(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7175924号
(P7175924)

(45)発行日　令和4年11月21日(2022.11.21)

(24)登録日　令和4年11月11日(2022.11.11)

(51)国際特許分類

		FI		
G06F	*9/48 (2006.01)*	G06F	9/48	300H
G06F	*9/38 (2006.01)*	G06F	9/38	370Z
G06F	*9/50 (2006.01)*	G06F	9/50	150E

請求項の数　15　（全24頁）

(21)出願番号　特願2019-563486(P2019-563486)
(86)(22)出願日　平成30年4月27日(2018.4.27)
(65)公表番号　特表2020-520519(P2020-520519A)
(43)公表日　令和2年7月9日(2020.7.9)
(86)国際出願番号　PCT/US2018/029769
(87)国際公開番号　WO2018/212958
(87)国際公開日　平成30年11月22日(2018.11.22)
審査請求日　令和2年12月23日(2020.12.23)
(31)優先権主張番号　15/596,306
(32)優先日　平成29年5月16日(2017.5.16)
(33)優先権主張国・地域又は機関
米国(US)

(73)特許権者　502303739
オラクル・インターナショナル・コーポレイション
アメリカ合衆国カリフォルニア州９４０６５レッドウッド・シティー，オラクル・パークウェイ５００
(74)代理人　110001195弁理士法人深見特許事務所
(72)発明者　コラチャラ，スブラマンヤム
アメリカ合衆国、３００９６ ジョージア州、ダルース、マクルーア・ウッズ・ドライブ、３５２５
(72)発明者　シュ，ジエンウー
アメリカ合衆国、３０００５ ジョージア州、アルファレッタ、ストーンベリー・ドライブ、５８５

最終頁に続く

(54)【発明の名称】　計算プロセスの動的並列化

(57)【特許請求の範囲】

【請求項１】

ルールベースの数式のセットを評価するために並列計算プランを生成するためのコンピュータによって実行される方法であって、

前記ルールベースの数式のセットを複数のタスクユニットに分割するステップと、

前記複数のタスクユニットを、並列化可能なタスクグループのシーケンシャルなセットに配列するステップとを備え、前記ルールベースの数式のセットに関連付けられた論理的依存性および問題分割によってシーケンシャルな順序が決定され、前記方法はさらに、

前記並列化可能なタスクグループのシーケンシャル実行順序と、前記並列化可能なタスクグループの各々の中の前記タスクユニットの並列実行順序とを備える前記並列計算プランを生成するステップを備え、

前記ルールベースの数式の各々は、１つ以上の右辺の項と、当該右辺の項の関数として表される左辺の項を含み、

前記複数のタスクユニットは、前記ルールベースの数式のセットにおける前記ルールベースの数式の各々を評価するために必要な１つ以上の初期化タスクユニットを備え、

各前記１つ以上の初期化タスクユニットは、前記ルールベースの数式の各々について、当該数式の左辺の項に関連付けられた記憶場所をクリアすること、および前記右辺の項に関連付けられたデータを定量化した定量化データを格納するデータ構造を作成することを含む、コンピュータによって実行される方法。

【請求項２】

前記複数のタスクユニットは、前記数式のセットにおける前記ルールベースの数式の各々を評価するために必要な１つ以上の集約のタスクユニットと、１つ以上の計算タスクユニットとを備え、前記集約のタスクユニットは前記右辺の項に関連付けられたデータを定量化するため実行され、前記定量化データは前記計算タスクユニットにより使用され、前記集約のタスクユニットの実行は前記計算タスクユニットの実行に先行する、請求項１に記載のコンピュータによって実行される方法。

【請求項３】

前記並列化可能なタスクグループのシーケンシャルなセットからの１つ以上の並列化可能なタスクグループを並列化可能なサブタスクグループの並列なセットに分割するステップをさらに備える、請求項１または２に記載のコンピュータによって実行される方法。

【請求項４】

前記並列化可能なサブタスクグループの並列なセットは、前記並列化可能なタスクグループのシーケンシャルなセットからの前記１つ以上の並列化可能なタスクグループの範囲ベースの並列化によって生成される、請求項３に記載のコンピュータによって実行される方法。

【請求項５】

前記並列計算プランは、計算エンジンによって生成され、中間層レイヤーに送られて、スケジューリングされ、実行される、請求項１～４のいずれかに記載のコンピュータによって実行される方法。

【請求項６】

前記計算エンジンは、小売り予測アプリケーションサーバ（ＲＰＡＳ）の一部である、請求項５に記載のコンピュータによって実行される方法。

【請求項７】

前記並列計算プランは、コンピュータのクラスタにわたる並列実行をスケジューリングされる、請求項１～６のいずれかに記載のコンピュータによって実行される方法。

【請求項８】

ルールベースの数式のセットを評価するために並列計算プランを生成するためのシステムであって、

ルールベースの数式の１つ以上のセットを複数のタスクユニットに分割するための第１のユニットと、

前記複数のタスクユニットを、並列化可能なタスクグループのシーケンシャルなセットに配列するための第２のユニットとを備え、前記ルールベースの数式のセットに関連付けられた論理的依存性および問題分割によってシーケンシャルな順序が決定され、前記システムはさらに、

前記並列化可能なタスクグループのシーケンシャル実行順序と、前記並列化可能なタスクグループの各々の中の前記タスクユニットの並列実行順序とを備える前記並列計算プランを生成するための第３のユニットを備え、

前記ルールベースの数式の各々は、１つ以上の右辺の項と、当該右辺の項の関数として表される左辺の項を含み、

前記複数のタスクユニットは、前記ルールベースの数式のセットにおける前記ルールベースの数式の各々を評価するために必要な１つ以上の初期化タスクユニットを備え、

各前記１つ以上の初期化タスクユニットは、前記ルールベースの数式の各々について、当該数式の左辺の項に関連付けられた記憶場所をクリアすること、および前記右辺の項に関連付けられたデータを定量化した定量化データを格納するデータ構造を作成することを含む、システム。

【請求項９】

前記複数のタスクユニットは、前記数式のセットにおける前記ルールベースの数式の各々を評価するために必要な１つ以上の集約のタスクユニットと、１つ以上の計算タスクユニットとを備え、前記集約のタスクユニットは前記右辺の項に関連付けられたデータを定量化するため実行され、前記定量化データは前記計算タスクユニットにより使用され、前

記集約のタスクユニットの実行は前記計算タスクユニットの実行に先行する、請求項８に記載のシステム。
【請求項１０】
前記第２のユニットは、さらに、前記並列化可能なタスクグループのシーケンシャルなセットからの１つ以上の並列化可能なタスクグループを並列化可能なサブタスクグループの並列なセットに分割する、請求項８または９に記載のシステム。
【請求項１１】
前記並列化可能なサブタスクグループの並列なセットは、前記並列化可能なタスクグループのシーケンシャルなセットからの前記１つ以上の並列化可能なタスクグループの範囲ベースの並列化によって生成される、請求項１０に記載のシステム。
【請求項１２】
前記第１のユニットおよび前記第２のユニットは、計算エンジンを構成し、前記計算エンジンによって生成された前記並列計算プランは、中間層レイヤーに送られて、スケジューリングされ、実行される、請求項８～１１のいずれかに記載のシステム。
【請求項１３】
前記計算エンジンは、小売り予測アプリケーションサーバ（ＲＰＡＳ）の一部であり、前記並列計算プランは、コンピュータのクラスタにわたる並列実行をスケジューリングされる、請求項１２に記載のシステム。
【請求項１４】
前記並列計算プランは、コンピュータのクラスタにわたる並列実行をスケジューリングされる、請求項８～１３のいずれかに記載のシステム。
【請求項１５】
請求項１～７のいずれかに記載の方法をコンピュータに実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
分野
一実施形態は、一般にプランニングシステムに向けられ、特にプランニングシステムにおける計算プロセスの向上に向けられる。
【背景技術】
【０００２】
背景情報
プランニングシステムは、小売業者が小売業全体にわたって売上高、営業利益率および在庫回転率を計画および管理することに役立つ。このようなシステムでは、ビジネスロジックは、「ルール」を使用して表されてもよい。たとえば、ルール：売上高＝価格×個数は、「売上高」を一個当たりの「価格」および販売「個数」に応じて表し、一個当たりの「価格」に販売「個数」を乗算することによってそれを評価する。ルールにおける各項は、プランニングデータベースにおけるデータオブジェクトにマッピングされる。大規模なプランニングシステムは、このようなルールを数百から数千個含んでもよい。これらのルールは、予め構成されるか、または動的に生成される。このようなルールは、時にはオンラインで、時にはバッチ形式で、頻繁に再評価されなければならない。データの量が多く、システムにおいて定義されるルールの数が多いために、計算の性能がプランニングシステムの全体性能の重要な側面になる。
【発明の概要】
【０００３】
概要
一実施形態は、ルールベースの数式のセットを評価するために並列計算プランを生成するシステムに向けられる。上記システムは、上記ルールベースの数式のセットを複数のタスクユニットに分割する。次いで、上記システムは、上記複数のタスクユニットを、並列化可能なタスクグループのシーケンシャルなセットに再配列し、上記ルールベースの数式

のセットに関連付けられた論理的依存性および問題分割によってシーケンシャルな順序が決定される。

【図面の簡単な説明】

【０００４】

【図１】本発明の実施形態に係る、計算プロセスを最適化するための並列計算プラン生成モジュールを含むコンピュータサーバ／システムのブロック図である。

【図２】本発明の実施形態に係る計算並列化プランのブロック図である。

【図３】本発明の実施形態に係るＲＰＡＳプラットフォーム上での計算並列化プラン実行の動作フロー図である。

【図４】本発明の実施形態に係る並列計算プランにおけるシーケンシャル順序付け基準の図である。

【図５】本発明の実施形態に係るメジャー（measure）ベースの並列化された計算プランのブロック図である。

【図６】本発明の実施形態に係るメジャーベースおよび範囲ベースの並列化された計算プランのブロック図である。

【発明を実施するための形態】

【０００５】

詳細な説明

アナリティクスの計画および管理のためのルールベースの数式の評価は、数式間の論理的相互依存性によって決定されるシーケンシャルなプロセスである。数式のセットの評価は、しばしば、シーケンシャルなパターンを含む。なぜなら、特定の数式の評価は、リスト内の他の箇所に登場する数式の計算結果に左右され得るからである。したがって、既存の計算プラットフォームに関連付けられた計算エンジンは、既存の論理的依存性を識別し、それに応じて、ルールセットまたはルールベースの数式リストの評価を実行するために必要な計算プロセスをシーケンシャル化する。しかし、ルールセットのシーケンシャルな評価は、ビッグデータアナリティクスおよび管理のための高度にスケーラブルな計算および処理プラットフォームを提供する新たな技術および高性能ハードウェアアーキテクチャの開発と足並みをそろえてこなかった。新たな高性能の計算および処理アーキテクチャの利用を可能にする革新的アプローチを使用して、ルールベースの数式リストまたはルールセットの評価に関連する速度およびスケーラビリティを向上させる必要がある。

【０００６】

旧世代のソリューションは、「スケールアップ」によって（すなわち、より高価なハードウェアを利用することによって）性能問題に対処する傾向がある。しかし、クラウドアプローチでは、既存のソリューションは、「水平スケーリング」によって（すなわち、より多くのコモディティハードウェアを追加して実行を並列に行うことによって）性能問題に対処する傾向がある。ルールベースの計算では、水平スケーリングアプローチにおける重要な問題は、ルールベースの計算を並列に実行することに関連する難しさである。既存のソリューションでは、可能なら、アプリケーション開発者または設計者が、どのようにして計算を分割して並列に実行できるかを指定する。

【０００７】

本発明の実施形態では、シーケンシャルに実行される計算ステップを、並列化されたタスクグループのシーケンシャルなセットに分解することによって、ルールベースの数式の計算のための並列化が実行される。本発明の実施形態によって開示されているルールベースの数式の計算プロセスを並列化することは、新たな高性能のスケーラブルな計算および処理プラットフォームを十分に活用することができる新規のアプローチであり、従来のアプローチと比較して速度、時間および効率の点で大幅な性能の向上をもたらす。

【０００８】

なお、本発明の実施形態によれば、並列化されたまたは並列化可能なタスクまたはサブタスクグループとは、複数の処理ノードにわたってまたは複数の処理スレッドを使用して並列に実行され得る、同一の階層計算レベルにおける論理的に独立したオペレーションを

含むタスクまたはサブタスクのグループを指す。タスクまたはタスクユニットとは、実行プラン内で個々に実行可能なオペレーションの一単位を指す。

【０００９】

さらに、本発明の実施形態によれば、並列化されたまたは並列化可能なタスクグループのシーケンシャルなセットとは、シーケンシャルな実行順序に従って配列されることにより、各々の並列化されたタスクグループの並列実行が、以前にシーケンシャルなセットの中に登場した１つ以上の並列化されたタスクグループの完了に左右される複数の並列化されたタスクグループを指す。このシーケンシャルな順序は、ルールセットにおける数式間の論理的依存性および／または問題分割（すなわち、関連する計算の階層レベル）によって決められてもよい。

【００１０】

実施形態は、ルールベースの数式リストを評価するために計算プロセスのシーケンシャルな側面の実行に並列化を導入する。実施形態は、ルールベースの数式リストに固有の論理的依存性のセットによって実現されるシーケンシャルな順序を維持しながら、シーケンシャルなステップの評価に関与するタスクおよびオペレーション間の並列化を導入する。その結果、シーケンシャルに実行される計算ステップのセットに関与する各計算ステップは、より速い速度で、より高い時間およびリソース利用効率で評価を行うことになる。複数のシーケンシャルな計算ステップを実行することを含む従来のルールセット評価アプローチと比較して、本発明の実施形態では、たとえばシーケンシャルな計算ステップを１つ以上の並列化されたタスクグループのセットとして再構築することによって、ルールセット評価プロセスに関連付けられた時間およびリソース利用を大幅に向上させる。

【００１１】

上記のように、既存のソリューションは、アプリケーション開発者または分析者に頼って、どのようにして計算プロセスを直接的または間接的に並列に実行できるかを指定する。これは、一般に、「スパーク」もしくは「Ｊａｖａ（登録商標）ストリームＡＰＩ」などのアプリケーションプログラミングインターフェイス（「ＡＰＩ」）またはスクリプトを使用することによって達成され、または実行フローをパイプライン（たとえば、「アパッチピッグ」または「クランチ」）として指定することによって達成される。並列化を実行プロセスに組み込むこれらの既存の方法論の共通の特徴は、それらが予め定義されなければならないことである。本発明の実施形態では、ルールセット評価に関与する実行プロセスを動的に並列化して、性能を実質的に向上させてオペレーション時間を削減するユニークな方法について説明する。

【００１２】

実施形態では、ルールベースの数式のセットを自動的に分析して、１つ以上のコンピュータクラスタにわたって並列に実行可能な計算タスクのセットに変換することによって、ルールセット計算性能を向上させる。実施形態では、ルールベースの数式のセットを分析し、論理的依存性および問題分割に基づいてルールをより小さなタスクに分解し、次いで、生成されたタスクおよび問題分割に基づいてルールのセットのための並列実行プランを自動的に生成する。次いで、実行プランは、送られて、コンピュータクラスタを利用して並列に実行される。

【００１３】

スケーラブルプロセッサアーキテクチャ（「ＳＰＡＲＣ」）などの既存の技術は、プログラミング言語を介して並列化を実行する。したがって、計算プロセスにおける依存性を指定して実行順序を決定するのは、プログラマである。これに対して、本発明の実施形態は、ルールベースの数式のセットのための並列実行プランを生成することを含む計算並列化への高度に設定可能で動的なアプローチを開示する。したがって、本発明の実施形態によれば、依存性は、評価対象のルールベースの数式に基づいて動的にアドレス指定される。

【００１４】

本発明の実施形態では、ルールセットを評価することに関連付けられた計算は、複数のより小さなタスクに分割され、複数の並列化されたタスクグループに配列される。並列化

されたタスクグループに配列されるタスクは、並列に実行され得る各々のルールベースの数式の構成要素を備える。これらの構成要素は、前に引き出されて、１つ以上の並列化されたタスクグループの一部として並列に実行されてもよく、１つ以上の並列化されたタスクグループは、さらに、１つ以上の並列化されたサブタスクグループに細分されてもよい。ＳＰＡＲＣに関する１つの相違点は、開示されている実施形態では数式および範囲に基づいて並列化が実行されることである。

【００１５】

図１は、本発明の実施形態に係るコンピュータサーバ／システム（すなわち、システム１０）のブロック図である。単一のシステムとして示されているが、システム１０の機能は、分散システムとして実現されてもよい。さらに、本明細書に開示されている機能は、ネットワーク上で結合され得る別々のサーバまたはデバイス上で実現されてもよい。さらに、システム１０の１つ以上の構成要素は、含まれていなくてもよい。たとえば、データベース管理システムの機能の場合は、システム１０は、一般にディスプレイ２４または図１に示される１つ以上の他の構成要素を必要としないサーバであってもよい。

【００１６】

システム１０は、情報を通信するためのバス１２または他の通信機構と、情報を処理するための、バス１２に結合されたプロセッサ２２とを含む。プロセッサ２２は、プロセッサ２２が複数の計算を並列に実行できるように各ノードが独立してスケジューリングされ得るように処理ノードのクラスタも備えてもよい。プロセッサ２２は、いかなるタイプの汎用または特殊目的プロセッサであってもよい。システム１０は、情報およびプロセッサ２２によって実行される命令を格納するためのメモリ１４をさらに含む。メモリ１４は、１つ以上のルールセットのための１つ以上の並列化された実行テーブルまたは計算プランも含んでもよく、これらの１つ以上の並列化された実行テーブルまたは計算プランは、複数の処理ノードにわたる並列実行をスケジューリングされてもよい。メモリ１４は、ランダムアクセスメモリ（「ＲＡＭ」）、リードオンリメモリ（「ＲＯＭ」）、磁気もしくは光ディスクなどのスタティックストレージ、またはその他のタイプのコンピュータ読取可能媒体の任意の組み合わせで構成することができる。システム１０は、ネットワークへのアクセスを提供するためのネットワークインターフェイスカードなどの通信装置２０をさらに含む。したがって、ユーザは、直接的に、またはネットワークを介してリモートで、またはその他の方法で、システム１０と接続してもよい。

【００１７】

コンピュータ読取可能媒体は、プロセッサ２２によってアクセス可能であって、かつ、揮発性および不揮発性媒体、リムーバブルおよびノンリムーバブル媒体、ならびに通信媒体を含む任意の入手可能な媒体であってもよい。通信媒体は、搬送波または他の搬送機構などの変調データ信号の中にコンピュータ読取可能な命令、データ構造、プログラムモジュールまたは他のデータを含んでもよく、任意の情報配信媒体を含む。

【００１８】

プロセッサ２２は、さらに、バス１２を介して液晶ディスプレイ（「ＬＣＤ」）などのディスプレイ２４に結合されてもよい。キーボード２６およびコンピュータマウスなどのカーソル制御装置２８は、さらに、ユーザが必要に応じてシステム１０と接続できるようにバス１２に結合されてもよい。

【００１９】

一実施形態では、メモリ１４は、プロセッサ２２によって実行されたときに機能を提供するソフトウェアモジュールを格納する。これらのモジュールは、オペレーティングシステム機能をシステム１０に提供するオペレーティングシステム１５を含む。これらのモジュールは、並列化された実行スキームを生成するための並列計算プラン生成モジュール１６、および本明細書に開示されている全ての他の機能をさらに含む。システム１０は、オラクル社製のオラクルデータベースシステムまたは任意のデータベース管理システムに対する追加機能などの、より大規模なシステムの一部であってもよい。したがって、システム１０は、追加機能モジュール１８などの１つ以上の追加機能モジュールを含み得る。デ

ータベース１７は、並列計算プラン生成モジュール１６および追加機能モジュール１８（すなわち、データベース管理モジュール）に集中ストレージを提供するようにバス１２に結合される。一実施形態では、データベース１７は、非構造化照会言語（「ＮｏＳＱＬ」）データベースであり、並列計算プラン生成モジュール１６は、オラクル小売り予測アプリケーションサーバ（「ＲＰＡＳ」）の一部として実現される。本発明のさまざまな実施形態に関連付けられたこれらの例示的な特徴、すなわちＮｏＳＱＬデータベースおよびＲＰＡＳについては、以下でさらに詳細に説明する。

【００２０】

オラクル小売り予測アプリケーションサーバ（一般に「ＲＰＡＳ」と称される）は、予想およびプランニングアプリケーションを開発するための設定可能なソフトウェアプラットフォームである。ＲＰＡＳプラットフォームは、多次元データベース構造バッチおよびオンライン処理、設定可能なスライス・アンド・ダイスユーザインターフェイス、高性能の設定可能な計算エンジン、インポートおよびエクスポートなどのユーザセキュリティおよびユーティリティ機能などの機能を提供する。

【００２１】

ＲＰＡＳは、オラクル小売り需要予想、オラクル製品財務計画、オラクル品揃え計画、オラクルアイテム計画、オラクルサイズプロファイル最適化、オラクル補充最適化およびオラクル高度在庫計画などのオラクル小売りソリューションフットプリントの一部である相当数のアプリケーションの基礎である。現在のところ、ＲＰＡＳは、２つのデータ永続性オプション、すなわちオラクル・バークレー・データベースに基づく専有のデータストア、またはオラクルＲＤＢＭＳ、を提供する。

【００２２】

ＮｏＳＱＬデータベースは、リレーショナルデータベースで使用される表形式の関係以外の手段でモデル化されるデータの格納および検索のための機構を提供する。ＮｏＳＱＬデータベースは、ビッグデータおよびリアルタイムウェブアプリケーションでますます使用されるようになっている。ＮｏＳＱＬシステムは、ＳＱＬのような照会言語をサポートできることを強調するために、時には「ノット・オンリー・ＳＱＬ」とも呼ばれる。このアプローチに対する動機付けとしては、設計がシンプルであること、マシンのクラスタへの「水平」スケーリング（リレーショナルデータベースにとっての問題である）がよりシンプルであること、および可用性に対する微調整がきくことが挙げられる。ＮｏＳＱＬデータベースによって使用されるデータ構造（たとえば、キーバリュー、ワイドカラム、グラフまたはドキュメント）は、リレーショナルデータベースでデフォルトで使用されるものとは異なっており、ＮｏＳＱＬにおけるいくつかのオペレーションを高速化する。所与のＮｏＳＱＬデータベースの特定の適性は、それが解決しなければならない問題に左右される。往々にして、ＮｏＳＱＬデータベースによって使用されるデータ構造は、リレーショナルデータベーステーブルよりも「柔軟性がある」とも考えられる。

【００２３】

一般に、本発明の実施形態では、各ステップを複数の独立したタスクおよびサブタスクに分割することによって、ルールセット評価に関与する各々のシーケンシャルな計算ステップ中に並列化を生じさせ、これらの複数の独立したタスクおよびサブタスクは、可能なときは並列に実行されてもよく、可能でないときはサブタスクが順番に実行される。

【００２４】

図２は、本発明の実施形態に係る並列化プランの概要を示す。並列計算プラン２０２は、図２における２０４，２０６，２０８および２１０によって指定されるルールベースの数式リストまたはルールセットの評価を実行するために１つ以上のシーケンシャルな計算ステップを含む。シーケンシャルなステップは、各々のシーケンシャルな計算ステップが実行可能になる前に以前のステップの完了を必要とするようにシーケンシャルな関係を維持する。しかし、各々のシーケンシャルなステップは、並列タスクのセットとして実行されてもよい。タスクは、実行プラン内で個々に実行可能なオペレーションの一単位である。

【００２５】

図２における計算プラン２０２を参照して、第１のシーケンシャルなステップ２０４は、独立して同時に実行され得る並列タスク１～４に分解される。この並列実行の結果、第１のシーケンシャルなステップ２０４は、より早く完了することになり、その結果、第２のシーケンシャルなステップ２０６が早く開始される。同様に、第２のシーケンシャルなステップ２０６は、図２に示されるように並列化可能なタスクのセットとして実行されてもよく、その結果、第３のシーケンシャルなステップ２０８の開始がさらに早められる。同一のオペレーションが第４のシーケンシャルなステップ２１０でも同様に行われる。このようにして、計算プラン２０２は、シーケンシャルなステップの並列実行により、大幅なスピードアップを提供しながら計算ステップのシーケンシャルな順序を維持する並列化可能なタスクグループのシーケンスに分解される。

【００２６】

図３は、実施形態に係る並列化された計算プロセスの動作フローを示す。サーバ側のマイクロサービス層３０２が、ルールセット評価プロセスを起動する。命令がＲＰＡＳ層３０４に送られ、そこで、並列化された計算プランが計算エンジン（CalcEngine）によって生成される。CalcEngineは、ルールセットの評価を実行するために必要なタスクのシーケンシャルかつ並列な実行パターンを論理的に判断し、それに応じて実行プランを生成する。バックエンドで実行されるCalcEngineは、タスクスケジューリング機能を持たないため、プランは、一旦CalcEngineによって生成されると、中間層マイクロサービス層３０２に戻されて、実行され、並列実行のための処理ノードのクラスタに分散される。たとえば、CalcEngineによって生成された実行プランが、たとえば並列に実行され得る５００個の並列タスクを含むように特定のシーケンシャルなステップを指定する場合、マイクロサービス層３０２は、１つの計算タスクを各ノードに割り当てる５００個のノードにオペレーションを分散させてもよい。図３に示されるように、計算プランの実行は、ＲＰＡＳ層におけるタスク実行部３０８，３１０および３１２にわたるタスクの第１のセットの並列実行をスケジューリングするマイクロサービス層３０２によって指示される。実行が完了すると、プラン実行部は、計算の次のシーケンシャルなステップに関連付けられた並列タスクの後続のセットがスケジューリングされて、タスク実行部３１４，３１６および３１８に割り当てられ得るように通知される。このようにして、シーケンシャルなステップに含まれるタスクは、複数のタスク実行部にわたって同時に実行され、これにより、ルールセットの評価が大幅にスピードアップされる。

【００２７】

図３に示されるように、マイクロサービス層（中間層）３０２は、前の並列化されたタスクグループを実行するタスク実行部から完了通知を受信したときにのみ並列タスクの各セットがスケジューリングされるので計算のシーケンシャルな順序が維持されることを確実にしながら、タスクの並列実行をスケジューリングする。計算プランは、ＮｏＳＱＬデータベース層３２０におけるプラン／タスクテーブル３１９に格納される。タスクの実行は、メジャーデータテーブルからメジャーデータを読み取ることと、タスクの実行結果を用いてメジャーデータテーブルを更新することとを含む。したがって、ＮｏＳｑｌデータベース層３２０は、メジャーデータテーブル３２０も含む。メジャーは、特定のタイプのデータを保持するエンティティに相当する。たとえば、メジャーは、「売上高」について作成されてもよい。そして、売上高メジャーは、全てのストック・キーピング・ユニット（「ＳＫＵ」）についての関連データ（すなわち、売上データ）を含む。

【００２８】

ルールエンジンによって生成される最初のルールベースの数式リストは、分解されて、複数のシーケンシャルに順序付けられた計算ステップとして再編成される。計算ステップのシーケンシャルな順序は、計算ステップ間の論理的依存性および数式リストにおける評価対象のメジャーの交差によって決まる。したがって、他の数式の結果に依存しない全ての数式、たとえばＡ＝５またはＢ＝今日は、一緒に１つのシーケンシャルなステップにプールされて、並列に実行されてもよい。しかし、特定の数式は、他の数式の評価結果に依存し得る。たとえば、数式「Ａ＝Ｂ．ｔｏｐ」の評価では、「Ｂ」の計算を最初に完了さ

せる必要がある。したがって、「Ｂ」の評価は、この場合、「Ａ」の評価に先行しなければならないため、シーケンシャルな計算順序は、このシーケンシャルな関係を反映しなければならない。

【００２９】

上記のように、ルールセットの評価に必要なシーケンシャルな順序およびシーケンシャルな計算ステップの数を判断するための１つの基準は、計算が実行される階層または交差レベルである。たとえば、より高いレベルでのメジャーの計算は、より低いレベルでのこのメジャーに関連付けられた値の集約を必要とし得る。計算レベルの変化は、並列処理レベルの変化を伴うため、新たな計算ステップを必要とし得る。したがって、数式のレベル間評価に関連付けられた計算は、さまざまなシーケンシャルなステップにおいて評価されるが、同一レベルの数式に関連付けられた計算は、同一のステップにおいて並列に評価され得る。上記の関連付けられた計算レベルに加えて、ルールセットにおける数式間の論理的依存性は、本発明の実施形態に係るルールセットの完全な評価に関連付けられた複数のシーケンシャルな計算ステップの境界を明示する障壁として機能する。

【００３０】

既存の論理的依存性および関連付けられた計算間の交差またはレベル変化に基づいて、ルールベースの数式リストがシーケンシャルな計算ステップのセットに分解されて再編成されると、次のステップでは、各ステップに関連付けられた計算を実行する。ルールベースの数式は、１つ以上の右辺（「ＲＨＳ」）メジャー（項）の関数として表される左辺（「ＬＨＳ」）メジャー（項）を含む式または数式として表されてもよい。したがって、シーケンシャルな計算ステップに関連付けられた計算を実行することは、ＲＨＳメジャーの関数としてＬＨＳメジャーを評価することを含んでもよい。この動作は、最初に、ＲＨＳメジャーを定量化するために集約サブタスクを実行することを必要とし得る。集約サブタスクの実行は、さらに、初期化サブタスクの完了を必要とし得て、この初期化サブタスクは、関連付けられた記憶場所から１つ以上のＬＨＳメジャーインスタンスをクリアすることと、新たなＲＨＳメジャーインスタンスのためのデータ構造を集約レベルで作成することとを含む。したがって、計算ステップは、シーケンシャルに順序付けられたサブタスクセット、すなわち、メジャーデータをクリーンアップして、集約レベルメジャーインスタンスを格納するための新たなデータ構造を作成するための初期化サブタスクと、集約レベルメジャーインスタンスを生成するための集約サブタスクと、最後に、集約レベルメジャーインスタンスをパラメータとして使用して実際の数式を評価するための計算サブタスクとを含んでもよい。

【００３１】

複数の初期化サブタスクは、オペレーションの独立性のために同時に実行されてもよい。同様に、独立したメジャーインスタンスを含む複数の集約サブタスクは、並列に実行されてもよい。したがって、含まれる計算のタイプによっては、いくつかの計算ステップは、初期化サブタスクおよび／または集約サブタスクのうちの一方または両方を含まなくてもよい。したがって、計算プランの形成は、動的なプロセスである。最後の計算ステップに続く計算プランの終了時に、最後のクリアアップタスクが実行されてもよい。

【００３２】

図４は、実施形態に係る並列計算プラン４０２を示す。図４に示されるように、計算プラン４０２は、依存性および／または交差もしくは問題分割基準に基づいてシーケンシャルな計算ステップ４０４～４１０のセットに分解される。

【００３３】

図５は、並列化されたタスクグループ５０２，５０４および５０６をそれぞれ含む３つのサブステップのシーケンスへの第１のシーケンシャルなステップ４０４の並列分割を示す。並列化されたタスクグループ５０２は、クリーンアップタスク５０８とデータ構造作成タスク５１０とを含む初期化オペレーションを実行する。クリーンアップタスク５０８は、初期化ステップ５０２に配列されて並列に実行され得る、評価対象のルールセットにおける全ての数式に関連付けられたメジャークリアタスクに相当し得る。評価対象のルー

ルセットにおける全ての数式に関連付けられたデータ構造またはメジャー作成タスク５１０も、初期化サブステップ５０２に配列されて、クリーンアップタスク５０８と並列に実行される。図５に示される本発明の実施形態によれば、初期化サブステップ５０２は、集約サブタスク５１２，５１４および５１６によって実行されるメジャーＡ，ＢおよびＣの集約を含む第２の並列化されたタスクグループ５０４の実行に先行しなければならない。たとえば、５０４におけるメジャーＡ，ＢおよびＣの集約に先行してメジャーＡ，ＢおよびＣの集約レベルインスタンスを格納するために、５０２において、指定のメモリブロックがクリアされなければならず、必要なメモリが割り当てられて初期化されなければならない。

【００３４】

上記のように、データ構造またはメジャーインスタンス作成タスク５１０は、同時に、メジャークリアアップタスク５０８と並列に行われてもよく、初期化プロセスを大幅にスピードアップさせる。

【００３５】

並列化されたタスクグループ５０４を含む第２のシーケンシャルなサブステップは、メジャーデータの集約、具体的にはメジャーＡを集約するための集約サブタスク５１２、メジャーＢを集約するための集約サブタスク５１４、およびメジャーＣを集約するための集約サブタスク５１６、を含む。集約オペレーションは、プランニングシステムに関連付けられた複数の店舗および／または売り場にわたって全てのストック・キーピング・ユニット（「ＳＫＵ」）についての全てのデータを合計することを含んでもよい。これは、非常に時間のかかるタスクであり得る。したがって、独立して集約されて全てのこのような集約タスクを並列に実行し得る各メジャーのために別々の集約タスクを作成することにより、ルールセットの評価時間を大幅に改善することができる。図５に示されるようなメジャーＡ，ＢおよびＣをそれぞれ集約するための集約サブタスク５１２，５１４および５１６が、異なる独立したメジャーについてのデータを集約することを含むので、それらは並列に実行されてもよい。

【００３６】

集約ステップ５０４が完了し、ルールセットにおける１つ以上の数式の実際の評価に必要なデータが利用可能になると、計算タスクグループ５０６における計算タスク５１８が実行されて、集約レベルメジャー値を使用してルールセットにおける数式を評価してもよい。共通の構成要素を含まない計算タスクは、並列に実行されてもよい。次いで、３つのシーケンシャルなサブステップの同一のセットが計算ステップ２，３，４で繰り返されてもよい。

【００３７】

上記のように、初期化、集約および計算サブステップは、シーケンシャルな実行順序を辿る。しかし、各サブステップは、ルールセットにおけるさまざまな数式に関連付けられた、並列に実行されるようにスケジューリングされた複数のサブタスクを含んでもよい。たとえば、初期化プロセスは、ルールセットにおけるＬＨＳメジャーインスタンスのうちの一部または全てをクリーンアップしながら、同時にルールセットにおける数式のうちの一部または全てに関連付けられた集約レベルでＲＨＳメジャーインスタンスを作成するために、複数のサブタスクを作成することを含んでもよい。したがって、ルールセットにおける複数の数式のためのクリーンアップおよび作成タスクは、単一の初期化サブステップに配列されて、並列に実行されてもよい。同様に、集約サブステップは、ルールセットにおける全ての数式から集約されたＲＨＳメジャーインスタンスを収集することと、並列に実行され得る全ての集約タスクのために複数のサブタスクを作成することとを含んでもよい。１つ以上の数式を評価するための計算サブステップも、既存の依存性および／またはルールセットにおける数式間の計算レベルに基づいて並列化されてもよい。現在のバッチにおいて全ての計算が完了した後で、または実行の途中でバッチが機能しなくなっても、最後のクリーンアップタスクが実行されてもよい。最後のクリーンアップタスクは、現在のバッチにおいて作成された全ての集約されたメジャーインスタンスを除去する。

【００３８】
依存性ベースの並列化に加えて、並列化されたタスクグループにおける各々のシーケンシャルなサブステップは、さらに、範囲による並列化を行ってもよい。範囲ベースの並列化は、同一の階層レベルにおける複数の独立したドメイン、エンティティまたはチャネル（すなわち、店舗、売り場など）にわたって単一のサブタスクの実行を並列化することを含む。この単一のサブタスクは、複数の数式、集約などを含み得る。範囲ベースの並列化は、追加の並列化層を計算プランに追加するため、ルールセット評価プロセスをさらにスピードアップさせる。
【００３９】
図６は、本発明の実施形態に係る、メジャーベースの並列化も範囲ベースの並列化も含む最終的な計算プラン６０２を示す。第１のシーケンシャルな計算ステップの並列化された実行パターン６０３は、それぞれ初期化、集約および計算オペレーションの第１のセットを実行するための３つの並列化されたタスクグループのシーケンスに分割される。並列化された実行パターン６０３は、メジャーデータクリーンアップオペレーション６０６および新集約レベルメジャーインスタンス作成オペレーション６０８に関連付けられた第１のシーケンシャルなサブステップ６０４を含む。並列化された実行パターン６０３の第１のシーケンシャルなサブステップ６０４は、ルールセットにおける全てのルールベースの数式に関連付けられた全ての初期化オペレーションの並列実行を構成する。これは、ルールセットにわたって並列化され得る全てのクリーンアップおよび新インスタンス作成オペレーションを含む。並列化された実行パターン６０３の第２のシーケンシャルなサブステップまたは並列化されたタスクグループ６１０は、メジャーＡおよびＢのための集約サブタスクを含み、この第２のシーケンシャルなサブステップまたは並列化されたタスクグループ６１０は、チャネルごとに、チャネル１についてメジャーＡおよびＢに関連付けられたデータを集約する並列化されたサブタスクグループ６１２、チャネル２についてメジャーＡおよびＢに関連付けられたデータを集約する並列化されたサブタスクグループ６１４、および１つ以上のチャネルについてメジャーＣに関連付けられたデータを集約する並列化されたサブタスクグループ６１６にさらに分割される。
【００４０】
なお、並列化されたサブタスクグループ６１２，６１４および６１６は、計算プラン６０２の第１のシーケンシャルな計算ステップに関連付けられた並列化された実行パターン６０３によって指定されるような同時実行をスケジューリングされる。したがって、計算プラン６０２における第１のシーケンシャルな計算ステップの第２のシーケンシャルなサブステップ６１０（集約サブステップ）は、まずメジャーによってメジャーＡ，ＢおよびメジャーＣにわたって並列化され、次いで範囲によって並列化されて、複数のチャネルにわたるメジャーＡ，ＢおよびＣの並列集約を同時に実行する。
【００４１】
一般に、たとえば計算ドメインの中にｎ個のチャネルがある場合、ｎ個全てのチャネルにわたってメジャーを並列に集約するためにｎ個の並列タスクが作成されてもよい。図６における計算プラン６０２は、分割または範囲ベースの並列化スキームに加えて、２つのメイン並列化スキーム、すなわち依存性ベースの並列化スキームを利用する。
【００４２】
計算プラン６０２の第１のシーケンシャルな計算ステップに関連付けられた並列化された実行パターン６０３は、第３のシーケンシャルなサブステップ６１８も含む。並列化された実行パターン６０３の第３のシーケンシャルなサブステップ６１８は、チャネル１について数式リストを評価するための並列化されたサブタスクグループ６２０と、チャネル２について数式リストを評価するための並列化されたサブタスクグループ６２２と、１つ以上の他のチャネルについて数式リストを評価するための並列化されたサブタスクグループ６２４とを含む。なお、並列化されたサブタスクグループ６２０，６２２および６２４は、計算プラン６０２の第１のシーケンシャルな計算ステップに関連付けられた並列実行パターン６０３によって指定されるような並列実行をスケジューリングされる。したがっ

て、計算プラン６０２における第１のシーケンシャルな計算ステップの第３のシーケンシャルなサブステップ６１８（計算サブステップ）は、まずメジャーによってメジャーＡ，ＢおよびＣにわたって並列化され、次いで範囲によって並列化されて、複数のチャネルにわたって数式リストの並列計算を同時に実行する。次いで、並列化された初期化、集約および計算タスクグループの同一のセットまたはサブセットが、６２６によって示されるように、計算プランにおける他の計算ステップのために実行されてもよい。

【００４３】

本発明の一実施形態によれば、エンプティベースのアレイを有する全てのメジャーについてシーケンシャルな集約タスクが作成され、スカラーに集約する場合にソース範囲集約タスクが作成され、最も外側の次元が小さい場合に二次元範囲集約タスクが作成され、最後に、宛先の最も外側の次元によって宛先範囲集約タスクが作成される。

【００４４】

本発明の一実施形態によれば、いかなる範囲並列化可能なタスクの範囲情報も以下の仕様に従って構築される。すなわち、ソース範囲集約の場合、範囲は、ソースアレイの最も外側の次元によって構築される。宛先範囲集約の場合、範囲は、宛先アレイの最も外側の次元によって構築される。二次元範囲集約の場合、範囲は、宛先アレイの最も外側および中間の次元によって構築される。非スカラー計算の場合、範囲は、ＬＨＳメジャーインスタンスの最も外側の次元によって構築される。

【００４５】

本発明の一実施形態によれば、メジャーベース（数式ベース）および範囲ベースの並列化スキームを含む並列化された計算プランは、各タスク内の並列化可能なサブタスク（メジャーベースおよび／または範囲ベース）の１つ以上のセットを識別する各タスクについての情報とともに、各々のシーケンシャルなステップ内の並列化可能なタスク（メジャーベースおよび／または範囲ベース）の１つ以上のセットを識別する各ステップについての情報を有するシーケンシャルなステップの一次式に平坦化されてもよい。

【００４６】

本発明の一実施形態によれば、１つ以上のルールセットの実質的に高速の評価を可能にするために、関連付けられた計算ステップ並列化情報とともにシーケンシャルな計算ステップの線形セットを含む平坦化された計算プランは、ＮｏＳＱＬデータベースなどの格納モジュールに格納され、ＲＰＡＳプラットフォーム上で実行されてもよい。

【００４７】

本発明の実施形態の１つの特定の実現例を表すサンプル疑似コードは、以下の通りである。

【００４８】

【数１】

```
//      Generating the Parallel Calcultion Plan
// Input is a list of expressions pre-ordered by Rule Engine
generate_plan(expr_list)
{
        for (expr in expr_list)
        {
                if expr does not depends on others, then
                {
                        pull this expr out from expr_list
                        put this expr in independ_expr_list
                }
        }

        if independ_expr_list is not empty, then
        {

generate_measure_based_parallel_plan_for_ind_expressions(
                        independent_expr_list
                )
        }

        generate_range_based_parallel_plan(expr_list)
}

// Extract the range parallelizable dimension of the expression
dimension_name
extract_parallelizable_level(expr)
{
        if (expr is special expression )
```

【００４９】

```
        {
            return outermost dimension of this expr's rangeableIntx
        }
        else
        {
            is the expression a constant expression?
            {
                //      Constant expression is considered as scalar,
identified by empty string
                return ""
            }
            else
            {
                return outermost dimension of LHS measure
instance's intersection
            }
        }
    }

    generate_measure_based_parallel_plan_for_ind_expressions(
        independent_expr_list)
    {
        // Break up all expressions by level of parallelization
        create map of dimension name to list of expressions
        for (expr in independent_expr_list)
        {
            parallelizable_dim = extract_parallelizable_level (expr)
            add expr to the map entry for parallelizable_dim
        }

        for each entry in the map
        {
```

【００５０】

```
                    let dim_name be the key in the entry
                    let sub_expr_list be expression list of the entry

                    if size of sub_expr_list is too large
                    we further break sub_expr_list to multiple smaller
sub_expr_list
                    for each smaller sub_expr_list or just the sub_expr_list itself
                    {

        generate_plan_for_same_level_same_group(dim_name, sub_expr_list)

                    }
            }
        }

        generate_range_based_parallel_plan(expr_list)
        {
            let outerdim_divided_list be a list of Pair (
                    dimension name,
                    list of expressions)

            for (expr in expr_list)
            {
                    outer_dim_name = extract_parallelizable_level (expr)

                    if outer_dim_name is different than the dim name of the last
element in outerdim_divided_list
                    {
                            append a new pair of (outer_dim_name, empty expr
list) to outerdim_divided_list
                    }
```

【００５１】

```
                append expr to the expression list of the last element in
outerdim_divided_list
            }

            for each pair of (dim_name, expr_list) in the outerdim_divided_list
            {
                generate_plan_for_same_level(dim_name,  expr_list)
            }

            store the generated plan and return
        }

        //      Generate the Range Parallelized Calculation Plan
        generate_plan_for_same_level(parallelizable_dim, expr_list)
        {
            if (parallelizable_dim is empty)
            {
                //      Handle the scalar evaluation case

        generate_plan_for_same_level_same_group(parallelizable_dim, expr_list)
            }
            else
            {
                //      Parallelization by range, first break up based on
barrier calculations
                let expr_list_list be a list of list of expressions
                for (expr in expr_list)
                {
                    let the current_expr_list be the last element in the
expr_list_list

                    first check barrier calculation
```

【００５２】

```
                    collect this expr's RHS (right handside) measure
instances
                    if any of the RHS measure instance satisfies:
                        1.  RHS is calculated by a previous
expression in current_expr_list
                        2.  And is at aggregated level above
parallelizable_dim
                    then
                    {
                        //  Expr must be evaluated after all
expressions in current_expr_list are evaluated
                        append a new expression list to the
expr_list_list
                        reset the current_expr_list to the new
expression list just created
                    }

                    append expr to current_expr_list
                }

                for (each sub_list in the expr_list_list)
                {

    generate_plan_for_same_level_same_group(parallelizable_dim, sub_list)
                }
            }
        }

        generate_plan_for_same_level_same_group(parallelizable_dim, expr_list)
        {
            //  First collect all measure instance that needs to be aggregated
or cleaned
```

【００５３】

```
let lhs_set be the set of LHS measure instances
let agg_map be a map that maps from aggregated intersection to
measure instance

for (expr in expr_list)
{
    collect measure instances in the expr,
    if the measure instance is at aggregated level
    add an entry to the agg_map, key is the aggregated
intersection,
        value is the set of measure instances

    collect LHS measure instances in the expr to lhs_set
}

//  Handle the trivial case - no further parallelization is needed
if agg_map is empty, and parallelizable_dim is also empty
{
    create a new calculation step
    create a sequential calculation task in this step
    add the expr_list to this new task
    the seqential calculation task will simply evaluate all
expressions
    in sequential mode, no further parallelization needed

    return;
}

//  Handle aggregation
if (agg_map is not empty, or lhs_set is not empty)
{
    create a new step of calculation (init_step), append it to
```

【００５４】

```
current plan
                also create a new task (init_task) that does the measure
creation and clean up
                add init_task to init_step

                for (all measure instances in the lhs_set and the agg_map)
                {
                        add the measure instance to the init_task
                        the init task, when executed, will create the measure
instance if not exist
                        or clean it up if it already exists
                }
        }

        //      Create aggregation step
let seq_agg_map, src_range_agg_map, dst_range_agg_map,
twodim_range_agg_map
be 4 maps, each maps intersection to a set of measure instances
        if the agg_map is not empty
        {
        for each intersection, set of measure instances pair in the agg_map
                if the measure to aggregate has no data
                {
                // Use regular sequential aggregation if no data in involved
                add this intersection and its associated set of measures to
                seq_agg_map
                }
                else if the intersection is scalar, like aggregating everything to
all product / all location
                {
                        //      Use source ranged aggregation
                        add this intersection and its associated set of
```

【００５５】

```
        measures to src_range_agg_map
            }
            else if the outer most dimension has small number of
positions, defined by a system threshold
                but next outer dimension has larger number of
positions
            {
                //      Use Two Dim Range Aggregation
                add this intersection and its associated set of
measures to twodim_range_agg_map
            }
            else
            {
                //      Range by Destination as default
                add the intersection and its associated set of
measures to dst_range_agg_map
            }
        }

        Create a new calculation step (agg_step), append it to current plan
        if seq_agg_map is not empty
        {
            create a sequential aggregation task, put all measures in the
seq_agg_map to this task
            add this task to the current aggregation step
        }

        if src_range_agg_map, dst_range_agg_map,
twodim_range_agg_map is not empty
        {
            for each (intersection, set of measures pair) in the three maps
            {
```

【００５６】

```
                    if per intersection, the set of measures is too large
                    break up the set of measures to smaller sets

                    create a source range agg task for entries in
src_range_agg_map
                        or a dst range agg task for entries in
dst_range_agg_map
                        or a twodim range agg task for entries in
twodim_range_agg_map
                    add the set of measures to this task
                    add the task to current agg step
                    further parallelize the task by range by adding
partition information to it
                }
            }

            create a Calculation step and append the step to current plan
            if (parallelizable_dim is empty)
            {
                create a sequential calc task
                add the expr_list to the sequential calc task
                add the task to the calculation step
            }
            else
            {
                create a range parallelized calculation task
                add the expr_list to the range based calc task
                add the task to the calculation step
                further parallelize the task by range by adding partition
information to it
            }
}
```

【００５７】

開示されているように、実施形態によって、数式特性に基づく計算プロセスの並列化が可能になり、計算の同時的な範囲ベースの並列化によってさらなる性能の向上が可能になる。

【００５８】

本明細書では、いくつかの実施形態が具体的に示され、および／または、説明されている。しかし、開示されている実施形態の変更例および変形例が、本発明の精神および所期の範囲から逸脱することなく、上記の教示によって包含され、添付の特許請求の範囲の範囲内であるということが理解されるであろう。

【図面】

【図１】

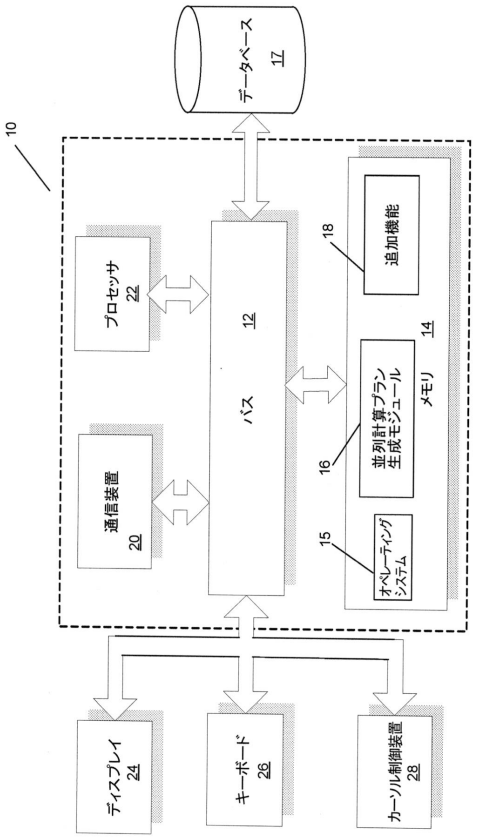

FIG. 1

【図２】

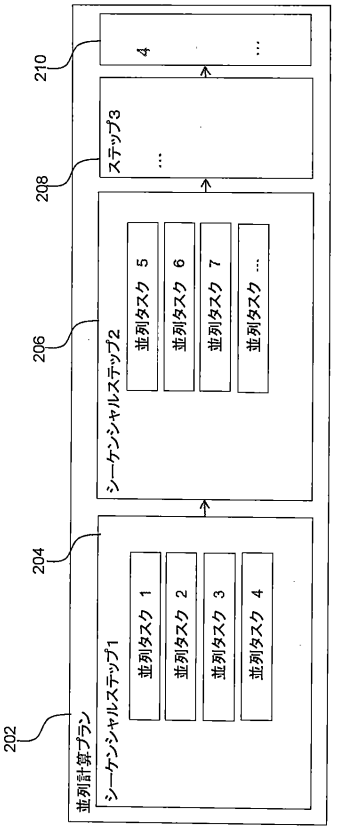

FIG. 2

【図３】

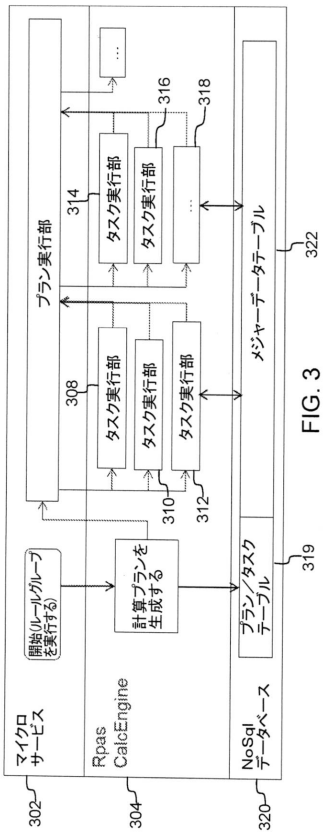

FIG. 3

【図４】

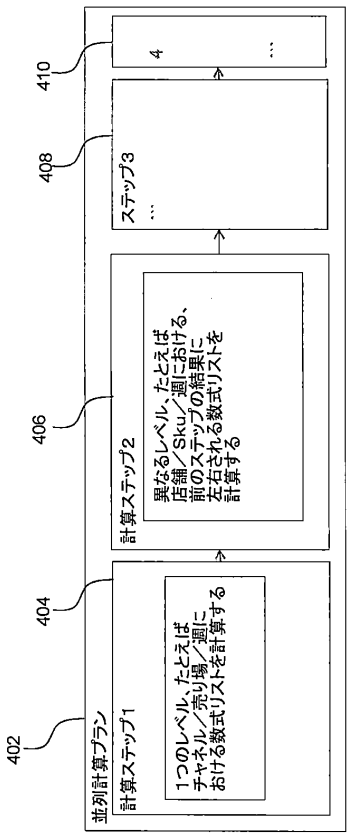

FIG. 4

【図５】

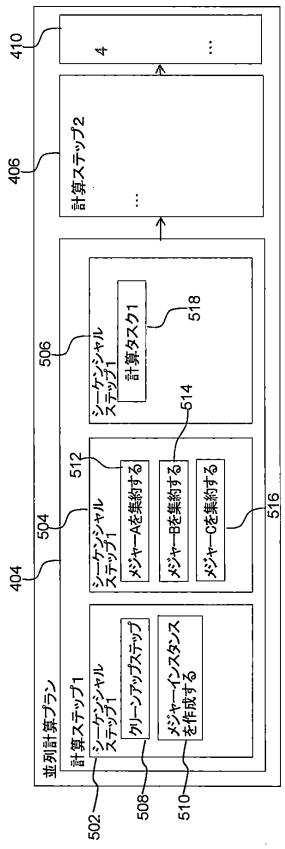

FIG. 5

【図６】

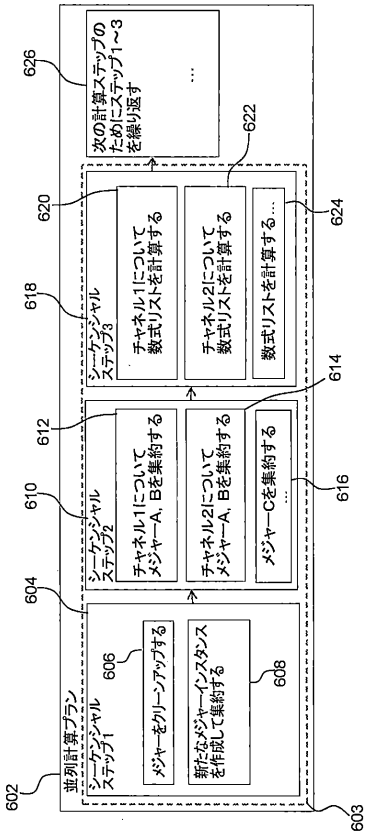

FIG. 6

フロントページの続き

(72)発明者　ホン，タク
アメリカ合衆国、３００６２ ジョージア州、マリエッタ、クロイスター・ドライブ、４０５９
(72)発明者　ロッデンベリー，ラリー
アメリカ合衆国、３０３０９ ジョージア州、アトランタ、ピーチツリー・ストリート・エス・イー・５１６、８７８
(72)発明者　フアン，ドンション
アメリカ合衆国、３００６２ ジョージア州、マリエッタ、ネソル・クリーク・ドライブ、２９６２
(72)発明者　ホスラビ，メディ
アメリカ合衆国、３００９７ ジョージア州、ダルース、ベルフライ・コート、８７３５
(72)発明者　ホランド，フィリップ
アメリカ合衆国、０２１３８ マサチューセッツ州、ケンブリッジ、デイナ・ストリート、５７
(72)発明者　パテル，ビマル
イギリス、ティ・ダブリュ・７ ４・キュー・エフ ミドルセックス、アイルワース、オスタリー・アベニュ、１１
(72)発明者　モハン，アヌープ
アメリカ合衆国、３００４０ ジョージア州、カミング、ノースビュー・レイク、５３６０
(72)発明者　シェティ，キラン
アメリカ合衆国、３０３２８ ジョージア州、アトランタ、ワン・グレンレイク・パークウェイ、フォース・フロア、キューブ・４１５９
(72)発明者　ラン，ウェイガン
アメリカ合衆国、３００９７ ジョージア州、ダルース、キャンペストラル・コート、９０４０
(72)発明者　ブレメケ，エリク
アメリカ合衆国、３０００９ ジョージア州、アルファレッタ、アダージョ・ドライブ、１８００
(72)発明者　フアン，ヂービン
アメリカ合衆国、３０３２８ ジョージア州、アトランタ、ワン・グレンレイク・パークウェイ、スイート・３００
(72)発明者　コルター，スコット
アメリカ合衆国、３００６０ ジョージア州、マリエッタ、アイボリー・トレイル、３１９３

審査官　河合 弘明

(56)参考文献　特開平０４－０５８３７０（ＪＰ，Ａ）
特開２００９－１５１６４５（ＪＰ，Ａ）
大越 雄一，外５名，「ＧＰＵクラスタを用いた大規模生体シミュレーションにおけるＣＰＵ・ＧＰＵ間データ転送の効率化」，情報処理学会 研究報告 ハイパフォーマンスコンピューティング（ＨＰＣ） ２０１３－ＨＰＣ－１４０ ［ｏｎｌｉｎｅ］ ，情報処理学会，2013年07月24日，第２０１３－ＨＰＣ－１４０巻，第３１号，pp.1-7
Judith MESKILL，「Oracle Retail Predictive Application Server Administration Guide for the Classic Client」，Release 14.1，Oracle Corporation，2014年12月，p.9-10，インターネット ＜URL: https://docs.oracle.com/cd/E12478_01/rpas/pdf/141/rpas-141-classic-admin.pdf＞

(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　９／４５５－９／５４
Ｇ０６Ｆ　９／３８