseSize[2] ≤ PartSize[0] and Try.baseSize[0] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[1]
    orient[NumOrient, 0] = 2
    orient[NumOrient, 1] = 0
    orient[NumOrient, 2] = 1
    NumOrient = NumOrient + 1
End If
If Try.baseSize[2] ≤ PartSize[0] and Try.baseSize[1] ≤ PartSize[1] and Try.baseSize[2]
    ≤ PartSize[0]
    orient[NumOrient, 0] = 2
    orient[NumOrient, 1] = 1
    orient[NumOrient, 2] = 0
    NumOrient = NumOrient + 1
End If
End If
```

10

特定の実施例では、build2Dtryは以下のロジック例を包含する：

【0 1 4 1】

【表 3 2】

```

Rank(PartSize, pRank)
build2DFold(PartSize, pRank, RequestedSize, NumFolds, FoldList)
For i = 0 to NumFolds - 1
    d1 = RequestedSize[FoldList[i].fixDimension] + FoldList[i].foldLength +
        FoldList[i].NumFolds
    If FoldList[i].remainder not equal 0
        d1 = d1 + 1
    End If
    For j = i + 1 to NumFolds - 1
        D2 = RequestedSize[FoldList[j].fixDimension] + FoldList[j].foldLength +
            FoldList[j].NumFolds
        If FoldList[j].remainder not equal 0
            D2 = d2 + 1
        End If
        If d2 < d1
            TempFold = FoldList[j]
            FoldList[j] = FoldList[i]
            FoldList[i] = TempFold
            d1 = d2
        End If
    End For
End For
NumberOfTries = 0
For i = 0 to NumFolds - 1
    try.baseSize[FoldList[i].fixDimension] = RequestedSize[FoldList[i].fixDimension]
    try.baseSize[FoldList[i].foldDimension] = FoldList[i].foldLength
    try.baseSize[FoldList[i].oneDimension] = FoldList[i].NumFolds
    If FoldList[i].remainder not equal 0
        try.NumConcat = 1
        If FoldList[i].NumFolds is odd
            Try.concatStartNode[0, FoldList[i]. foldDimension] = FoldList[i].foldLength -
                FoldList[i].remainder
        Else
            Try.concatStartNode[0, FoldList[i]. foldDimension] = 0
        End If
        try.concatStartNode[0, FoldList[i]. fixDimension] = 0
        try.concatStartNode[0, FoldList[i]. oneDimension] = FoldList[i].NumFolds
        try.concatSize[0, FoldList[i]. fixDimension] = try.baseSize[FoldList[i]. fixDimension]
        try.concatSize[0, FoldList[i]. foldDimension] = FoldList[i]. remainder
        try.concatSize[0, FoldList[i]. oneDimension] = 1
    Else
        try.NumConcat = 0
    End If
    try.NumFoldMaps = 1
    try.foldLength[0] = FoldList[i].foldLength
    try.foldFrom[0] = FoldList[i].foldDimension
    try.foldTo[0] = FoldList[i]. oneDimension
    try.foldFix[0] = FoldList[i].fixDimension
    build3Dtry(Try, NumberOfTries)
End For

```

特定の実施例では、2次元のメッシュの、考えられるフォールディング全てを構築する
b u i l d 2 D f o l d は以下のロジック例を包含する：

【 0 1 4 2 】

【表 3 3】

```

j=0
oneD=-1
For i=0 to 2
  If size[j]=1 and oneD=-1
    oneD=i
  Else
    twoD[j]=1
    j=j+1
  End If
End For
If size[twoD[1]] ≥ size[twoD[0]]
  bigD=twoD[1]
  littleD=twoD[0]
Else
  bigD=twoD[0]
  littleD=twoD[1]
End If
startFoldB=sqrt(size[bigD])
If startFoldB × startFoldB not equal size[bigD] or startFoldB=1
  StartFoldB=startFoldB+1
End If
endFoldB=size[bigD]/2
startFoldL=sqrt(size[littleD])
If startFoldL × startFoldL not equal size[littleD] or startFoldL=1
  StartFoldL=startFoldL+1
if size[bigD] not equal size[littleD]
  endFoldL=size[littleD]/2
else
  endFoldL=1
End If
NumFolds=1
If endFoldB ≥ startFoldB
  NumFolds=NumFolds+(endFoldB-startFoldB+1)
End If
If endFoldL ≥ startFoldL
  NumFolds=NumFolds+(endFoldL-startFoldL+1)
End If
foldIndex=0;
FoldList[foldIndex].foldLength=size[littleD]
FoldList[foldIndex].NumFolds=1
FoldList[foldIndex].remainder=0
FoldList[foldIndex].foldD=littleD
FoldList[foldIndex].fixD=bigD
FoldList[foldIndex].oneD=oneD

```

以下のロジック例によって構成されるアレイ t は結果の T r y のメッシュ・サイズである。スケジューラ 5 1 5 は t の R a n k をアレイ t R a n k に記録する：

【 0 1 4 3 】

【表 3 4】

```

    sz[littleD] = size[bigD]
    sz[bigD] = FoldList[foldIndex].foldLength
    sz[oneD] = FoldList[foldIndex].NumFolds
    rank(t, tRank)
    hit = False
    For i1 = 0 to 2 while hit = False
    If sz[tRank[i1]] > PartSize[pRank[i1]]
        hit = True
    End If
    If hit = False
        foldIndex = foldIndex + 1
    End If
    For i = startFoldB to endFoldB
        FoldList[foldIndex].foldLength = i
        FoldList[foldIndex].NumFolds = size[bigD] / i
        FoldList[foldIndex].remainder = size[bigD] mod i
        FoldList[foldIndex].foldD = bigD
        FoldList[foldIndex].fixD = littleD
        FoldList[foldIndex].oneD = oneD
        sz[littleD] = size[littleD]
        sz[bigD] = FoldList[foldIndex].foldLength
        If (FoldList[foldIndex].remainder not equal 0
            sz[oneD] = FoldList[foldIndex].NumFolds + 1
        Else
            sz[oneD] = FoldList[foldIndex].NumFolds
        End If
        Rank(t, tRank)
        hit = False
        For i1 = 0 to 2 while hit = False
            If sz[tRank[i1]] > PartSize[pRank[i1]]
                hit = True
            End If
        End For
        if hit = False
            foldIndex = foldIndex + 1
        End If
    End For
    For i = startFoldL to endFoldL
        FoldList[foldIndex].foldLength = i
        FoldList[foldIndex].NumFolds = size[littleD] / i
        FoldList[foldIndex].remainder = size[littleD] mod i
        FoldList[foldIndex].foldD = littleD
        FoldList[foldIndex].fixD = bigD
        FoldList[foldIndex].oneD = oneD
        sz[bigD] = size[bigD]
        sz[littleD] = FoldList[foldIndex].foldLength
        If FoldList[foldIndex].remainder not equal 0
            sz[oneD] = FoldList[foldIndex].NumFolds + 1
        Else
            sz[oneD] = FoldList[foldIndex].NumFolds
        End If
    End For

```

【表 3 5】

```

Rank(t, tRank)
hit = False
for i1 = 0 to 2 while hit = False
  If tRank[i1] > PartSize[pRank[i1]]
    hit = True
  End If
End For
If hit = False
  FoldIndex = foldIndex + 1
End If
End For

```

10

特定の実施例では、`buildTry`は、1次元の要求のフォールディングのリストを生成し、フォールディング毎に、`build2DFold`をコールして、1つ又は複数の追加のフォールディングのリストを生成する。`buildTry`は、以下の構造例を包含する`OneDFoldList`にフォールディングのリストを記録する：

【0145】

【表 3 6】

```

Structure oneDFold
  Fold Structure oneD
  Fold Structure twoD[x]
  integer NumTwoDFolds
  integer twoDFoldSize[3]
End Structure

```

20

特定の実施例では、`oneD`は第1フォールディングを含む。特定の実施例では、`twoD`は第1フォールディングから生成されるフォールディングのリストを含む。`NumTwoDFolds`は、`twoD`におけるフォールディング数を示す。特定の実施例では、`build2DFold`に転送されるメッシュ・サイズを示す。スケジューラ515は、`twoD`の要素について`Try`構造を生成し、`build3Dtry`をコールして各`Try`構造の、考えられる回転全てを構築する。特定の実施例では、`buildTry`は以下のロジック例を包含する：

30

【0146】

【表 3 7】

```

Rank(PartSize, pRank)
Rank(RequestedSize, jRank[0])
end = sqrt(RequestedSize[jRank[2]])
start = 2
OneDFoldList[0].oneD.foldLength = RequestedSize[jRank[2]]
OneDFoldList[0].oneD.NumFolds = 1
OneDFoldList[0].oneD.remainer = 0
OneDFoldList[0].oneD.foldD = jRank[2]
OneDFoldList[0].oneD.oneD = jRank[1]
OneDFoldList[0].oneD.fixD = jRank[0]
OneDFoldList[0].twoDFoldSize[jRank[2]] = RequestedSize[jRank[2]]
OneDFoldList[0].twoDFoldSize[jRank[1]] = 1
OneDFoldList[0].twoDFoldSize[jRank[0]] = 1
hit = False
For j = 0 to 2 while hit = False
    if RequestedSize[jRank[j]] > PartSize[pRank[j]]
        hit = True
    End If
End For
If hit = False
    build2DFold(PartSize, pRank, RequestedSize, OneDFoldList[0].twoD,
                OneDFoldList[0].nTwoDFolds)
    OneDFoldList[0].nTwoDFolds = 1
    Num1DFolds = 1;
Else
    Num1DFolds = 0
End If
gotRemZero = False
For i = start to end
    OneDFoldList[Num1DFolds].oneD.foldLength = i
    OneDFoldList[Num1DFolds].oneD.NumFolds = RequestedSize[jRank[2]] / i
    OneDFoldList[Num1DFolds].oneD.remainer = RequestedSize[jRank[2]] mod i
    OneDFoldList[Num1DFolds].oneD.foldD = jRank[2]
    (OneDFoldList[Num1DFolds].oneD.oneD = jRank[1]
    OneDFoldList[Num1DFolds].oneD.fixD = jRank[0]
    OneDFoldList[Num1DFolds].twoDFoldSize[jRank[2]] =
        OneDFoldList[Num1DFolds].oneD.foldLength
    OneDFoldList[Num1DFolds].twoDFoldSize[jRank[1]] =
    OneDFoldList[Num1DFolds].oneD.NumFolds
    OneDFoldList[Num1DFolds].twoDFoldSize[jRank[0]] = 1
    If OneDFoldList[Num1DFolds].oneD.remainer not equal 0 or gotRemZero = False
        If OneDFoldList[Num1DFolds].oneD.remainer = 0
            gotRemZero = True
        End If
        build2DFold(PartSize, pRank, RequestedSize,
                    OneDFoldList[Num1DFolds].twoDFoldSize,
                    OneDFoldList[Num1DFolds].twoD,
                    OneDFoldList[Num1DFolds].nTwoDFolds)
        Num1DFolds = Num1DFolds + 1
    End If
End For

```

【0 1 4 7】

【表 3 8】

```

NumberOfTries = 0
For i = 0 to Num1DFolds
  For j = 0 to OneDFoldList[i].nTwoDFolds
    If OneDFoldList[i].oneD.foldD not equal OneDFoldList[i].twoD[j].foldD or
      OneDFoldList[i].oneD.remainer = 0
      try.baseSize[OneDFoldList[i].twoD[j].fixD] =
        OneDFoldList[i].twoDFoldSize[OneDFoldList[i].twoD[j].fixD]
      try.baseSize[OneDFoldList[i].twoD[j].foldD] =
        OneDFoldList[i].twoD[j].foldLength
      try.baseSize[OneDFoldList[i].twoD[j].oneD] =
        OneDFoldList[i].twoD[j].NumFolds;
    if OneDFoldList[i].twoD[j].remainder not equal 0
      try.NumConcat = 1
      if OneDFoldList[i].twoD[j].NumFolds is odd
        try.concatStartNode[0, OneDFoldList[i].twoD[j].foldD] =
          OneDFoldList[i].twoD[j].foldLength -
          OneDFoldList[i].twoD[j].remainder
      Else
        try.concatStartNode[0, OneDFoldList[i].twoD[j].foldD] = 0
    End If
    try.concatStartNode[0, OneDFoldList[i].twoD[j].fixD] = 0
    try.concatStartNode[0, OneDFoldList[i].twoD[j].oneD] =
      OneDFoldList[i].twoD[j].NumFolds
    try.concatSize[0, OneDFoldList[i].twoD[j].fixD] =
      try.baseSize[OneDFoldList[i].twoD[j].fixD]
    try.concatSize[0, OneDFoldList[i].twoD[j].foldD] =
      OneDFoldList[i].twoD[j].remainder
    try.concatSize[0 OneDFoldList[i].twoD[j].oneD] = 1;
  Else
    try.NumConcat = 0
  End If
  If OneDFoldList[i].oneD.remainer not equal 0
    if OneDFoldList[i].oneD.NumFolds is odd
      try.concatStartNode[try.NumConcat, OneDFoldList[i].oneD.foldD] =
        OneDFoldList[i].oneD.foldLength -
        OneDFoldList[i].oneD.remainer
    Else
      try.concatStartNode[try.NumConcat, OneDFoldList[i].oneD.foldD] = 0
    End If
    try.concatStartNode[try.NumConcat, OneDFoldList[i].oneD.fixD] = 0
    try.concatStartNode[try.NumConcat, OneDFoldList[i].oneD.oneD] =
      OneDFoldList[i].oneD.NumFolds
    try.concatSize[try.NumConcat, OneDFoldList[i].oneD.fixD] = 1
    try.concatSize[try.NumConcat, OneDFoldList[i].oneD.foldD] =
      OneDFoldList[i].oneD.remainer
    try.concatSize[try.NumConcat, OneDFoldList[i].oneD.oneD] = 1
    oneDEnd[0] = try.concatStartNode[try.NumConcat, 0] +
      try.concatSize[try.NumConcat, 0] - 1
    oneDEnd[1] = try.concatStartNode[try.NumConcat, 1] +
      try.concatSize[try.NumConcat, 1] - 1
  
```

【表 3 9】

```

oneDEnd[2] = try.concatStartNode[try.NumConcat, 2] +
              try.concatSize[try.NumConcat, 2] - 1
k = try.concatStartNode[try.NumConcat, OneDFoldList[i].twoD[j].foldD]
/= oneDEnd[OneDFoldList[i].twoD[j].foldD]
If OneDFoldList[i].twoD[j].NumFolds is odd
    try.concatStartNode[try.NumConcat, OneDFoldList[i].twoD[j].foldD] =
        OneDFoldList[i].twoD[j].foldLength - 1 - (k mod
        OneDFoldList[i].twoD[j].foldLength)
    oneDEnd[OneDFoldList[i].twoD[j].foldD] =
        OneDFoldList[i].oneD.foldLength - 1 - (1 mod
        OneDFoldList[i].oneD.foldLength)
Else
    try.concatStartNode[try.NumConcat, OneDFoldList[i].twoD[j].foldD] = k
    mod OneDFoldList[i].twoD[j].foldLength
    oneDEnd[OneDFoldList[i].twoD[j].foldD] = 1 mod
    OneDFoldList[i].oneD.foldLength
End If
try.concatStartNode[try.NumConcat, OneDFoldList[i].oneD.oneD] = k /
    OneDFoldList[i].twoD.foldLength
oneDEnd[OneDFoldList[i].oneD.oneD] = 1 / OneDFoldList[i].oneD.foldLength
try.concatSize[try.NumConcat, 0] = oneDEnd[0] -
    try.concatStartNode[try.NumConcat, 0] + 1
try.concatSize[try.NumConcat, 1] = oneDEnd[1] -
    try.concatStartNode[try.NumConcat, 1] + 1
try.concatSize[try.NumConcat, 2] = oneDEnd[2] -
    try.concatStartNode[try.NumConcat, 2] + 1
try.NumConcat = try.NumConcat + 1
End If
try.NumFoldMaps = 2
try.foldLength[0] = OneDFoldList[i].oneD.foldLength
try.foldFrom[0] = OneDFoldList[i].oneD.foldD
try.foldTo[0] = OneDFoldList[i].oneD.oneD
try.foldFix[0] = OneDFoldList[i].oneD.fixD
try.foldLength[1] = OneDFoldList[i].twoD[j].foldLength
try.foldFrom[1] = OneDFoldList[i].twoD[j].foldD
try.foldTo[1] = OneDFoldList[i].twoD[j].oneD
try.foldFix[1] = OneDFoldList[i].twoD[j].fixD
build3Dtry(Try, NumberOfTries)
End For
End For
NumDeleted = 0
For i = 0 to NumberOfTries - 1
    curMax = TryList[i].baseSize[0] + TryList[i].baseSize[1] + TryList[i].baseSize[2]
    if TryList[i].NumConcat > 0
        curMax = curMax + 1
    End If
    For j = i + 1 to NumberOfTries - 1
        duplicate = True
        For i1 = 0 to 2 while duplicate = True
            If TryList[i1].baseSize[i] not equal TryList[i].baseSize[i]
                duplicate = False
            End If
        End For
    End For
End For

```

【 0 1 4 9 】

50

【表 4 0】

```

If duplicate = True and TryList[j].NumConcat = TryList[i].NumConcat)
  For i1 = 0 to TryList[i].NumConcat while duplicate = True
    For j1 = 0 to 2 while duplicate = True
      If TryList[j].concatStartNode[i1, j1] not equal
        TryList[i].concatStartNode[i1, j1]
        duplicate = False
      Else If TryList[j].concatSize[i1, j1] not equal TryList[i].concatSize[i1, j1]
        duplicate = False
    End For
  End For
End For
End If
If duplicate = True
  For i1 = 0 to 2
    TryList[j].baseSize[i1] = TorusSize[i1] + 1
  End For
  NumDeleted = NumDeleted + 1
Else
  nxtMax = TryList[j].baseSize[0] + TryList[j].baseSize[1] + TryList[j].baseSize[2]
  If TryList[j].NumConcat > 0
    nxtMax = nxtMax + 1
  End If
  If nxtMax < curMax
    TempTry = TryList[j]
    TryList[j] = TryList[i]
    TryList[i] = tempTry
    curMax = nxtMax
  End If
End If
End For
End For
NumberOfTries = NumberOfTries - NumDeleted

```

10

20

30

特定の実施例では、BestFit [3] を構成する buildCompactFit
s は以下のロジック例を包含する：

【0150】

【表 4 1】

```

Rank(PartSize,PartRank)
l = QubeRoot(ResuestedNodes)
hit = False
For i = 1 to l+1 while hit = False
  For j = i to l+1 while hit = False
    For (k = j to l+1 while hit = False
      If  $i \times j \times k \geq \text{RequestedNodes}$ 
        d[0] = i
        d[1] = j
        d[2] = k
        hit = True
      End If
    End For
  End For
End For
If d[0] ≤ PartSize[PartRank[0]]
  If d[1] > PartSize[PartRank[1]]
    d[1] = d[1] - 1
    hit = False
    For d[2] = RequestedNodes / (d[0] × d[1]) to PartSize[PartRank[2]] while hit = False
      If d[0] × d[1] × d[2] ≥ RequestedNodes
        Hit = True
      End If
    End For
  End If
Else
  d[0] = PartSize[PartRank[0]]
  l = sqrt(RequestedNodes / d[0])
  hit = False;
  For j = l to l + 1 while hit = False
    For (k = j to l + 1 while hit = False
      If (d[0] × j × k ≥ RequestedNodes
        d[1] = j
        d[2] = k
        hit = True
      End If
    End For
  End For
  if d[1] > PartSize[PartRank[1]]
    d[1] = PartSize[PartRank[1]]
    d[2] = RequestedNodes / (d[0] × d[1])
    If d[0] × d[1] × d[2] < RequestedNodes
      d[2] = d[2] + 1
    End If
  End If
End If

```

【 0 1 5 1】

【表 4 2】

```

bestFit[pRank[0]] = d[0];
bestFit[pRank[1]] = d[1];
bestFit[pRank[2]] = d[2];
NumberOfFits = 0
For i = BestFit[0] to PartSize[0]
  For j = BestFit[1] to PartSize[1]
    For k = BestFit[2] to PartSize[2]
      Fit[NumberOfFits,0] = i
      Fit[NumberOfFits,1] = j
      Fit[NumberOfFits,2] = k
      Hit = True
      If (i not equal to PartSize[0]) and (j not equal to PartSize[1]) and (k not equal to
        PartSize[2])
        For m = 0 to NumMapDimensions While Hit = True
          If Fit[NumberOfFits,MapDimension[m]] mod MapMod[m] not equal to 0
            Hit = False
          End If
        End For
      End If
      If Hit = True
        NumberOfFits = NumberOfFits + 1
      End If
    End For
  End For
End For
For i = 0 to NumBerOffits - 1
  d1 = Fit[i, 0] + Fit[i, 1] + Fit[i, 2]
  For j = i + 1 to NumBerOffits - 1
    d2 = Fit[j, 0] + Fit[j, 1] + Fit[j, 2]
    if d2 < d1
      k = Fit[j, 0]
      Fit[j, 0] = Fit[i, 0]
      Fit[i, 0] = k
      k = Fit[j, 1]
      Fit[j, 1] = Fit[i, 1]
      Fit[i, 1] = k
      k = Fit[j, 2]
      Fit[j, 2] = Fit[i, 2]
      Fit[i, 2] = k
      d1 = d2
    Else If d2 = d1
      Rank(Fit[i], iRank)
      Rank(Fit[j], jRank)
      hit = 0
      For k = 0 to 2 while hit = 0
        If Fit[j, jRank[k]] > Fit[i, iRank[k]]
          hit = 1
        Else If Fit[j, jRank[k]] < Fit[i, iRank[k]]
          Hit = -1
        End For
    End For
  End For
End For

```

【 0 1 5 2 】

【表 4 3】

```

        If hit = 1
            k = Fit[j, 0]
            Fit[j, 0] = Fit[i, 0]
            Fit[i, 0] = k
            k = Fit[j, 1]
            Fit[j, 1] = Fit[i, 1]
            Fit[i, 1] = k
            k = Fit[j, 1]
            Fit[j, 1] = Fit[i, 1]
            Fit[i, 1] = k
            d1 = d2
        End If
    End If
End For
lastMax = 0
NumMaxDistances = 0
For i = 0 NumberOfFits - 1
    currentMax = Fit[i, 0] + Fit[i, 1] + Fit[i, 2]
    If currentMax not equal lastMax
        MaxDistance[NumberOfMaxDistance, 0] = i
        MaxDistance[NumberOfMaxDistance, 1] = currentMax
        NumberOfMaxDistance = NumberOfMaxDistance + 1
    End If
End For

```

10

20

特定の実施例では、buildFreeMeshes Functionは以下のロジック例を包含する：

【0 1 5 3】

【表 4 4】

```

NumFreeMeshes = 0
For i = partStart[0] to PartEnd[0]
  For j = PartStart[1] to PartEnd[1]
    For k = PartStart[2] to PartEnd[2]
      If NodeInUse[i,j,k] = NODE_NOT_IN_USE
        NodeInUse[i,j,k] = NODE_ON_HOLD
        meshStart[0] = i
        meshStart[1] = j
        meshStart[2] = k
        inMesh = True
        for mz = k + 1 to PartEnd[2] and inMesh = True
          if NodeInUse[i,j,mz] not equal NODE_NOT_IN_USE
            inMesh = False
          End If
        End For
        If inMesh = True
          mEnd[2] = mz - 1
        Else
          mEnd[2] = mz - 2
        End If
        If PartWraps[2] and meshStart[2] = 0 and meshEnd[2] not equal PartEnd[2]
          inMesh = True;
          For mz = PartEnd[2] to meshEnd[2] by -1 and inMesh = True
            If NodeInUse [i,j,mz] not equal NODE_NOT_IN_USE
              inMesh = False
            End If
          End For
          If inMesh = True
            mz = mz + 1
          Else
            mz = mz + 2
          End If
          if mz ≤ PartEnd[2]
            meshStart[2] = mz;
            meshEnd[2] = meshEnd[2] + TorusSize[2]
          End If
        End If
        inMesh = True
        For my = j + 1 to PartEnd[1] and inMesh = True
          For mz = meshStart[2] to meshEnd[2] and inMesh = True
            If NodeInUse[i, my, mz mod TorusSize[2]] not equal
              NODE_NOT_IN_USE
              inMesh = False
            End If
          End For
        End For
      End For
    End For
  End For
End For

```

【0 1 5 4】

【表 4 5】

```

    If inMesh = True
        meshEnd[1] = my - 1
    Else
        meshEnd[1] = my - 2
    End If
    If PartWraps[1] and meshStart[1] = 0 and meshEnd[1] not equal
        PartEnd[1]
        inMesh = True
        For my = PartEnd[1] to meshEnd[1] by -1 and inMesh = True
            For mz = meshStart[2] to meshEnd[2] and inMesh = True
                If NodeInUse[i,my,mz mod Torus Size[2] not equal
                    NODE_NOT_IN_USE
                    inMesh = False
                End If
            End For
        End For
        If inMesh = True
            My = my + 1
        Else
            my = my + 2
        End If
        if my ≤ PartEnd[1]
            meshStart[1] = my
            meshEnd[1] = meshEnd[1] + TorusSize[1]
        End If
    End If
End For
inMesh = True
for mx = i + 1 to PartEnd[0] and inMesh = True
    for my = meshStart[1] to meshEnd[1] and inMesh = True
        for mz = mStart[2] to mEnd[2] and inMesh = True
            If NodeInUse[mx,my mod TorusSize[1],mz mod TorusSize[2]] not
                equal NODE_NOT_IN_USE
                inMesh = False
            End If
        End For
    End For
End For
End For
If inMesh = True
    meshEnd[0] = mx - 1
Else
    meshEnd[0] = mx - 2
End If
If partWraps[0] and meshStart[0] = 0 and meshEnd[0] not equal PartEnd[0]
    inMesh = True
    For mx = partEnd[0] to meshEnd[0] by -1 and inMesh = True
        For my = meshStart[1] to meshEnd[1] and inMesh = True
            For mz = meshStart[2] to meshEnd[2] and inMesh = True
                If NodeInUse[mx,my mod TorusSize[1],mz Mod TorusSize[2]]
                    not equal NODE_NOT_IN_USE
                    inMesh = False
                End If
            End For
        End For
    End For
End For

```

【 0 1 5 5 】

【表 4 6】

```

    If inMesh = True
        Mx = mx + 1
    Else
        Mx = mx + 2
    End If
    If mx ≤ PartEnd[0]
        meshStart[0] = mx
        meshEnd[0] = meshEnd[0] + TorusSize[0]
    End If
End If
FreeMesh[NumFreeMeshes].Start[0] = meshStart[0]
FreeMesh[NumFreeMeshes].Start[1] = meshStart[1]
FreeMesh[NumFreeMeshes].Start[2] = meshStart[2]
FreeMesh[NumFreeMeshes].end[0] = meshEnd[0]
FreeMesh[NumFreeMeshes].end[1] = meshEnd[1]
FreeMesh[NumFreeMeshes].end[2] = meshEnd[2]
FreeMesh[NumFreeMeshes].NumNodes = (meshEnd[0] - meshStart[0] + 1)
                                     × (meshEnd[1] - meshStart[1] + 1)
                                     × (meshEnd[2] - meshStart[2] + 1)
For mx = meshStart[0] to meshEnd[0]
    mx1 = mx mod TorusSize[0]
    For my = meshStart[1] to meshEnd[1]
        my1 = my mod TorusSize[1]
        For mz = meshStart[2] to meshEnd[2]
            mz1 = mz mod TorusSize[2]
            NodeInUse[mx1, my1, mz1] = NODE_ON_HOLD
        End For
    End For
End For
For i = 0 to 2
    FreeMesh[NumFreeMeshes].Rank[i] = 2 - i
End For
For l = 0 to 2
    For m = l+1 to 3
        l1 = FreeMesh[NumFreeMeshes].Rank[l]
        m1 = FreeMesh[NumFreeMeshes].Rank[m]
        If meshEnd[m1] - meshStart[m1] < meshEnd[l1] - meshStart[l1]
            FreeMesh[NumFreeMeshes].Rank[l] = m1
            FreeMeshRank[m] = l1
        End If
    End For
End For
NumFreeMeshes = NumFreeMeshes + 1
End If
End For
End For
End For
For i = partStart[0] to PartEnd[0]
    For j = PartStart[1] to PartEnd[1]
        For k = PartStart[2] to PartEnd[2]
            If NodeInUse[i, j, k] = NODE_ON_HOLD
                NodeInUse[i, j, k] = NODE_NOT_IN_USE
            End If
        End For
    End For
End For

```

【表 4 7】

```

End For
End For
For i= 0 to NumFreeMeshes - 1
  For j= i+1 to NumFreeMeshes - 1
    hit = False
    if FreeMesh[j].NumNodes < freeMesh[i].NumNodes
      hit = True;
    Else If FreeMesh[j].NumNodes = freeMesh[i].NumNodes
      hit = True
      For l= 0 to 2 while hit = True
        If FreeMesh[j].Rank[l] > freeMesh[i].Rank[l]
          Hit = False
        End If
      End For
    End If
    If hit = True
      TempMesh = FreeMesh[j]
      FreeMesh[j] = FreeMesh[i]
      FreeMesh[i] = TempMesh
    End If
  End For
End For

```

10
20

特定の実施例では、スケジューラ 5 1 5 がジョブ 1 5 0 を正常にスケジューリングする場合 True を戻す ScheduleJob は以下のロジック例を包含する：

【 0 1 5 7】

【表 4 8】

```

If SpatialAllowed = True
  If scheduleSpatial() = True
    return True
  Else If CompactAllowed = True
    return scheduleCompact()
  End If
Else If CompactAllowed = True
  return scheduleCompact()
Else
  Return scheduleAny()
End If

```

30

特定の実施例では、scheduleSpatial は以下のロジック例を包含する：

【 0 1 5 8】

【表 4 9】

```

GotFit = False
For i= 0 to NumberOfTries - 1 while GotFit = False
  If scanSpatial(TryList[i],Start) = True
    GotFit = True
    setSpatialNodeInUse(Try, Start)
  End If
End For
Return GotFit

```

特定の実施例では、AssignedNodeListを構築するsetSpatialNodeInUseは以下のロジック例を包含する：

【0159】

【表 5 0】

```

NodeIndex = 0
For (cNode[0] = 0 to OriginalSize[0] - 1
  For cNode[1] = 0 to OriginalSize[1] - 1
    For cNode[2] = 0 to OriginalSize[2] - 1
      For i = 0 to 2
        jcNode[jobMap[i]] = cNode[i]
      End For
      If Try.NumFoldMaps = 1
        mNode[0, Try.foldFix[0]] = jcNode[Try.foldFix[0]]
        mNode[0, Try.foldTo[0]] = jcNode[Try.foldFrom[0]] / Try.foldLength[0]
        If mNode[0, Try.foldTo[0]] is odd
          mNode[0, Try.foldFrom[0]] = Try.foldLength[0] - 1
            -(jcNode[Try.foldFrom[0]] mod
              Try.foldLength[0])
        Else
          mNode[0, Try.foldFrom[0]] = jcNode[Try.foldFrom[0]] mod
            Try.foldLength[0]
        End If
        For i = 0 to 2
          node[i] = mNode[0, Try.rMap[i]]
        End For
      Else
        mNode[0, Try.foldFix[0]] = jcNode[Try.foldFix[0]]
        mNode[0, Try.foldTo[0]] = jcNode[Try.foldFrom[0]] / Try → foldLnt[0]
        If mNode[0, Try.foldTo[0]] is odd
          mNode[0, Try.foldFrom[0]] = Try.foldLength[0] - 1
            -(jcNode[Try.foldFrom[0]] mod
              Try.foldLength[0])
        Else
          mNode[0, Try.foldFrom[0]] = jcNode[Try.foldFrom[0]] mod
            Try.foldLength[0]
        End If
        mNode[1, Try.foldFix[1]] = mNode[0, Try.foldFix[1]]
        mNode[1, Try.foldTo[1]] = mNode[0, Try.foldFrom[1]] / Try.foldLength[1]
        If mNode[1, Try.foldTo[1]] is odd
          mNode[1, Try.foldFrom[1]] = Try.foldLength[1] - 1 - (mNode[0,
            Try.foldFrom[1]] mod Try.foldLength[1])
        Else
          mNode[1, Try.foldFrom[1]] = mNode[0, Try.foldFrom[1]]
            mod Try → foldLnt[1]
        End If
        For i = 0 to 2
          node[i] = mNode[1, Try.rMap[i]]
        End For
      End If
      For i = 0 to 2
        Node[i] = node[i] mod TorusSize[i]
      End For
      NodeInUse[node[0], node[1], node[2]] = NODE_IN_USE
      AssignedNodeList[NodeIndex, 0] = node[0]
      AssignedNodeList[NodeIndex, 1] = node[1]
      AssignedNodeList[NodeIndex, 2] = node[2]
      NodeIndex = NodeIndex + 1
    End For
  End For
End For

```

特定の実施例では、scanSpatialは以下のロジック例を包含する：

【 0 1 6 0 】

【表 5 1】

```

For i = 0 to 2
  If PartWraps[i]
    End[i] = PartEnd[i]
  Else
    End[i] = PartEnd[i] - Try.baseSize[i] + 1
  End If
End For
zPlaneCnt = Try.baseSize[0] × Try.baseSize[1];
For i = PartStart[0] to End[0]
  newX = True
  For (n = PartStart[2] to PartEnd[2]
    zPlane[n] = 0
  End For
  For l = i to i + Try.baseSize[0]
    For n = PartStart[2] to PartEnd[2]
      l1 = l mod TorusSize[0]
      m1 = PartStart[1]
      m2 = (m1 + Try.baseSize[1]) mod TorusSize[1]
      If PartStart[1] + Try.baseSize[1] ≤ PartEnd[1]
        ZPlane[n] = zPlane[n] + FreeY[l1, m1, n] - FreeY[l1, m2, n]
      Else
        ZPlane[n] = zPlane[n] + FreeY[l1, m1, n]
      End If
    End For
  End For
  For j = PartStart[1] to End[1]
    if newX = False
      l1 = i mod TorusSize[0]
      l2 = (i + Try.baseSize[0]) mod TorusSize[0]
      m1 = (j - 1) mod TorusSize[1]
      if PartWraps[0] = False or i + Try.baseSize[0] ≤ PartEnd[0]
        For n = PartStart[2] to PartEnd[2]
          If i + Try.baseSize[0] ≤ PartEnd[0]
            zPlane[n] = zPlane[n] - (FreeX[l1, m1, n] - FreeX[l2, m1, n])
          Else
            zPlane[n] = zPlane[n] - FreeX[l1, m1, n]
          End If
        End For
      Else
        For n = PartStart[2] to PartEnd[2]
          zPlane[n] = zPlane[n] - (FreeX[l1, m1, n] + (FreeX[0, m1, n] -
            FreeX[l2, m1, n]))
        End For
      End If
      l1 = i mod TorusSize[0]
      l2 = (i + Try.baseSize[0]) mod TorusSize[0]
      m1 = (j + Try.baseSize[1]) mod TorusSize[1]
      If PartWraps[0] = False or i + Try.baseSize[0] ≤ PartEnd[0]
        For n = PartStart[2] to PartEnd[2]
          If i + Try.baseSize[0] ≤ PartEnd[0]
            ZPlane[n] = zPlane[n] + FreeX[l1, m1, n] - FreeX[l1, m2, n]
          Else
            ZPlane[n] = zPlane[n] + FreeX[l1, m1, n]
          End If
        End For
      End For
    End For
  End For

```

【 0 1 6 1 】

【 表 5 2 】

```

Else
  For n = PartStart[2] to PartEnd[2]
    ZPlane[n] = zPlane[n] + FreeX[l1,m1,n] + FreeX[0,m2,n] -
      FreeX[l1,m2,n]
  End For
End If
Else
  newX = False;
  k = PartStart[2];
  while k ≤ End[2]
    hit = True;
    For n = k; to k + Try.baseSize[2] - 1 while hit = True
      If zPlane[n mod TorusSize[2]] not equal zPlaneCnt
        hit = False;
      End If
    End For
    if hit = True
      Start[0] = i;
      Start[1] = j;
      Start[2] = k;
      For cNdx = 0 to try.NumConcat - 1 while hit = True
        For m = 0 to 2 while hit = True
          cStart[m] = Start[m] + Try.concatStartNode[cNdx, m]
          cEnd[m] = cStart[m] + Try.concatSize[cNdx, m] - 1;
          if (cEnd[m] ≥ TorusSize[m] && PartWraps[m] = False)
            hit = False;
          End For
        For l = cStart[0] to cEnd[0] while hit = True
          For m = cStart[1] to cEnd[1] while hit = True
            For n = cStart[2] to cEnd[2] while hit = True
              l1 = l mod TorusSize[0]
              m1 = m mod TorusSize[1]
              n1 = n mod TorusSize[2]
              If NodeInUse[l1,m1,n1] not equal NODE_NOT_IN_USE
                hit = False;
              End If
            End For
          End For
        End For
      If hit = True
        Return True;
      Else
        K = k + 1
      End If
    End If
  End If
End If
End For
End For
Return False

```

ctFunctionは以下のロジック例を包含する：

【0162】

【表53】

```

HighFit = NumberOfFits - 1
For i = 0 to 2
    HighStart[i] = PartStart[i]
End For
LowFit = -1
While True
    CurrentFit = LowFit + (HighFit - LowFit) / 2
    If scanCompact(NumberOfNodes, Fit[CurrentFit], HighStart) = True
        HighFit = CurrentFit
    Else
        LowFit = CurrentFit
    End If
    If HighFit = LowFit + 1
        Return
    End If
End While
Hit = False
For i = 0 to NumMaxDistances - 1 While Hit = False
    If HighFit ≥ MaxDistance[i,0]
        HigMaxDistance = i
        Hit = True
    End If
End For
Hit = True
For i = HighMaxDistance - 1 to 0 by -1
    StartFit = MaxDistance[i,0]
    If i = NumMaxDistance - 1
        EndFit = NumberOfFits - 1
    Else
        EndFit = MaxDistance[i+1,0] - 1
    End If
    Hit = False
    For j = StartFit to EndFit While Hit = False
        If scanCompact(NumberOfNodes, Fit[j], HighStart) = True
            HighFit = j
            HighMaxDistance = i
            Hit = True
        End If
    End For
End For
setCompactNodeInUse(Fit(HighFit), HighStart)

```

特定の実施例では、setCompactNodeInUseは以下のロジック例を包含する：

【0163】

【表 5 4】

```

node = 0
For i = 0 to 2
  if Start[i] ≥ TorusSize[i]
    Start[i] = Start[i] mod TorusSize[i]
    End[i] = Start[i] + Size[i] - 1
  End If
End For
If NumMapDiminsions = 1
  If MapDiminsion[0] = 0
    order[0] = 1
    order[1] = 2
    order[2] = 0
  Else If MapDiminsion[0] = 1
    order[0] = 0
    order[1] = 2
    order[2] = 1
  Else
    order[0] = 0
    order[1] = 1
    order[2] = 2
  End If
Else
  order[0] = 3 - MapDiminsion[0] - MapDiminsion[1]
  order[1] = MapDiminsion[0]
  order[2] = MapDiminsion[1]
End If
count = 0
For i = Start[order[0]] to end[order[0]] and count < RequestedNodes
  index[order[0]] = i mod TorusSize[order[0]]
  For j = Start[order[1]] to end[order[1]] and count < RequestedNodes
    index[order[1]] = j mod TorusSize[order[1]]
    For k = Start[order[2]] to end[order[2]] and count < RequestedNodes
      index[order[2]] = k mod TorusSize[order[2]]
      If NodeInUse[index[0], index[1], index[2]] = NODE_NOT_IN_USE
        NodeInUse[index[0], index[1], index[2]] = NODE_IN_USE
        AssignedNodeList[node, order[0]] = index[order[0]]
        AssignedNodeList[node, order[1]] = index[order[1]]
        AssignedNodeList[node, order[2]] = index[order[2]]
        node = node + 1
      End If
    End For
  End For
End For
End For

```

10

20

30

40

特定の実施例では、ScanCompactは以下のロジック例を包含する：

【0164】

【表 5 5】

```

For i = 0 to 2
  If PartWraps[i] = True
    end[i] = PartEnd[i]
  Else
    end[i] = PartEnd[i] - Start[i] + 1
  End If
  For i = PartStar[0] to end[0]
    newX = True
    For n = 0 to TorusSize[2]
      ZPlane[n] = 0
    End For
    for (l = i to i + size[0]
      for (n = pStart[2]; n ≤ pEnd[2]; n++)
        l1 = l mod TorusSize[0];
        m1 = PartStart[1]
        m2 = (PartStart[1] + size[1]) mod TorusSize[1]
        If PartStart[1] + size[1] ≤ PartEnd[1]
          ZPlane[n] = zPlane[n] + FreeY[l1, m1, n] - FreeY[l1, m2, n]
        Else
          ZPlane[n] = zPlane[n] + FreeY[l1, m1, n]
        End If
      End For
    End For
  End For
  For j = PartStart[1] to End[1]
    newY = True
    If newX = False
      l1 = i
      l2 = (i + size[0]) mod TorusSize[0]
      m1 = j - 1
      If PartWraps[0] = False or i + Start[0] ≤ PartEnd[0]
        For n = PartStart[2] to PartEnd[2]
          If i + size[0] ≤ PartEnd[0]
            ZPlane[n] = zPlane[n] - (FreeX[l1, m1, n] - FreeX[l2, m1, n])
          else
            zPlane[n] = zPlane[n] - FreeX[l1, m1, n]
          End If
        End For
      Else
        For n = PartStart[2] to PartEnd[2]
          zPlane[n] = zPlane[n] - (FreeX[l1, m1, n] + (FreeX[0, m1, n] - FreeX[l2, m1, n]))
        End For
      End If
      l1 = i
      l2 = (i + Start[0]) mod TorusSize[0]
      m1 = (j + size[1] - 1) mod TorusSize[1]
      If PartWraps[0] = False or i + Start[0] ≤ PartEnd[0]
        For n = PartStart[2] to PartEnd[2]
          If (i + Start[0] ≤ PartEnd[0])
            ZPlane[n] = zPlane[n] + (FreeX[l1, m1, n] - FreeX[l1, m2, n])
          Else
            ZPlane[n] = zPlane[n] + FreeX[l1, m1, n]
          End If
        End For
      End For
    End For
  End For

```

【0 1 6 5】

50

【表 5 6】

```

Else
  For  $n = \text{PartStart}[2]$  to  $\text{PartEnd}[2]$ 
     $\text{ZPlane}[n] = \text{zPlane}[n] + (\text{FreeX}[\text{I1}, m1, n] + (\text{FreeX}[0, m1, n] -$ 
       $\text{FreeX}[\text{I1}, m2, n]))$ 
  End For
End If
Else
  newX = False
End If
For  $k = \text{PartStart}[2]$  to  $\text{end}[2]$ 
  if newY = True
    newY = False
    count = 0;
    For  $n = k$  to  $k + \text{size}[2]$ 
      count = count +  $\text{zPlane}[n \bmod \text{TorusSize}[2]]$ 
    End For
  Else
    count = count -  $\text{zPlane}[k - 1]$ 
     $k1 = (k + \text{size}[2] - 1) \bmod \text{TorusSize}[2]$ 
     $\text{zPlane}[k1] = 0$ 
     $\text{I1} = i$ 
     $\text{I2} = (i + \text{size}[0]) \bmod \text{TorusSize}[0]$ 
    If  $\text{PartWraps}[0] = \text{False}$  or  $i + \text{size}[0] \leq \text{PartEnd}[0]$ 
      For  $m = j$  to  $j + \text{size}[1]$ 
         $m1 = m \bmod \text{TorusSize}[1]$ 
        If  $i + \text{size}[0] \leq \text{PartEnd}[0]$ 
           $\text{ZPlane}[k1] = \text{zPlane}[k1] + (\text{FreeX}[\text{I1}, m1, k1] -$ 
             $\text{FreeX}[\text{I2}, m1, k1])$ 
        Else
           $\text{ZPlane}[k1] = \text{zPlane}[k1] + \text{FreeX}[\text{I1}, m1, k1]$ 
        End For
      Else
        For  $m = j$  to  $j + \text{size}[1]$ 
           $\text{ZPlane}[k1] = \text{zPlane}[k1] + \text{FreeX}[\text{I1}, m1, k1] + (\text{FreeX}[0, m1, k1] -$ 
             $\text{FreeX}[\text{I2}, m1, k1])$ 
        End For
      End If
      count = count +  $\text{zPlane}[k1]$ 
    End If
    If count  $\geq \text{NumberOfNodes}$ 
      Start[0] =  $i$ 
      Start[1] =  $j$ 
      Start[2] =  $k$ 
      return True
    End If
  End For
End For
End For
return False

```

特定の実施例では、scheduleAnyは以下のロジックを包含する：

【0166】

【表 5 7】

```

Node = 0
Remainder = RequestedNodes
For m = 0 to NumFreeMeshes while Remainder > 0
  If FreeMesh[m].Rank[0] = 2
    iNdx = FreeMesh[m].Rank[2]
    jNdx = FreeMesh[m].Rank[1]
  Else If FreeMesh[m].Rank[1] = 2
    iNdx = FreeMesh[m].Rank[2]
    jNdx = FreeMesh[m].Rank[0]
  Else
    iNdx = FreeMesh[m].Rank[1]
    jNdx = FreeMesh[m].Rank[0]
  End If
  For i = FreeMesh[m].Start[iNdx] to FreeMesh[m].end[iNdx] while Remainder > 0
    For j = FreeMesh[m].Start[jNdx] to FreeMesh[m].end[jNdx] while Remainder > 0
      For k = FreeMesh[m].Start[2] to FreeMesh[m].end[2] while Remainder > 0
        i1 = i mod TorusSize[iNdx]
        j1 = j mod TorusSize[jNdx]
        k1 = k mod TorusSize[2]
        If iNdx = 0
          NodeInUse[j1, j1, k1] = NODE_IN_USE
        Else
          NodeInUse[j1, i1, k1] = NODE_IN_USE
        End If
        AssignedNodeList[Node].[iNdx] = i1
        AssignedNodeList[Node].[jNdx] = j1
        AssignedNodeList[Node, 2] = k1
        Node = Node + 1
      End For
    End For
  End For
End For

```

特定の実施例では、setMpiRankは以下のロジックを包含する：

【0167】

30

【表 5 8】

```

For node = 0 to RequestedNodes - 1
  to[0] = AssignedNodeList[node, 0]
  to[1] = AssignedNodeList[node, 1]
  to[2] = AssignedNodeList[node, 2]
  If NumMapDiminsions = 1
    to[MapDiminsion[0]] = AssignedNodeList[node, MapDiminsion[0]] / MapMod[0]
    to[3] = AssignedNodeList[node, MapDiminsion[0]] mod MapMod[0]
  Else
    to[MapDiminsion[0]] = AssignedNodeList[node, MapDiminsion[0]] / MapMod[0]
    to[MapDiminsion[1]] = AssignedNodeList[node, MapDiminsion[1]] / MapMod[1]
    to[3] = (AssignedNodeList[node, MapDiminsion[0]] mod MapMod[0]) × MapMod[1]
    +
    AssignedNodeList[node, MapDiminsion[1]] mod MapMod[1]
  End If
  hit = False
  for (node1 = 0 to NumFreeNodes - 1 while hit = False
    If to[0] = FreeNodeList[node1].coordinate[0] and
      to[1] = FreeNodeList[node1].coordinate[1] and
      to[2] = FreeNodeList[node1].coordinate[2] and
      to[3] = FreeNodeList[node1].coordinate[3]
      FreeNodeList[node1].mpiRank = node
      Hit = True
    End If
  End For
End For

```

特定の実施例では、スケジューラ 5 1 5 は、以下に定義する以下の構造例を用いてノード 1 1 5 をジョブ 1 5 0 に割り当てる。上記のように、クラスタ管理エンジン 1 3 0 は `FreeNode` 構造のリストをジョブ 1 5 0 とともにスケジューラ 5 1 5 に伝達する。このリストはスケジューリングに利用可能なノード 1 1 5 全てを含む。このリストでは、スイッチ・ベースの座標がリストにおける利用可能ノード 1 1 5 を識別する。スケジューラ 5 1 5 がジョブ 1 5 0 をスケジューリングする場合、スケジューラ 5 1 5 は戻す前に `mpiRank` を設定する。

【0 1 6 8】

【表 5 9】

```

Structure FreeNode
  integer    coordinate[4]
  integer    mpiRank
End Structure

```

特定の実施例では、スケジューラ 5 1 5 は `Fold Structure` を用いて、1 次元の空間的な要求と 2 次元の空間的な要求とをスケジューラ 5 1 5 がフォールディングする方法を記録する。

【0 1 6 9】

【表 6 0】

```

Structure Fold
  integer    foldLength
  integer    numFolds
  integer    remainder
  integer    foldDimension
  integer    fixDdimension
  integer    oneDimension
End Structure

```

10

特定の実施例では、スケジューラ 5 1 5 は T r y 構造を用いて、空間的なジョブ 1 5 0 をスケジューリングするのに用いるメッシュに関する情報を記憶する。T r y 構造は、ベース・メッシュと、最大 2 つまでの連結メッシュとに関する情報を含む。

【 0 1 7 0 】

【表 6 1】

```

Structure Try
  integer    baseSize[3]
  integer    numConcat
  integer    concatSize[2,3]
  integer    concatStartNode[2,3]
  integer    rMap[3]
  integer    irMap[3]
  integer    numFoldMaps
  integer    foldLength[2]
  integer    foldFrom[2]
  integer    foldTo[2]
  integer    foldFix[2]
End Structure

```

20

特定の実施例では、スケジューラ 5 1 5 は、F r e e M e s h 構造を用いて、スケジューリングに利用可能な、グリッド 1 1 0 におけるメッシュに関する情報を記憶させる。スケジューラ 5 1 5 は F r e e M e s h を用いて「何れかの」要求をスケジューリングする。

30

【 0 1 7 1 】

【表 6 2】

```

Structure FreeMesh
  integer    start[3]
  integer    end[3]
  integer    size[3]
  integer    rank[3]
  integer    numberOfNodes
End Structure

```

40

特定の実施例では、スケジューラ 5 1 5 は、以下に定義する以下の変数例を用いてノード 1 1 5 をジョブ 1 5 0 に割り当てる。

【 0 1 7 2 】

【表 6 3】

• RequestedNodes: ジョブ 150 に要求されるノード数	
• RequestType: ジョブ要求の種類: SPATIAL、COMPACT、又は ANY	
• OriginalSize[3]: RequestType = SPATIAL の場合、ジョブ 150 のサイズ	
• AggressiveFlag: ジョブ 150 にノード 115 を割り当てる目的でスケジューラ 515 に割り当てられる自由度を示す、ゼロと 1 との間の浮動小数	
• JobMap[3]: RequestType = SPATIAL の場合、スケジューラ 515 に、より適切な順序で OriginalSize の指数をマッピングする	10
• RequestedSize[3]: RequestType = SPATIAL の場合、スケジューラ 515 が JobMap を適用した後のジョブ 150 のサイズ	
• TorusSize[3]: CPU164 による、グリッド 110 のサイズ	
• NodesPerSwitch: スイッチ 166 毎のノード 115 の数	
• NumFreeNodes: スケジューリングに利用可能なノード 115 の数	
• FreeNodeList[NumFreeNodes]: スケジューラ 515 に転送される、スケジューリングに利用可能なノードのリスト	
• SpatialAllowed: 空間的なスケジューリングが可能である場合に True に設定する	20
• CompactAllowed: コンパクトなスケジューリングが可能である場合に True に設定する	
• AnyAllowed: 何れかのスケジューリングが可能である場合に True に設定する	
• TorusMap[3]: スイッチ・ベースのトーラスからの指数をスケジューラ 515 に、より適切な順序でマッピングする	
• InverseTorusMap[3]: TorusMap の逆数; クラスタ管理エンジン 130 への戻りの前に全ての出力ノード 115 に適用される。	
• NumMapDimesions: スイッチ・ベースのトーラスからノード・ベースのトーラスに移る場合に修正される次元の数; 考えられる値は 1 と 2 である。	
• MapDimensions[2]: スイッチ・ベースのトーラスからノード・ベースのトーラスに移る場合に修正される次元の指数	30
• MapMod[2]: スイッチ・ベースのトーラスからノード・ベースのトーラスに移る場合に用いられる乗数; 考えられる値は、NumMapDimesions = 1 の場合 MapMod[0] = 4 であり、NumMapDimesions = 2 の場合 MapMod[0] = 2 である。	
• PartSize[3]: パーティションのサイズ	
• PartStart[3]: パーティションの始点座標	
• PartEnd[3]: パーティションの終点座標	
• PartWraps[3]: パーティションが次元 i においてラッピングする場合、PartWraps[i] = True である。	40
• NodeInUse[TorusSize[0],TorusSize[1],TorusSize[2]]: NodeInUse[i,j,k]はノード 115 の状態を示す; 考えられる値は、NODE_IN_USE (別のジョブ 150 に割り当てられるノード 115)、NODE_NOT_IN_USE (利用可能なノード 115)、及び NODE_ON_HOLD (ノード 115 をジョブ 150 に割り当てる際に用いる一時的な状態) である。	

【0 1 7 3】

【表 6 4】

- **FreeY[TorusSize[0],TorusSize[1],TorusSize[2]]:** **FreeY[i,j,k]**は*i,j,k*と
*i,TorusSize[1]-1,k*とを含む、*i,j,k*から*i,TorusSize[1]-1,k*までの線におけるフリー・ノード 115 の数を示す。スキャン・ルーチンは **FreeY** を用いる。
- **FreeX[TorusSize[0],TorusSize[1],TorusSize[2]]:** **FreeX[i,j,k]**は*i,j,k*と
*i,TorusSize[1]-1,k*とを含む、*i,j,k*から*i,TorusSize[1]-1,k*までの線におけるフリー・ノード 115 の数を示す。スキャン・ルーチンは **FreeX** を用いる。
- **NumberOfTries:** 空間的な要求について構成される **Try** 構造の数
- **TryList[NumberOfTries]:** 空間的な要求についての **Try** 構造のリスト 10
- **NumberOfFits:** コンパクトな要求について構成されるメッシュの数
- **Fit[NumberOfFits,3]:** コンパクトな要求について構成されるメッシュのリスト
 - **Fit[i,0]** = *x* 次元におけるメッシュ *i* のサイズ
 - **Fit[i,1]** = *y* 次元におけるメッシュ *i* のサイズ
 - **Fit[i,2]** = *z* 次元におけるメッシュ *i* のサイズ
- **NumMaxDistances:** **Fit** における、一意の最大距離の数
- **MaxDistance[NumMaxDistances,2]:** **Fit** における、一意の最大距離のリスト。イ何れかの
 $0 \leq i < \text{NumMaxDistances}$ の場合、最大距離 = **MaxDistance[i,1]** の第 1 メッシュの **Fit** にお
ける、**MaxDistance[i,0]** = 指数 20
- **NumFreeMeshes:** グリッド 110 におけるフリー・メッシュの数。フリー・メッシュは、フ
リー・ノード 115 のみを含むメッシュである。
- **FreeMesh[NumFreeMeshes]:** **FreeMesh** 構造のアレイ
- **AssignedNodeList[RequestedNodes,3]:** MPI 評価順序における、ジョブ 115 に割り当てら
れるノード 115 のリスト

クラスタ管理エンジン 1 3 0 は更に、スケジューラ 5 1 5 などによって、効率的なチェ
ック・ポインティングを行うよう動作可能であり得る。再起動ダンプは通常、ディスクに
書き込まれるデータの 7 5 % を超えるデータを備える。この I / O は、多くの場合、処理 30
がプラットフォームの障害によって喪失されないように行われる。このことに基づいて、
ファイル・システムの I / O は、プロダクティブ I / O とディフェンシブ I / O との 2 つ
の部分に分離することが可能である。プロダクティブ I / O は、例えば、視覚化ダンプ、
主要物理変数の経時的なトレーシング他などの科学的方法を行ううえでユーザが要求する
、データの書き込みである。ディフェンシブ I / O は、かなりの期間にわたって実行され
る大規模シミュレーションを管理するよう行われる。よって、I / O 帯域幅の増加によ
って、チェック・ポインティングに係わる時間とリスクとを大いに削減する。

【0 1 7 4】

エンジン 1 3 0 に戻れば、局所メモリ 5 2 0 は、システム 1 0 0 の複数の特徴の論理的
な記述（すなわちデータ構造）を備える。局所メモリ 5 2 0 は、互換コードによって定義 40
、処理、又は取り出しを行うよう動作可能な何れかの物理データ記憶機構又は論理データ
記憶機構に記憶させ得る。例えば、局所メモリ 5 2 0 は、1 つ又は複数の拡張マーク付け
言語（XML）のテーブル又は文書を備え得る。種々の構成要素は、SQL ステートメン
ト若しくは SQL スクリプト、仮想記憶アクセス方法（VSAM）ファイル、フラット・
ファイル、2 進データ・ファイル、ビートリーブ（Btrieve）・ファイル、データ
ベース・ファイル、又はコンマで区切られた値（CSV）ファイルによって記述し得る。
各構成要素は変数、テーブル、又は何れかの別のデータ構造を備え得るということが分か
るものである。局所メモリ 5 2 0 は 1 つのサーバ 1 0 2 上に記憶されているか複数のサー
バ又はノードにわたって記憶されている複数のテーブル又はファイルを備える場合もある
。更に、エンジン 1 3 0 内部に存在しているように示されているが、局所メモリ 5 2 0 の 50

一部又は全部は、本願の開示の範囲から逸脱することなく内部のものか外部のものであり得る。

【0175】

図示する局所メモリ520は、物理リスト521、仮想リスト522、グループ・ファイル523、ポリシー・テーブル524、及びジョブ・キュー525を含む。しかし、図示していないが、局所メモリ520は、本願の開示の範囲から逸脱することなく、ジョブ・テーブルと監査ログとを含む別のデータ構造を含み得る。図示された構造に戻れば、物理リスト521はノード115に関する識別管理情報と物理的な管理情報とを記憶させるよう動作可能である。物理リスト521は、ノード115毎に少なくとも1つのレコードを含む多次元のデータ構造であり得る。例えば、物理レコードは、「ノード」、「利用可能性」、「プロセッサ利用率」、「メモリ利用率」、「温度」、「物理位置」、「アドレス」、「ブート画像」他などのフィールドを含み得る。各レコードは上記例示的フィールドを全く含まない場合があり、上記例示的フィールドの一部又は全部を含む場合もある。一実施例では、物理レコードは、例えば仮想リスト522などの別のテーブルへの外部キーを備え得る。

10

【0176】

仮想リスト522は、ノード115に関する論理管理情報又は仮想管理情報を記憶させるよう動作可能である。仮想リスト522は、ノード115毎に少なくとも1つのレコードを含む多次元データ構造であり得る。例えば、仮想レコードは、「ノード」、「利用可能性」、「ジョブ」、「仮想クラスタ」、「2次ノード」、「論理位置」、「互換性」他などのフィールドを含み得る。各レコードが、上記例示的フィールドを何ら含まない場合があり、上記例示的フィールドの一部又は全部を含む場合もあるということが分かるものである。一実施例では、仮想レコードは、例えばグループ・ファイル523などの別のテーブルへのリンクを含み得る。

20

【0177】

グループ・ファイル523は、アクセス・コントロール・リスト（すなわちACL）などの、ユーザ・グループ情報及びセキュリティ情報を記憶させるよう動作可能な1つ若しくは複数のテーブル又はレコードを備える。例えば、各グループ・レコードは、ユーザが利用可能なサービス、ノード115、又はジョブのリストを含み得る。各論理グループは、ジョブ150を投入するか、システム100の少なくとも一部分を管理することができる1つ又は複数のユーザの事業グループ若しくは事業単位、部署、プロジェクト、セキュリティ・グループ、又は何れかの別の集合と関連し得る。この情報に基づいて、クラスタ管理エンジン130は、ジョブ150を投入するユーザが有効なユーザであるかを判定し、判定結果が肯定の場合、ジョブを実行する上で最適なパラメータを判定し得る。更に、グループ・テーブル523は、各ユーザ・グループを、仮想クラスタ220と関連させるか、特定のグループの領域内部に存在するノードなどの1つ又は複数の物理ノード115と関連させる場合がある。これによって、各グループが、資源をめぐって競合することなく個々の処理空間を有することを可能にする。しかし、上記のように、仮想クラスタ220の形状とサイズは、動的である場合があり、必要性、時間、又は何れかの別のパラメータによって変動し得る。

30

40

【0178】

ポリシー・テーブル524は1つ又は複数のポリシーを含む。ポリシー・テーブル524とポリシー524を適宜、同義に用い得るということが分かるものである。ポリシー524は一般的に、ジョブ150及び／又は仮想クラスタ220に関する処理情報及び管理情報を記憶している。例えば、ポリシー524は、問題サイズ、問題実行時間、時間スロット、横取り、ノード115又は仮想クラスタ220のユーザ割り当て率などを含む何れかの数のパラメータ又は変数を含み得る。

【0179】

ジョブ・キュー525は、実行を待つ1つ又は複数の、ジョブ150のストリームを表す。一般的に、キュー525は、（ゼロを含む）何れかの数のジョブ150又は、それに

50

対する参照を記憶させる、バブル・アレイ、データベース・テーブル、又はポインタ・アレイなどの、何れかの適切なデータ構造を含む。グリッド110又は複数のキュー525と関連する1つのキュー525が存在し、各キュー525が、グリッド110内の一意の仮想クラスタ220の1つと関連付けられる場合がある。

【0180】

動作の一特徴では、クラスタ管理エンジン130は、計算を行い、情報を交換することによって問題を協調して解決するNのタスクから成るジョブ150を受信する。クラスタ管理エンジン130は、Nのノード115を割り当て、Nのタスクの各々を1つの特定のノード515に何れかの適切な手法を用いて割り当て、それによって問題が効率的に解決されることが可能になる。例えば、クラスタ管理エンジン130は、ユーザによって供給される、ジョブ・タスク配置ストラテジなどのジョブ・パラメータを利用し得る。とにかく、クラスタ管理エンジン130は、サーバ102のアーキテクチャを駆使しようとし、同様に、ユーザにとってより高速なターンアラウンドを備え、システム100の全体スループットを向上させると思われる。

10

【0181】

一実施例では、クラスタ管理エンジン130は更に、次の例のトポロジの何れかによって、ノード115を選択し、割り当てる。

【0182】

特定の、2次元(x, y)又は3次元(x, y, z)。ノード115を割り当て、タスクを特定の次元で順序付けして、それによって、効率的な、近隣同士の通信を確保する場合がある。特定のトポロジは、物理的な通信トポロジが問題トポロジに一致して、ジョブ150の協調タスクが近隣のタスクと頻繁に通信することを可能にするということが望ましい種々のジョブ150を管理する。例えば、2x2x2次元(2, 2, 2)における8つのタスクの要求は立方体に割り当てられることになる。最善のフィットの目的では、2Dの割り当ては、効率的な近隣同士の通信を確保する一方で3次元に「フォールディング」することが可能である。クラスタ管理エンジン130は、特定の次元の形状を何れかの向きで自由に割り当てることができる場合がある。例えば、2x2x8のボックスは、垂直方向又は水平方向に、利用可能な物理ノード内に割り当て得る。

20

【0183】

最善のフィットの立方体。クラスタ管理エンジン130は、Nのノード115を立方体の体積において割り当てる。このトポロジはジョブ150を効率的に処理し、それによって、協調タスクが、何れかの2つのノード115間での距離を最小にすることによって、何れかの別のタスクとデータを交換することを可能にする。

30

【0184】

最善のフィットの球体。クラスタ管理エンジン130は、Nのノード115を球体の体積において割り当てる。例えば、第1タスクが球体の中心ノード115に配置され、残りのタスクが中央ノード115を取り囲むノード115上に配置される場合がある。残りのタスクの配置順序は通常重要でないということが分かるものである。このトポロジは、第1タスクと全ての別のタスクとの間の距離を最小にする場合がある。このことは、タスク2乃至Nが第1タスクと通信するが、お互いには通信しない大規模の問題クラスを効率的に処理する。

40

【0185】

任意。クラスタ管理エンジン130がNのノード115を割り当てるが、ノード115が論理的に位置特定されるか物理的に位置特定されることに対する考慮の度合いは低くなっている。一実施例では、このトポロジはバックフィル目的でのグリッド110の積極的な利用を奨励し、別のジョブ150への影響はわずかなものである。

【0186】

上記トポロジと、付随する説明は図示する目的に過ぎず、実際に使用されるトポロジもそのようなトポロジを割り当てる手法も表さない場合がある。

【0187】

50

クラスタ管理エンジン130は、ジョブ150のパラメータ又はポリシー524のパラメータとして記憶される配置重みを利用し得る。一実施例では、配置重みは0と1との間のモディファイア値であり、この値は、要求されるタスク配置ストラテジ（又は処理配置ストラテジ）によってクラスタ管理エンジン130がどれだけ積極的にノード115を配置させようとするものであるかを表す。この例では、0の値は最適なストラテジ（又は次元）が考えられる場合のみノード115を配置させることを表し、1の値は、要求を処理するのに十分な、空き状態のノード115が存在するか、別のかたちで利用可能なノード115が存在する限り、直ちにノード115を配置させることを表す。通常、配置重みは、大規模のジョブ150の枯渇を妨げてHRCシステム100のジョブ・スループットを確保するために、資源予約などの管理ポリシー524をオーバーライドするものでない。

10

【0188】

上記図示と、付随する説明は、ノード115とジョブ150とを管理する論理的な手法を実施するエンジン130の例示的なモジュール図を備える。しかし、この図は例示的なものに過ぎず、システム100はこれらやその他のアルゴリズムを実施する論理構成要素の何れかの適切な組み合わせと配置とを用いることが想定されている。よって、これらのソフトウェア・モジュールは、ノード115とジョブ150とを効果的に管理する構成要素の何れかの適切な組み合わせと配置とを含み得る。しかし、種々の図示モジュールの処理は、適宜、合成される場合及び／又は分離される場合がある。

【0189】

図11はインタフェース104の例を示す。インタフェース104は、ネットワーク106とHPCサーバ102との間のインタフェースを備える、ハードウェア構成部分、ソフトウェア構成部分、若しくは組み込み論理構成部分、又は2つ以上のそのような構成部分の組み合わせを含む。特定の実施例では、インタフェース104は、インスタネーション・マネージャ534とインスタネーション・データ536とを含む。インスタネーション・マネージャ534は、クライアント120からの接続要求に応じてノード115でのホストを動的にインスタンス化する、ハードウェア構成部分、ソフトウェア構成部分、若しくは組み込み論理構成部分、又は2つ以上のそのような構成部分を含む。特定の実施例では、クライアント120からの接続要求は、伝送制御プロトコル（TCP）接続要求である。TCP接続要求を説明するが、本発明は何れかの適切な接続要求を想定している。制約ではなく例として、クライアント120からの接続要求は、ユーザ・データグラム・プロトコル（UDP）又は別のIPベースの接続要求であり得る。

20

30

【0190】

特定の実施例では、HPCサーバ102の内部にあるホスト名とポート番号とに対して、HPCサーバ102に対して外部に共用宣言されるホスト名とポート番号をマッピングする、ルータ又は、ルータに対するインタフェースとして機能する。インスタネーション・マネージャ534は、特定のニーズに応じて、クライアント120からの接続要求に応答して1つ又は複数のノード115での1つ又は複数のホストを動的にインスタンス化するよう、（例えば、物理マネージャ505と仮想マネージャ510との何れか又は両方などの、）クラスタ管理エンジン130の1つ又は複数の構成部分と相互作用し得る。

【0191】

40

インスタネーション・データ536は、クライアント120からの接続要求に応じてノード115でのホストをインスタンス化するデータを含む。特定の実施例では、インスタネーション・データ536は、HPCサーバ102に対して外部に共用宣言される、サービスの1つ又は複数のリストを含む。サービスへの言及は、適宜、アプリケーションを包含し、逆も、適宜、あてはまる。HPCサーバ102に対して外部に共用宣言される、サービス・リストへの言及は、適宜、ルーティング・テーブルを包含し、逆も、適宜、あてはまる。特定の実施例では、インスタネーション・マネージャ534は、そのようなルーティング・テーブルを設定し、維持する。特定の実施例では、HPCサーバ102に対して外部に共用宣言されるサービス・リストにおけるエントリは：（1）サービス；（2）サービスに相当する、HPCサーバ102に対して外部に共用宣言されるホ

50

スト名とポート番号；及び（３）インスタンス化される場合、サービスを備えるホストに相当する、H P Cサーバ１０２に対して内部にあるホスト名とポート番号；を規定する。エントリは、ホストを利用可能にすることとする時点、ホストのインスタンス化を行うこととする時点、及びホストを利用不能とすることとする時点を支配する規則と条件との何れか又は両方を規定する場合もある。制約ではなく例として、ホストはウェブ・サーバを備え得る。インスタンシエーション・マネージャ５３４が営業時間中にウェブ・サーバに相当するH T T PポートでのH T T P要求を何ら受信しない場合、ホストは営業時間中に非インスタンス化状態にとどまっている場合があり、インスタンス化される場合にホストが用いることになる１つ又は複数の（グリッド１１０におけるノード１１５などの）１つ又は複数の資源は、別のホストとサービスとの何れか又は両方に利用可能であり得る。クライアント１２０の場所にいるユーザがウェブ・ブラウザを用いてウェブ・サーバを営業時間中にアクセスする場合、インスタンシエーション・マネージャ５３４はホストをインスタンス化してウェブ・サーバをクライアント１２０に備え得る。クライアント１２０の場所にいるユーザが、ウェブ・ブラウザを用いて営業時間外にウェブ・サーバをアクセスする場合、インスタンシエーション・マネージャ５３４はウェブ・サーバに相当するH T T Pポートをブロック化して、ウェブ・サーバをクライアント１２０に備えることからホストを妨げる。

【０１９２】

特定の実施例では、サービスを備えるよう、インスタンシエーション・データ５３６はノード１１５でホストをインスタンス化する１つ又は複数のブート画像を含む。特定の実施例では、サービスを備えるよう、インスターシエーション・データ５３６はノード１１５でホストをインスタンス化する１つ又は複数のファイル・システムを含む。特定の実施例では、インスタンシエーション・データ５３６は更に、サービスを備えるよう、ノード１１５でのホストをインスタンス化する１つ又は複数のＯＳ構成ファイルも含む。制約ではなく例として、H P Cサーバ１０２に対して外部に共用宣言されるポート番号を規定するクライアント１２０からの接続要求を受信するインスタンシエーション・マネージャ５３４に応じて、インスタンシエーション・マネージャ５３４は、ノード１１５でのサービス用のホストを初期化するよう、サービス用の、ブート画像と、１つ又は複数のファイル・システムとを用いて、グリッド１１０における利用可能なノード１１５をブートし得る。インスタンシエーション・マネージャ５３４は、I Pトラフィックをクライアント１２０からノード１１５までルーティングするよう、局所ルーティング・テーブルと、１つ又は複数のＯＳ構成ファイルとの１つ又は複数を更新する場合もある。特定の実施例では、クライアント１２０からの接続要求に応じるH P Cサーバ１０２に関連した時間要求を軽減するよう、インスタンシエーション・マネージャ５３４はターゲット・ホストのI P／M A Cアドレスをスプーフィングし、ターゲット・ホストのためにT C P／I P接続シーケンスを開始する。クライアント１２０とインスタンシエーション・マネージャ５３４との間のT C P／I P接続シーケンスは、ターゲット・ホストがブートしている間に行われる。T C P／I P接続シーケンスを説明したが、本発明は何れかの適切な接続シーケンスを想定している。制約ではなく例として、インスタンシエーション・マネージャ５３４は、ターゲット・ホストのためにユーザ・データグラム・プロトコル（U D P）オーバI P（U D P／I P）又は別のI Pベースの接続シーケンスを開始することが可能である。

【０１９３】

特定の実施例では、インスタンシエーション・マネージャ５３４は、H P Cサーバ１０２での各ホストがアクティブ状態にあるか、非アクティブ状態にあるかを追跡する。特定の実施例では、インスタンシエーション・マネージャ５３４は、H P Cサーバ１０２での各ホストがアクティブ状態にあるか、非アクティブ状態にあるかの制御も行う。特定の実施例では、インスタンシエーション・マネージャ５３４はサービスが利用可能であることとするか否かを判定し得る。サービスはもはや利用可能であることとするものでないということをインスタンシエーション・マネージャ５３４が判定する場合、インスタンシエーション・マネージャ５３４は、サービスを備えるようインスタンシエーション・マネージャ

10

20

30

40

50

ャ 5 3 4 がホストをインスタンス化した 1 つ又は複数のノード 1 1 5 をシャットダウンするか、アイドル状態にするか、別の方法で利用不能にし、それに応じて 1 つ又は複数のルーティング・テーブルを更新する。

【0194】

図 1 2 は、本願の開示の一実施例による、ジョブ投入を動的に処理する方法 6 0 0 の例を示す流れ図である。一般的に、図 1 2 は、バッチ・ジョブ投入を受信し、ジョブ・パラメータと、関連ポリシー 5 2 4 とに基づいてノード 1 1 5 をジョブ空間 2 3 0 に動的に割り当て、割り当て空間を用いてジョブ 1 5 0 を実行する方法 6 0 0 を表す。以下の説明は、方法 6 0 0 を行ううえでのクラスタ管理モジュール 1 3 0 の処理に焦点を当てる。しかし、機能が適切な状態のままである限り、記載機能の一部又は全てを実施する論理構成要素の何れかの適切な組み合わせと配置とを用いることを想定している。

10

【0195】

モジュール 6 0 0 は、工程 6 0 5 から始まり、H P C サーバ 1 0 2 はジョブ投入 1 5 0 をユーザから受信する。上記のように、一実施例では、ユーザはジョブ 1 5 0 を、クライアント 1 2 0 を用いて投入し得る。別の実施例では、ユーザは、H P C サーバ 1 0 2 を用いてジョブ 1 5 0 を直接投入し得る。次に、工程 6 1 0 で、クラスタ管理エンジン 1 3 0 は、ユーザに基づいてグループ 5 2 3 を選択する。ユーザが検証されると、クラスタ管理エンジン 1 3 0 は、工程 6 1 5 で、ユーザをグループ・アクセス制御リスト (A C L) と比較する。しかし、クラスタ管理エンジン 1 3 0 が何れかの適切なセキュリティ手法を用いてユーザを検証する場合があるということが分かるものである。判定グループ 5 2 3 に基づいて、クラスタ管理エンジン 1 3 0 は、ユーザが要求サービスに対するアクセスを有するかを判定する。要求されるサービス及びホスト名に基づいて、クラスタ管理エンジン 1 3 0 は、工程 6 2 0 で、仮想クラスタ 2 2 0 を選択する。通常、仮想クラスタ 2 2 0 は、ジョブ 1 5 0 の投入前に識別され、割り当てられる。しかし、仮想クラスタ 2 2 0 が確立されていない場合、クラスタ管理エンジン 1 3 0 は上記手法のうちの何れかを用いて仮想クラスタ 2 2 0 を自動的に割り当てる。次に、工程 6 2 5 で、クラスタ管理エンジン 1 3 0 は、ジョブ 1 3 0 の投入に基づいてポリシー 5 2 4 を取り出す。一実施例では、クラスタ管理エンジン 1 3 0 は、ユーザ、ジョブ 1 5 0、又は何れかの別の適切な基準に関連した適切なポリシー 5 2 4 を判定し得る。クラスタ管理エンジン 1 3 0 は更に、工程 6 3 0 で、ジョブ 1 5 0 のディメンションを判定するか別の方法で計算する。適切なディメンションが長さ、幅、高さ、又は何れかの別の適切なパラメータ若しくは特性を含み得るということは分かるものである。上記のように、これらのディメンションは仮想クラスタ 2 2 0 内の適切なジョブ空間 2 3 0 (又はノード 1 1 5 の部分集合) を判定するのに用いられる。初期パラメータが確立された後、クラスタ管理 1 3 0 は、工程 6 3 5 乃至 6 6 5 で、H P C サーバ 1 0 2 上でジョブ 1 5 0 を実行しようとする。

20

30

【0196】

判定工程 6 3 5 では、クラスタ管理エンジン 1 3 0 は、既に確立されているパラメータを用いて、所望のジョブ空間 2 3 0 を割り当てるのに十分利用可能なノードが存在しているかを判定する。ノード 1 1 5 が十分存在していない場合には、工程 6 4 0 で、クラスタ管理エンジン 1 3 0 が仮想クラスタ 2 2 0 におけるノード 1 1 5 の最も早く利用可能な部分集合 2 3 0 を判定する。更に、クラスタ管理エンジン 1 3 0 は、工程 6 4 5 で、部分集合 2 3 0 が利用可能であるまでジョブ・キュー 1 2 5 にジョブ 1 5 0 を追加する。処理は次に判定工程 6 3 5 に戻る。利用可能なノード 1 1 5 が十分になると、クラスタ管理エンジン 1 3 0 は、工程 6 5 0 で、利用可能なノード 1 1 5 から最適な部分集合 2 3 0 を動的に判定する。最適な部分集合 2 3 0 は、最高速処理時間、最高の信頼度のノード 1 1 5、物理位置若しくは仮想位置、又は最初に利用可能なノード 1 1 5 を含む何れかの適切な基準を用いて判定し得るということが分かるものである。工程 6 5 5 では、クラスタ管理エンジン 1 3 0 は判定部分集合 2 3 0 を選択仮想クラスタ 2 2 0 から選択する。次に、工程 6 6 0 で、クラスタ管理エンジン 1 3 0 は、選択部分集合 2 3 0 を用いてジョブ 1 5 0 に選択ノード 1 1 5 を割り当てる。一実施例によれば、クラスタ管理エンジン 1 3 0 は、仮

40

50

想ノード・リスト522におけるノード115の状態を「非割り当て」から「割り当て」に変更し得る。部分集合230が適切に割り当てられると、クラスタ管理エンジン130は工程665で、ジョブ・パラメータ、取り出したポリシー524、及び何れかの別の適切なパラメータに基づいて割り当て空間を用いてジョブ150を実行する。何れかの適切な時点では、クラスタ管理エンジン130は、ジョブ結果160をユーザに通信するか別の方法で提示する場合がある。例えば、結果160は、フォーマット化され、GUI126を介してユーザに提示される場合がある。

【0197】

図13は、本願の開示の一実施例による、グリッド110における仮想クラスタ220の動的バックフィルを行う方法700の例を示す流れ図である。概略を述べれば、方法700は、仮想クラスタ220において利用可能な空間を判定する工程、その空間と互換の最適ジョブ150を判定する工程、及びその利用可能空間において判定ジョブ150を実行する工程を表す。以下の説明は、この方法を行ううえでのクラスタ管理モジュール130の処理に焦点を当てることとする。しかし、上記流れ図と同様に、システム100は、上記機能の一部又は全部を実施する論理構成要素の何れかの適切な組み合わせと配置とを用いることが想定されている。

【0198】

方法700は工程305から開始し、クラスタ管理エンジン130は、ジョブ・キュー525をソートする。図示する実施例では、クラスタ管理エンジン130は、キュー525に記憶されているジョブ150の優先度に基づいてキュー525をソートする。しかし、クラスタ管理エンジン130が、適切なジョブ150又は最適なジョブ150が実行されることになるように何れかの適切な特性を用いてキュー525をソートし得るということは分かるものである。次に、工程710で、クラスタ管理エンジン130は、仮想クラスタ200のうちの1つにおける利用可能ノード115の数を判定する。当然、クラスタ管理エンジン130がグリッド110又は、仮想クラスタ220のうちの何れか1つ又は複数のものにおける、利用可能なノード115の数を判定する場合もある。工程715では、クラスタ管理エンジン130は第1ジョブ150をソートされるジョブ・キュー525から選択する。次に、720で、クラスタ管理エンジン130は、選択ジョブ150の最適な形状（又は別のディメンション）を動的に判定する。選択ジョブ150の最適な形状又はディメンションが判定されると、クラスタ・エンジン130が、工程725乃至745で、適切な仮想クラスタ220においてジョブ150のバックフィルを行うかを判定する。

【0199】

判定工程725では、選択ジョブ150に十分に利用可能なノード115が存在するかをクラスタ管理エンジン130が判定する。利用可能なノード115が十分に存在する場合、工程730で、クラスタ管理エンジン130が何れかの適切な手法を用いて、選択ジョブ150にノード115を動的に割り当てる。例えば、クラスタ管理エンジン130は、図6に表す手法を用い得る。次に、工程735では、クラスタ管理エンジン130は、仮想クラスタ220における利用可能なノードの数を再計算する。工程740では、クラスタ管理エンジン130は割り当てノード115のジョブ150を実行する。ジョブ150が実行されることによって（、又は、選択ジョブ150に十分なノード115が存在しなかった場合）、クラスタ管理エンジン130は工程745で、ソートされるジョブ・キュー525における次のジョブ150を選択し、処理は工程720に戻る。ループとして図示したが、クラスタ管理エンジン130が、何れかの適切な時点で方法700に示す手法を開始し、実行し、終結する場合があるということが分かるものである。

【0200】

図14は、本願の開示の一実施例による、グリッド110におけるノード115の障害を動的に管理する方法800の例を示す流れ図である。概略を述べれば、方法800は、ノード115に障害が起きたということを判定する工程、ジョブの回復と管理とを自動的に行う工程、及び障害ノード115を2次ノード115と置き換える工程を表す。以下の

説明は、この方法を行ううえでのクラスタ管理モジュール130の処理に焦点を当てることとする。しかし、上記流れ図のように、システム100は、記載機能の一部又は全部を実施する論理構成要素の何れかの適切な組み合わせと配置とを用いることを想定している。

【0201】

方法800は工程805から始まり、クラスタ管理エンジン130は、ノード115に障害が起こったということを判定する。上記のように、クラスタ管理エンジン130は、何れかの適切な手法を用いて、ノード115に障害が起こったということを判定し得る。例えば、クラスタ管理エンジン130は、種々の時点でノード115（又はエージェント132）を取り出す場合があり、ノード115から応答がないことに基づいてノード115に障害が起こったということを判定し得る。別の例では、ノード115上に存在しているエージェント132は「ハートビート」を通信する場合があり、この「ハートビート」がないことはノード115の障害を示し得る。次に、工程810で、クラスタ管理エンジン130は障害ノード115を仮想クラスタ220から取り除く。一実施例では、クラスタ管理エンジン130は仮想リスト522におけるノード115の状態を「割り当て」から「障害」に変更し得る。クラスタ管理エンジン130は次に、判定工程815で、ジョブ150が障害ノード115に関連しているかを判定する。ノード115と関連しているジョブ150が何らない場合には、処理は終了する。上記のように、処理が終了する前に、クラスタ管理エンジン130は、管理者にエラー・メッセージを通信してもよく、置き換えノード115を自動的に判定してもよく、何れかの別の処理を行ってもよい。障害ノード115と関連したジョブ150が存在する場合、クラスタ管理エンジン130は、工程820で、ジョブ150と関連した別のノード115を判定する。次に、工程825では、クラスタ管理エンジン130は全ての適切なノード115のジョブ150を強制終了する。例えば、クラスタ管理エンジン130は、ジョブ150を終了するよう、強制終了ジョブ・コマンドを実行するか、何れかの別の適切な手法を用いる場合がある。次に、工程830では、クラスタ管理エンジン130は、仮想リスト522を用いてノード115の割り当て解除を行う。例えば、クラスタ管理エンジン130は、仮想リスト522におけるノード115の状態を「割り当て」から「利用可能」に変更し得る。ジョブが終結され、全ての適切なノード115の割り当て解除が行われると、工程835乃至850で、クラスタ管理エンジン130は利用可能ノード115を用いてジョブ150を再実行しようとする。

【0202】

工程835では、クラスタ管理エンジン130は、ポリシー524と、工程825での強制終了ジョブ150のパラメータとを取り出す。クラスタ管理エンジン130は更に、工程840で、取り出したポリシー524及びジョブ・パラメータに基づいて、仮想クラスタ220におけるノード115の最適部分集合230を判定する。ノード115の部分集合230が判定されると、工程845で、クラスタ管理エンジン130はノード115の部分集合230を動的に割り当てる。例えば、クラスタ管理エンジン130は、仮想リスト522におけるノード115のステータスを「非割り当て」から「割り当て」に変更し得る。このノード115の部分集合が、ジョブ150が実行していた元のノード部分集合とは異なり得るということは分かるものである。例えば、クラスタ管理エンジン130は、この実行をプロンプトしたノード障害が理由で別のノード部分集合が最適であるということを判定し得る。別の例では、2次ノード115が障害ノード115を置き換えるよう動作可能であり、新たな部分集合230が旧ジョブ空間230にかなり同様であるということを判定した場合がある。割り当て部分集合230が判定され、割り当てられると、工程850で、クラスタ管理エンジン130はジョブ150を実行する。

【0203】

上記流れ図と、付随する説明は、例示的方法600、700及び800を示す。要約すれば、システム100は、これらやその他のタスクを行うのに適切な何れかの手法を用いることを想定している。よって、この流れ図における工程の多くは、表しているものと同

10

20

30

40

50

時に行われる場合及び／又は表しているものとは異なる順序で行われる場合がある。更に、システム１００は、当該方法が適切な状態のままである限り、工程を追加して方法を用いるか、工程を少なくして方法を用いる場合、及び／又は別の工程を備えている方法を用いる場合がある。

【０２０４】

図１５は、ＨＰＣシステム１００におけるオン・デマンド・インスタンスーションの方法例を示す。方法は工程９００から始まり、インタフェース１０４は、ＨＰＣサーバ１０２に対して外部に共用宣言される、ポート番号とホスト名とを規定するクライアント１２０からの接続要求を受信する。工程９０２では、接続要求に応じて、インスタンスーション・マネージャ５３４は、ＨＰＣサーバ１０２に対して外部に共用宣言されるサービス・リストを備えるインスタンスーション・データ５３６をアクセスする。工程９０４では、インスタンスーション・マネージャ５３４は、接続要求において規定されるポート番号とホスト名とに相当するサービスを識別するよう、サービス・リストを用いる。工程９０６では、インスタンスーション・マネージャ５３４は、識別サービスがクライアント１２０に利用可能であるか否かをサービス・リストによって判定する。上記のように、識別サービスがクライアント１２０に利用可能であるか否かは、接続要求に関連した時間、クライアント１２０でのユーザの識別情報、又は接続要求の別の特徴によって変わってくる場合がある。工程９０６では、識別サービスがクライアント１２０に利用可能な場合、当該方法は工程９０８に進む。工程９０８では、インスタンスーション・マネージャ５３４は、グリッド１１０における１つ又は複数のノード１１５でのホストをインスタンス化して、クライアント１２０に識別サービスを備えるよう、識別サービスに相当するＨＰＣサーバ１０２の内部にある、ホスト名とポート番号とを示すインスタンスーション・データ５３６を用いる。上記のように、インスタンスーション・マネージャ５３４は、ノード１１５でのホストをインスタンス化するよう、ブート画像、ファイル・システム、及びＯＳ構成を含むインスタンスーション・データ５３６を用いる場合もあり、この時点で该方法が終了する。工程９０６では、識別サービスがクライアント１２０に利用不能である場合、当該方法は工程９１０に進む。工程９１０では、インスタンスーション・マネージャ５３４は、接続要求において規定されるポートをブロック化して、識別サービスをクライアント１２０がアクセスすることを妨げ、この時点で该方法が終了する。図１５に示す当該方法における特定の工程を、特定の順序で行われるものとして図示し、説明したが、図１５に示す当該方法における何れかの適切な工程は何れかの適切な順序で行い得る。

【０２０５】

本願の開示は、特定の実施例と、概括的に関連している方法とによって表したが、これらの実施例と方法との改変と置換が当業者に明らかとなる。よって、例示的实施例の上記記載は本願の開示を規定するものでも制約するものでもない。別の変更、置換及び改変も、本願の開示の趣旨及び範囲から逸脱することなく考えられる。

【図面の簡単な説明】

【０２０６】

【図１】本願開示の一実施例による、高性能計算システムの例を示す図である。

【図２】図１に示すＨＰＣシステムにおけるノードの例を示す図である。

【図３】ノードにおける中央処理装置（ＣＰＵ）の例を示す図である。

【図４】ノード対の例を示す図である。

【図５Ａ】図１のシステムにおけるグリッドとその利用との種々の実施例を示す図である。

【図５Ｂ】図１のシステムにおけるグリッドとその利用との種々の実施例を示す別の図である。

【図５Ｃ】図１のシステムにおけるグリッドとその利用との種々の実施例を示す更に別の図である。

【図５Ｄ】図１のシステムにおけるグリッドとその利用との種々の実施例を示す更に別の

図である。

【図 6 A】図 1 のシステムによる、グラフィカル・ユーザ・インタフェースの種々の実施例を示す図である。

【図 6 B】図 1 のシステムによる、グラフィカル・ユーザ・インタフェースの種々の実施例を示す別の図である。

【図 7】図 1 におけるシステムによる、クラスタ管理ソフトウェアの一実施例を示す図である。

【図 8】y 次元にフォールディングされる 1 次元の要求の例を示す図である。

【図 9】y 軸を内部ループとして用いて構成される 2 つのフリー・メッシュを示す図である。

10

【図 10】x 軸を内部ループとして用いて構成される 2 つのフリー・メッシュを示す図である。

【図 11】図 1 に示す H P C システムのインタフェース例を示す図である。

【図 12】図 1 の高性能計算システムによる、バッチ・ジョブを投入する方法を示す流れ図である。

【図 13】図 1 の高性能計算システムによる、グリッドの動的なバックフィルを行う方法を示す流れ図である。

【図 14】図 1 の高性能計算システムによる、ノード障害を動的に管理する方法を示す流れ図である。

【図 15】図 1 に示す H P C システムによる、オン・デマンド・インスタンスーションの方法例を示す図である。

20

【符号の説明】

【0207】

- 100 システム
- 102 H P C サーバ
- 104 インタフェース
- 105 管理ノード
- 106 ネットワーク
- 108 リンク
- 110 グリッド
- 115 ノード
- 120 局所クライアント
- 126 G U I
- 130 クラスタ管理エンジン
- 132 エージェント
- 140 ディスク・ファーム
- 142 ストレージ・クライアント
- 150 ジョブ
- 164 C P U
- 164 a C P U
- 164 b C P U
- 164 c C P U
- 164 d C P U
- 166 スイッチ
- 166 a スイッチ
- 166 b スイッチ
- 168 a リンク
- 168 b リンク
- 168 c リンク
- 168 d リンク

30

40

50

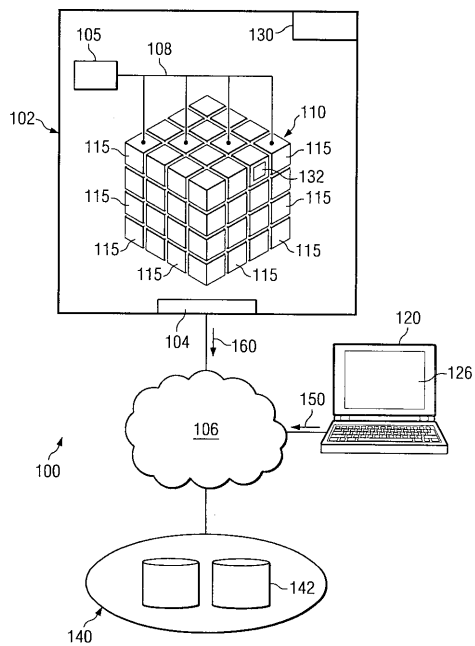
1 6 8 e	リンク	
1 6 8 f	リンク	
1 6 8 g	リンク	
1 6 8 h	リンク	
1 7 0	リンク	
1 7 2 a	リンク	
1 7 2 b	リンク	
1 7 2 c	リンク	
1 7 2 d	リンク	
1 7 2 e	リンク	10
1 7 2 f	リンク	
1 7 4	プロセッサ	
1 7 6	M C H	
1 7 8	メモリ装置	
1 8 0	H C A	
1 8 2	バス	
1 8 6	ノード対	
2 1 6	ノード・ライン	
2 2 0	仮想クラスタ	
2 3 0 a	ジョブ空間	20
2 3 0 b	ジョブ空間	
4 0 0 a	G U I	
4 0 0 b	G U I	
5 0 5	物理マネージャ	
5 1 0	仮想マネージャ	
5 1 5	ジョブ・スケジューラ	
5 2 0	局所メモリ	
5 2 1	物理リスト	
5 2 2	仮想リスト	
5 2 3	グループ・ファイル	30
5 2 4	ポリシー・テーブル	
5 2 5	ジョブ・キュー	
5 3 4	インスタンシエーション・マネージャ	
5 3 6	インスタンシエーション・データ	
6 0 0	方法	
6 0 5	工程	
6 1 0	工程	
6 1 5	工程	
6 2 0	工程	
6 2 5	工程	40
6 3 0	工程	
6 3 5	判定工程	
6 4 0	工程	
6 4 5	工程	
6 5 0	工程	
6 5 5	工程	
6 6 0	工程	
6 6 5	工程	
7 0 0	方法	
7 0 5	工程	50

7 1 0 工程
 7 1 5 工程
 7 2 0 工程
 7 2 5 判定工程
 7 3 0 工程
 7 3 5 工程
 7 4 0 工程
 7 4 5 工程
 8 0 0 方法
 8 0 5 工程
 8 1 0 工程
 8 1 5 判定工程
 8 2 0 工程
 8 2 5 工程
 8 3 0 工程
 8 3 5 工程
 8 4 0 工程
 8 4 5 工程
 8 5 0 工程
 9 0 0 工程
 9 0 2 工程
 9 0 4 工程
 9 0 6 判定工程
 9 0 8 工程
 9 1 0 工程

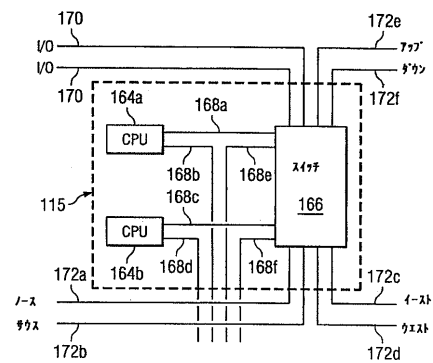
10

20

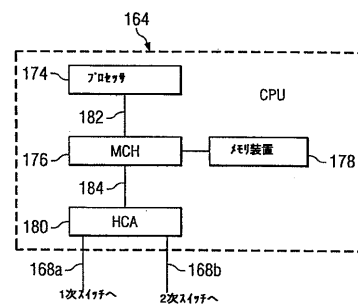
【図 1】



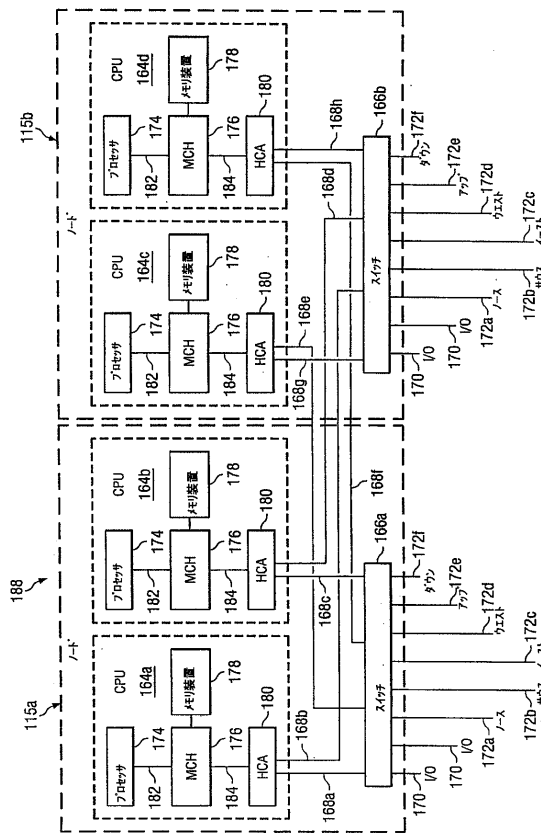
【図 2】



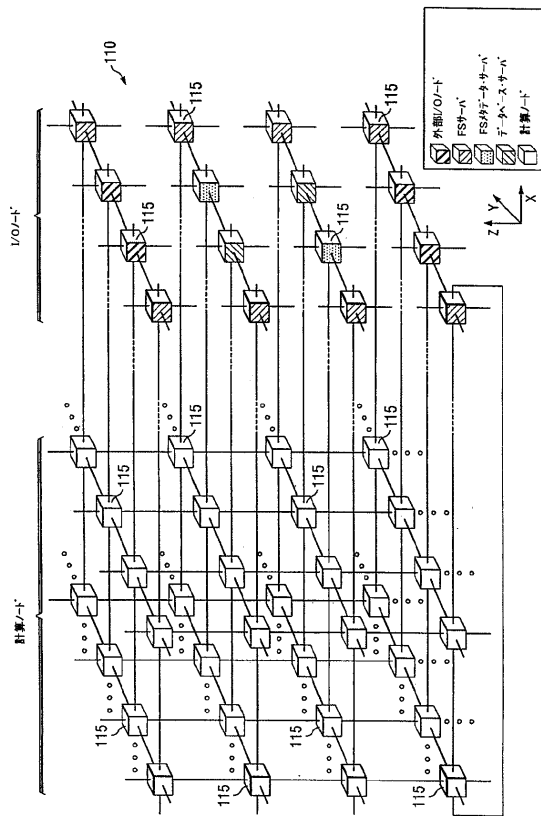
【図 3】



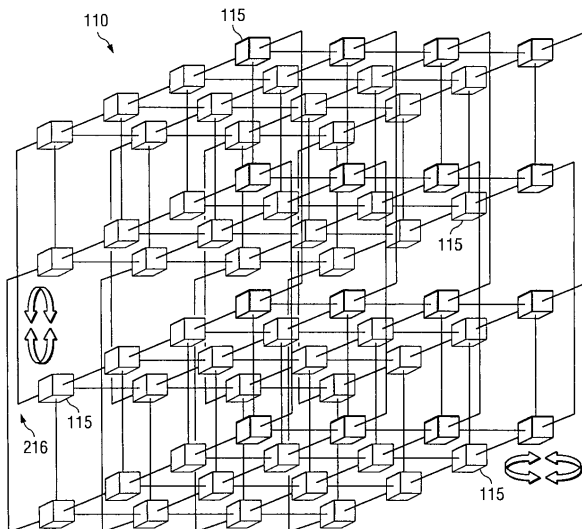
【図 4】



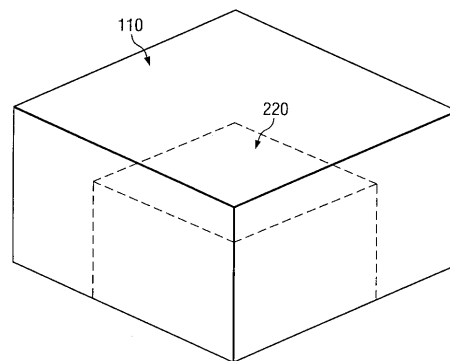
【図 5 A】



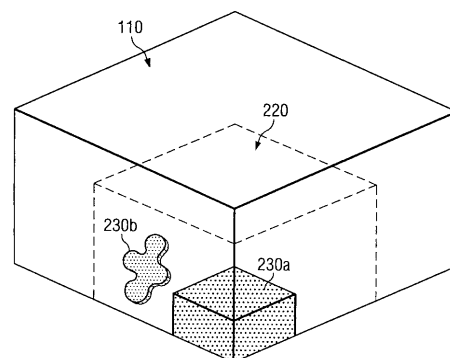
【図 5 B】



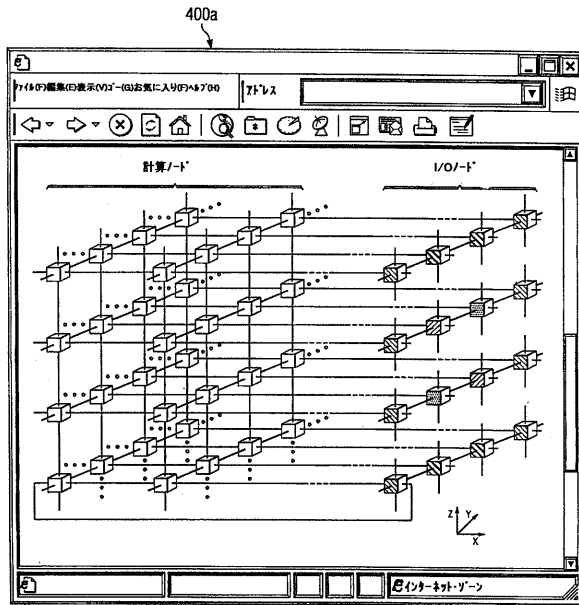
【図 5 C】



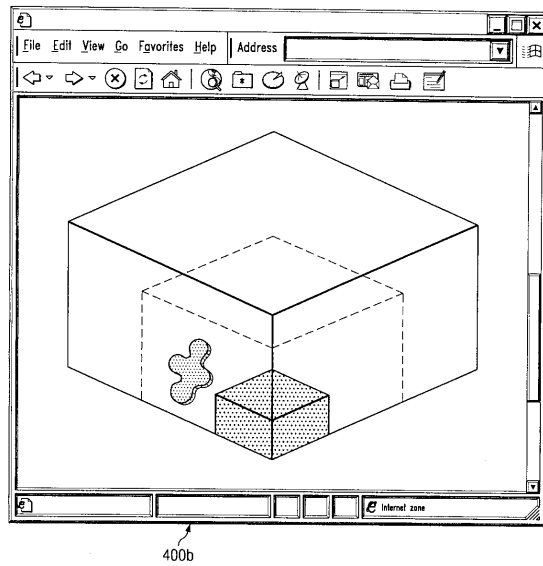
【図 5 D】



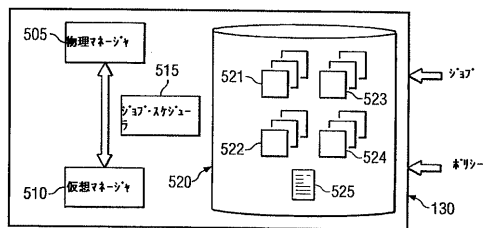
【図 6 A】



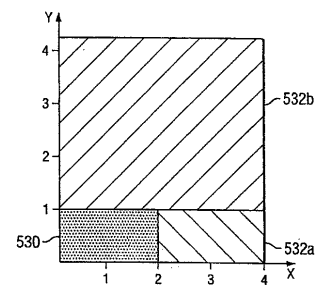
【図 6 B】



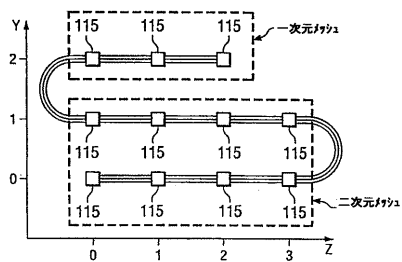
【図 7】



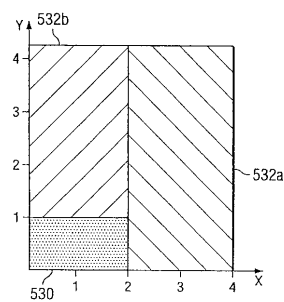
【図 9】



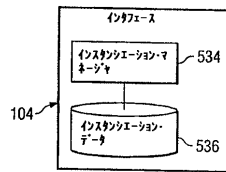
【図 8】



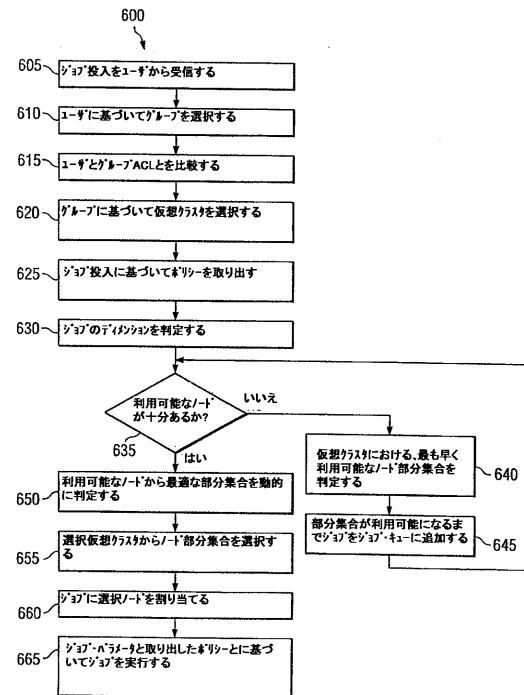
【図 10】



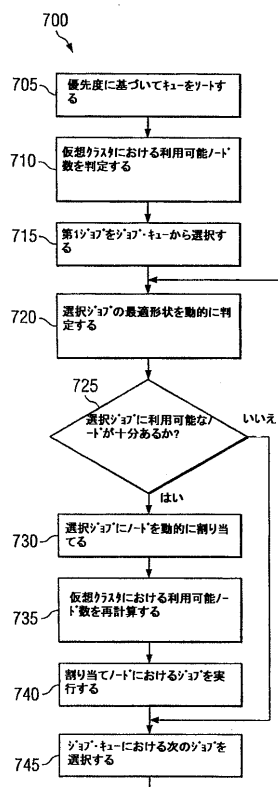
【図 1 1】



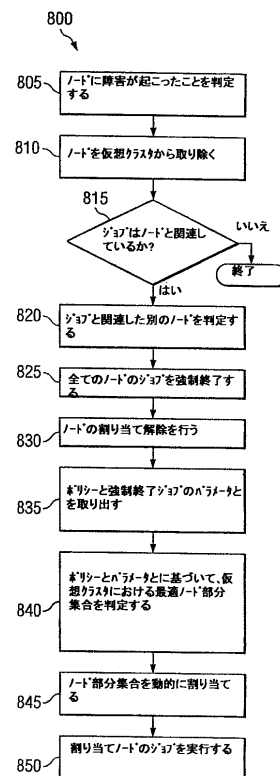
【図 1 2】



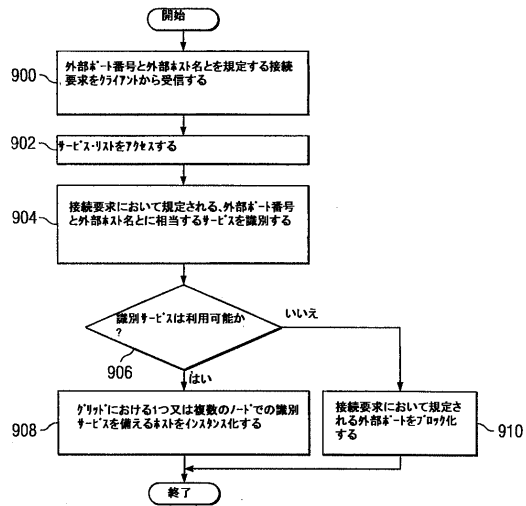
【図 1 3】



【図 1 4】



【図 15】



フロントページの続き

審査官 久保 正典

- (56)参考文献 特表2003-531412 (JP, A)
特開平05-241804 (JP, A)
特開平07-200496 (JP, A)
特開平07-141305 (JP, A)
国際公開第02/084509 (WO, A1)

- (58)調査した分野(Int.Cl., DB名)
G06F15/80
G06F9/06
G06F9/46-9/54