(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2010-244563
(P2010-244563A)
(43)公開日　平成22年10月28日(2010.10.28)

(51)Int.Cl.			F I			テーマコード（参考）
G06F	9/44	(2006.01)	G06F	9/06	620B	5B376
G06F	9/48	(2006.01)	G06F	9/46	452A	

審査請求 有　請求項の数 21　ＯＬ　（全 18 頁）

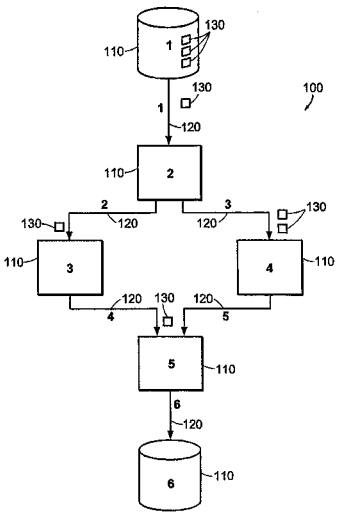
(21)出願番号　特願2010-126720 (P2010-126720)
(22)出願日　平成22年6月2日 (2010.6.2)
(62)分割の表示　特願2004-543637 (P2004-543637)の分割
原出願日　平成15年10月7日 (2003.10.7)
(31)優先権主張番号　10/268,509
(32)優先日　平成14年10月10日 (2002.10.10)
(33)優先権主張国　米国 (US)

(71)出願人　501278238
アビニシオ　ソフトウェア　エルエルシー
アメリカ合衆国　０２４２１　マサチューセッツ州　レキシントン，スプリング　ストリート　２０１
(74)代理人　110000213
特許業務法人プロスペック特許事務所
(72)発明者　スタンフィル　クレイグ　ダブリュ
アメリカ合衆国　０１７７３　マサチューセッツ州　リンカーン　ハックルベリー　ヒル　ロード　４３

Ｆターム(参考)　5B376 BB01　BC45　BC73

(54)【発明の名称】グラフに基づく計算の実行方法、当該方法を実行させる命令を格納するコンピュータ可読記憶媒体、並びに当該方法を実行するためのシステム

(57)【要約】
【課題】グラフに基づく計算を効率的に制御するための方法を提供する。
【解決手段】二つ以上のグラフテンプレートが生成される。各テンプレートは異なるタイプの計算グラフと関係付けられる。各計算グラフは相当数の頂点を含む。各頂点は前記計算の対応する部分と関係付けられる。プロセスの群が一つ以上形成される。計算グラフの各頂点はプロセスの群の対応する一つと関係付けられる。一つ以上のデータフローが処理される。各フローは、計算グラフの対応するタイプと結び付けられている。各フローの処理は、対応する計算グラフに対するテンプレートからグラフインスタンスを形成することを含む。インスタンスの各頂点に対し、コンピュータのリソースが対応する群から割り当てられる。各フローはインスタンスにより処理される。その処理には、前記割り当てられたプロセスを用いて、かかるインスタンスの頂点に対応する計算を実行することが含まれる。
【選択図】図１



【特許請求の範囲】
【請求項１】
データ記憶システムに格納されたコンピュータプログラムコードと連動して作動する少なくとも一つのプロセッサ及び揮発性データデータ記憶装置内の共有メモリセグメントを用いるコンピュータシステム上で、計算を表したグラフを実行する方法であって、
異なるタイプの計算グラフにそれぞれが関係付けられる二つ以上のグラフテンプレートを生成するステップであって、各テンプレートは、前記計算の対応する部分とそれぞれが関係付けられる幾つかの頂点を含み、
前記少なくとも一つのプロセッサ上で実行されるコンピュータシステムのプロセスの一つ以上の群を管理するステップであって、計算グラフの少なくとも一つの頂点がコンピュータシステムのプロセスの前記群の対応する一つと関係付けられ、
対応するグラフテンプレートにそれぞれが関係付けられる一つ以上のデータフローを受け取るステップ、並びに
プロセッサを用いて、一つ以上のデータフローを処理するステップであって、前記データフローそれぞれに対して、
前記データフローに関係付けられるグラフテンプレートのタイプを識別するステップ、
前記識別されたグラフテンプレートからランタイムグラフインスタンスを形成するステップであって、前記共有メモリセグメントにランタイムデータ構造を生成することを含み、前記ランタイムデータ構造がバッファセクションを含むステップ、
前記ランタイムグラフインスタンスの各頂点に対して、コンピュータシステムのプロセスの対応する群から、コンピュータシステムのプロセスを割り当てるステップであって、各割り当てられたプロセスが、当該プロセスのメモリ空間中に前記共有メモリセグメントをマッピングする、ステップ、および、
前記ランタイムグラフインスタンスにより、前記ランタイムデータ構造の前記バッファセクションに格納された前記データフローを処理するステップであって、前記割り当てたコンピュータシステムのプロセスを用いて、前記ランタイムグラフインスタンスの前記頂点と関係付けられる前記計算の一部に対応する前記計算を実行することを含むステップ、
を含むステップ、
を含む方法であって、
前記共有メモリセグメントは、前記コンピュータシステムによって提供されるシステムサービスを用いてアクセスされる、方法。
【請求項２】
前記プロセスの群はＵＮＩＸプロセスを含む、請求項１の方法。
【請求項３】
前記プロセスの群は事前に生成されたスレッドを含む、請求項１の方法。
【請求項４】
前記プロセスの群は、データベースへのアクセスを提供するための計算リソースであるデータベース接続を含む、請求項１の方法。
【請求項５】
前記二つ以上のグラフテンプレートを生成するステップは、揮発性メモリ内に前記テンプレートを格納することを含む、請求項１の方法。
【請求項６】
前記二つ以上のグラフテンプレートを生成するステップは、不揮発性メモリ内に前記テンプレートを格納することを含む、請求項１の方法。
【請求項７】
前記グラフテンプレートから前記ランタイムグラフインスタンスを形成するステップは、揮発性メモリ内に前記インスタンスを形成することを含む、請求項１の方法。
【請求項８】

前記ランタイムグラフインスタンスを形成するステップは、前記メモリの一部を前記ランタイムグラフインスタンスに割り当てるステップと前記メモリのこの部分に前記グラフテンプレートをコピーするステップとを含む、請求項７の方法。
【請求項９】
前記プロセスを割り当てるステップは、前記プロセスのそれぞれを、前記データフローを処理するために動的に割り当てることを含む、請求項１の方法。
【請求項１０】
前記プロセスのそれぞれを動的に割り当てるステップは、頂点に対する総ての入力のうちの少なくともある部分が利用可能であるときに生じる、請求項９の方法。
【請求項１１】
前記プロセスのそれぞれを動的に割り当てるステップは、頂点に対する総ての入力が利用可能であるときに生じる、請求項１０の方法。
【請求項１２】
プロセスを割り当てるステップは、前記データフロー上の総ての計算を処理するために前記頂点に対して前記プロセスのそれぞれを割り当てることを含む、請求項１の方法。
【請求項１３】
前記データフローの一つ以上を処理するステップは、頂点に割り当てられた前記プロセスを解放するステップと、前記グラフのインスタンスを削除するステップとを更に含む、請求項１の方法。
【請求項１４】
前記一つ以上のデータフローを処理するステップは、異なる計算グラフとそれぞれが関係付けられる少なくとも二つのデータフローを同時に処理することを含む、請求項１の方法。
【請求項１５】
前記異なる計算グラフそれぞれのインスタンスの少なくとも一つの頂点は、プロセスの同一の対応する群と関係付けられる、請求項１４の方法。
【請求項１６】
プロセスの前記同一の対応する群の少なくとも一つのプロセスは、前記異なる計算グラフの前記インスタンスの前記少なくとも一つの頂点に、異なる時点において割り当てられる、請求項１５の方法。
【請求項１７】
プロセスの一以上の群を形成するステップは、プロセスの群を少なくとも二つ形成するステップを含み、計算の第１の頂点は対応するプロセスの第１の群に関連付けられ、且つ、計算の第２の頂点は対応するプロセスの第２の群に関連付けられる請求項１の方法。
【請求項１８】
前記二つ以上のグラフテンプレートを生成するステップは、外部の格納手段に前記テンプレートを格納することを含む請求項１の方法。
【請求項１９】
前記グラフテンプレートからランタイムグラフインスタンスを形成するステップは、前記データ構造を共有メモリ内に形成することを含む請求項１の方法。
【請求項２０】
請求項１乃至請求項１９の何れか一項に記載の方法をコンピュータシステムに実行させる命令を格納するコンピュータ可読記憶媒体。
【請求項２１】
データ記憶システムに格納されたコンピュータプログラムコードと連動して作動する少なくとも一つのプロセッサ及び揮発性データデータ記憶装置内の共有メモリセグメントを用いるコンピュータシステム上で、計算を表したグラフを実行するシステムであって、
異なるタイプのグラフに基づく計算とそれぞれが関係付けられ且つデータ記憶装置に格納された二つ以上のグラフテンプレートを生成する手段であって、各テンプレートは、前記計算の対応する部分とそれぞれが関係付けられる幾つかの頂点を含む手段、

前記少なくとも一つのプロセッサ上で実行されるコンピュータシステムのプロセスの一つ以上の群を管理する手段であって、計算グラフの少なくとも一つの頂点がコンピュータシステムのプロセスの前記群の対応する一つと関係付けられている手段、

対応するグラフテンプレートにそれぞれが関係付けられる一つ以上のデータフローを受け取る手段、並びに

前記プロセッサを用いて、一つ以上のデータフローを処理する手段であって、それぞれの前記データフローに対して、

前記データフローに関係付けられるグラフテンプレートのタイプを識別し、

前記識別されたグラフテンプレートからランタイムグラフインスタンスを形成し、前記共有メモリセグメントにランタイムデータ構造を生成し、前記ランタイムデータ構造がバッファセクションを含み、前記ランタイムグラフインスタンスは前記グラフテンプレートの頂点に対応する頂点を有しており、

前記ランタイムグラフインスタンスの各頂点に対して、コンピュータシステムのプロセスの前記群の対応する一つから、コンピュータシステムのプロセスを割り当て、各割り当てられたプロセスが、当該プロセスのメモリ空間中に前記共有メモリセグメントをマッピングし、および、

前記ランタイムグラフインスタンスにより、前記ランタイムデータ構造の前記バッファセクションに格納された前記データフローを処理し、前記割り当てたコンピュータシステムのプロセスを用いて、前記ランタイムグラフインスタンスの前記頂点に関係付けられる前記計算の一部に対応する計算を実行する手段、

を含むシステムであって、

前記共有メモリセグメントは、前記コンピュータシステムによって提供されるシステムサービスを用いてアクセスされる、システム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、グラフに基づく計算の実行に関する。

【背景技術】

【０００２】

複雑な計算は、グラフの頂点（vertices，交点）に関係付けられる計算成分と、グラフのリンク（弧、辺）に対応する成分間のデータフローと、を伴う有向グラフ（directed graph）を通過するデータフローとして表わせることが多い。このようなグラフに基づく計算を実装するシステムは、米国特許第５，９６６，０７２号の「グラフとして表される計算の実行」に記載されている。グラフに基づく計算を実行するための一手法は、それぞれがグラフの異なる頂点（vertex）と関係付けられる幾つかのプロセスを実行し、且つ、そのグラフのリンクに応じてプロセス間の伝達経路（communication path，通信パス）を確立することである。例えば、伝達経路は、ＴＣＰ／ＩＰもしくはＵＮＩＸ（登録商標）ドメインソケットを用いるか、またはプロセス間でデータを渡すための共有メモリ（shared memory）を用いることができる。

【先行技術文献】

【特許文献】

【０００３】

【特許文献１】米国特許第５，９６６，０７２号公報

【発明の概要】

【０００４】

本発明の概略的な特徴において、グラフに基づく計算を効率的に起動するための方法は、計算グラフのランタイム構造を表すデータを事前に計算することを含む。即ち、要求されるタイプのグラフに対して事前に計算されたデータを用いて計算グラフのインスタンスを形成し、計算グラフのインスタンスに対するランタイムデータ構造を形成する。

【０００５】
本発明の別の概略的な特徴において、グラフに基づく計算を効率的に制御するための方法は、計算グラフの一つ以上の頂点と連携する計算を実行するためにそれぞれが適合しているプロセスの群を形成することを含む。プログラム実行時（at runtime）、頂点での処理のために入力が利用できるとき、これらのプロセスの群の構成要素（members）は計算グラフのインスタンスの特定の頂点へ動的に割り当てられる。
【０００６】
本発明の別の概略的な特徴は、効率的な起動とプロセスの群との組合せを含む。
【０００７】
概して、一態様において、本発明はグラフに基づく計算を処理するための方法を特徴とする。二つ以上のグラフテンプレートが生成される。各グラフテンプレートは異なるタイプの計算グラフと関係付けられる。各計算グラフは相当数のグラフ要素を含む。各グラフ要素は前記計算の対応する部分と関係付けられる。計算リソースの群が一つ以上形成される。計算グラフの各グラフ要素は計算リソースの群の対応する一つと関係付けられる。一つ以上のデータフローが処理される。それぞれのデータフローは、計算グラフの対応するタイプと結び付けられている。データフローのそれぞれに対し、データフローの処理は、対応する計算グラフに対するグラフテンプレートからグラフインスタンスを形成することを含んでいる。グラフインスタンスの各グラフ要素に対し、対応する群からのコンピュータのリソースは各グラフ要素に割り当てられる。各データフローはグラフインスタンスにより処理される。その処理には、前記割り当てられた計算リソースを用いて、かかるグラフインスタンスのグラフ要素に対応する計算を実行することが含まれる。
【０００８】
本発明の態様は、以下の特徴を一つ以上含む：
【０００９】
グラフ要素には、計算グラフの頂点またはリンクを含めることができる。
【００１０】
計算リソースには、プロセス、プロセススレッド（process thread）またはデータベース接続を含めることができる。
【００１１】
二つ以上のグラフテンプレートを生成することには、揮発性メモリまたは不揮発性メモリにテンプレートを格納することを含めることができる。
【００１２】
グラフインスタンスは、揮発性メモリ内のグラフテンプレートから形成され得る。
【００１３】
グラフインスタンスを形成することには、メモリの一部をそのグラフインスタンスに割り当て、グラフテンプレートをそのメモリの一部にコピーすることを含めることができる。
【００１４】
計算リソースは、データフローの計算の一部へ動的に割り当てることができる。
【００１５】
計算リソースは、計算に必要とされる総ての入力のうちの少なくともある部分の入力が利用できるとき、割り当てられ得る（can be assigned）。
【００１６】
計算リソースは、計算に必要とされる総ての入力が利用できるとき、割り当てられ得る。
【００１７】
各計算リソースの動的な割り当てには、グラフ要素からの計算リソース割り当て停止(deassign) を含めることができる。
【００１８】
グラフ要素の各計算リソースは、データフローの総てを処理するために割り当てられ得

る。
【００１９】
本方法は、グラフ要素に割り当てられた計算リソースを解放すること及びグラフのインスタンスを破棄することを更に含むことができる。
【００２０】
異なる計算グラフにそれぞれ関係付けられる少なくとも二つのデータフローを、同時に処理することができる。
【００２１】
異なる計算グラフのそれぞれのインスタンスの内の少なくとも一つのグラフ要素を、同じ計算リソースの対応する群と関係付けることができる。
【００２２】
計算リソースの同じ対応する群の内の少なくとも一つの計算リソースを、異なる時に、異なる計算グラフのインスタンスのグラフ要素へ割り当てることができる。
【００２３】
別の態様では、本発明は、概して、コンピュータ可読媒体に格納されてグラフに基づく計算を処理するためのソフトウエアを特徴とする。
【００２４】
別の態様では、本発明は、概して、グラフに基づく計算を処理するためのシステムを特徴とする。
【００２５】
本発明の態様には、以下の利点の内の一つ以上を含めることができる：
【００２６】
計算グラフのインスタンスを生成することによる計算オーバーヘッドは、グラフをインスタンス化（instantiate）する際に頂点間で別々の伝達経路を確立する場合に比べて、低下させることができる。
【００２７】
一実施の形態では、共有メモリの使用が、計算グラフの頂点計算を行うプロセス間でデータを渡すための効率的な伝達チャンネルを提供する。
【００２８】
プロセス群は、計算グラフの頂点と関連した計算を実行するためのプロセスの生成および初期化に伴うオーバーヘッドを低下させる。
【００２９】
動的に割り当てられたプロセスの群の使用は、他の場合には入力を待っているプロセスにより使用されるかもしれないリソースを減少する。
【図面の簡単な説明】
【００３０】
【図１】図１は、グラフに基づく計算のインスタンスを説明する線図である。
【図２】図２は、ワークフローを処理するためのシステムの論理ブロック図である。
【図３】図３は、グラフインスタンスのデータ構造の一実施の形態である。
【図４】図４は、図１に示す計算グラフに対するデータ構造である。
【図５】図５は、システム初期化のフロー図である。
【図６】図６は、各ワークフローを行うためのフロー図である。
【図７】図７は、計算グラフのインスタンス実行のフロー図である。
【図８】図８は、頂点処理を終えるためのフロー図である。
【発明を実施するための形態】
【００３１】
本発明のその他の特徴および利点は、以下の説明および特許請求の範囲から明らかである。
【００３２】
１．概説

以下に説明するシステムは、計算グラフで定義される計算を実行するための方法を実行する。図１を参照すると、計算グラフ１００の実施例は、一方向リンク１２０で結合された幾つかの頂点１１０を含む。図１に示す実施例においては、頂点１１０は１から６の番号が付され、リンク１２０も１から６の番号が付されている。計算グラフ１００は、例えば、取引処理システム（transaction processing system）と関係付けられた計算グラフに従い処理される個人取引（individual transaction）等の、一連のワーク要素（work element）１３０から成り立つワークフロー（work flow）を処理する。各頂点は、計算グラフ全体により定義される計算の一部と関係付けられる。本例では、頂点１は、一連の最初のワーク要素１３０に対する格納へのアクセスを提供し、その出力リンク１に一連のワーク要素を通過させる。各頂点に関係付けられる計算を実行するプロセスは、次に、ワーク要素１３０を処理し、典型的にはその頂点の一つ以上の出力リンク上にワーク要素を生じさせる。

【００３３】

図１に示すように、一つのワーク要素１３０はリンク１上を通過中であり、一つのワーク要素は頂点３での処理のための準備が整って待ち行列に入れられる（queued）二つのワーク要素は頂点４での処理のための準備が整って待ち行列に入れられる。従って、頂点３および頂点４のプロセスは、待ち行列に並ぶワーク要素を処理する実行準備ができている。図示のように、頂点５は、その一つの入力であるリンク４上に待ち行列に入れられるワーク要素を有するが、リンク５上にはワーク要素を有さない。従って、頂点５と関係付けられるプロセスは実行の準備が整っていない。

【００３４】

図２を参照すると、ワークフローを処理するためのシステム２００は、格納されたグラフデータ構造２１０を含んでいる。これらデータ構造は、グラフの頂点およびリンクの特性を含む計算グラフの仕様を含んでいる。そのシステムのグラフ実行・制御（「ＧＥＣ」）モジュール２２０は、格納されているグラフデータ構造２１０で規定された対応する計算グラフを用いて特定のワークフロー２３２を処理するコマンドを含む制御入力２２２を受信する。ＧＥＣモジュール２２０は、計算グラフの仕様を用いて、一般的に多数のプロセスで成り立ったグラフ計算処理２３０を制御する。グラフ計算処理２３０を実行するプロセスは、計算グラフの頂点に関係付けられる処理中にアクセスされるデータベースエンジン、データ記憶装置、または他のモジュール、を含む外部データおよびプロセス２４０を利用することができる。

【００３５】

通例、タイプが異なるワークフローは、タイプが異なる計算グラフ１００を用いて処理される。また、異なるワークフローは同時に処理されてもよく、それぞれのワークフローは一つのグラフの異なるインスタンスにより処理される。システム２００は、ＧＥＣモジュール２２０を介して、計算グラフのインスタンスに対するリソースの割り当てを行い、その実行を制御してワークフローを処理する。

【００３６】

２．グラフデータ構造

システム２００が含む幾つかの特徴は、限定されたリソースを効率的に共有するのは無論のこと、グラフ計算の起動を速やかにすることである。

【００３７】

計算グラフのインスタンスによるワークフローの処理を行う前に、ＧＥＣモジュール２２０は、機能的に共有されるメモリ内のグラフのインスタンスに対するランタイムデータ構造を生成する。一実施の形態では、グラフインスタンスの総てのランタイムデータ構造が内部で生成される単一の共有メモリセグメントが生成される。

【００３８】

プロセスは、ランタイム時にグラフの頂点と関係付けられ、これらプロセスのそれぞれは、共有メモリセグメントをアドレス空間にマッピングする。プロセスは、ワークフローの処理中、グラフインスタンスのランタイムデータ構造からワーク要素を読み込み、そし

てワーク要素をランタイムデータ構造へ書き込む。すなわち、グラフを流れるワーク要素のデータは、共有メモリセグメント内のこのランタイムデータ構造を通してプロセスからプロセスへと通過する。

【００３９】

グラフ計算処理２３０は、ＵＮＩＸ（登録商標）オペレーティングシステム等の適切なオペレーティングシステムの管理下で汎用コンピュータ上でホスティングされ得る。そのグラフの一つのインスタンスに対する共有メモリは、ランタイムグラフデータ構造を保持する共有メモリセグメントを、計算グラフを実行するプロセスのアドレス空間にマッピングするメモリマッピング機能を提供する標準のシステムサービス（例えば、ＵＮＩＸ（登録商標）システムサービスのｍｍａｐ（　）関数）を用いてアクセス可能であるのが好ましい。

【００４０】

図３は、計算グラフのインスタンスに対するランタイムグラフデータ構造３００の一実施の形態である。ヘッダーセクション３２０には、頂点の数３２２およびリンクの数３２４が含まれる。このランタイムグラフデータ構造３００には、異なる頂点にそれぞれ関係付けられる一連の記録３３２にあるグラフ頂点を規定する頂点セクション３３０も含まれる。ランタイム構造には、グラフの異なるリンクをそれぞれ規定するリンク記録３４２を含むリンクセクション３４０も含まれる。ランタイムグラフデータ構造３００には、ワーク要素が計算グラフの頂点間で渡され、頂点での処理の前に待ち行列に入れられるときにワーク要素データを保持するバッファセクション３５０も含まれる。

【００４１】

頂点セクション３３０において、各頂点記録３３２は、一般に、対応する頂点に対する入力リンク３３４およびその頂点に対する出力リンク３３５を特定するデータを含む。例えば、リンクおよび頂点は１から連続して番号が付され、ひとつの頂点に対する入力リンクおよび出力リンクのデータはこれらのリンクを含むインデックスのリストとして表されてもよい。

【００４２】

この実施例では、各頂点記録３３２には、待ち行列に入れられて処理を待っているワーク要素を有さない入力の数を示す入力カウント３３６に対する格納場所（storage）も含まれる。グラフ計算の実行中、この変数は、頂点に対する入力リンク数に初期化され、頂点に対する各入力に入力が利用できようになると数が一つ減らされ、入力待ち行列が空になると数が一つ増やされ、各入力に利用できる入力が存在しかつその頂点に対する処理の実行準備が整ったときにゼロになる。

【００４３】

この実施例では、各頂点記録３３２には、その頂点と関係付けられるプロセスの群（プロセス群の詳細な検討は本説明で後述する）を規定するプロセス群識別３３７も含まれる。識別された群内のプロセスを用いて頂点の処理を実行する。一般に、頂点に対する処理は、識別された群のいずれかの要素（member）により実行され得る。プロセスはワーク要素毎に動的に割り付けられるので、ある計算グラフの一つのインスタンスにおける同一の頂点により処理される異なったワーク要素は、識別された群の異なる要素により処理されてもよい。頂点記録３３２は、その頂点に対する特別な処理を実行するために識別された群のいずれかのプロセス要素を仕立てるのに使用される構成データ３３８をオプション的に含む。

【００４４】

ランタイムグラフデータ構造３００のリンクセクション３４０は、リンク記録３４２内においてグラフのリンクを規定する。各リンク記録３４２には、そのリンクに対するソース頂点３４６と、それらリンクに対する宛先頂点３４７と、を識別するデータが含まれていてもよい。各リンク記録には、ランタイムデータ構造を生み出すときに用いる構成データ３４５がオプション的に含まれる。

【００４５】

ランタイムグラフデータ構造３００のバッファセクション３５０には、グラフのリンクと関係付けられた別々のバッファ領域３５２が相当数含まれる。各リンク記録３４２には、そのリンク記録に対応するバッファ領域３５２を識別するバッファ位置３４４が含まれるのが典型的である。各バッファ領域には、対応するリンクの宛先頂点によって未だ処理されていないワーク要素３６２を多数保持するＦＩＦＯ方式待ち行列３６０が含まれるのが典型的である。

【００４６】

図４を参照すると、図１に示す計算グラフ１００の例に対応するランタイムグラフデータ構造３００は、ヘッダーに６個の頂点と６つのリンクがあることを示している。頂点セクション３４０は６つの頂点記録３３２を有する。例えば、頂点１に対する頂点記録３３２は、記載事項３３４において入力リンクがないことを示し、記載事項３３５においてリンク１が出力リンクであることを示す。頂点２に対する頂点記録３３２は、リンク１が入力リンクであり、リンク２および３が出力リンクであることを示す。図１に示すように、頂点２に対するワーク要素は、処理準備がまだ整わず待ち行列に入れられていないので、頂点記録の入力カウント３３６は、依然として一つの入力が満たされていないことを示す。

【００４７】

リンクセクション３５０には、計算グラフの各リンクにつき一つ、合計６つのリンク記録３４０が含まれる。例えば、リンク１のリンク記録３４２は、ソース頂点が頂点１であることを示す記載事項３４６、および、宛先頂点が頂点２であることを示す記載事項３４７を有する。リンク記録３４２の記載事項３４４は、バッファセクション３５０の対応するバッファ記録３５２にアクセスする方法を提供する（例えば、グラフデータ構造におけるバイトオフセットによる）。各バッファ記録３５２には、ＦＩＦＯ方式待ち行列３６０が含まれる。各ＦＩＦＯ方式待ち行列３６０は、多数のワーク要素３６２に対するデータを保持できる。図１に示す例に対応して、リンク２および４に対するＦＩＦＯ方式待ち行列３６０は一つのワーク要素を保持し、リンク３に対するＦＩＦＯ方式待ち行列３６０は２つのワーク要素を保持し、残りの待ち行列は空である。

【００４８】

図３を再度参照すると、ランタイムグラフデータ構造３００のある部分は、同一タイプのグラフのインスタンスの総てに共通なテンプレート３１０である。各頂点記録３３２の入力カウント３３６を除いて、ランタイムデータ構造の前記テンプレートの部分は変化しない。頂点に対する入力カウント３３６は、同一タイプのグラフのインスタンスの総てに対し共通の値、すなわち、頂点に対するどの入力も最初は満たされないことを示す同頂点の入力数に初期化される。

【００４９】

各タイプの計算グラフに対するテンプレート３１０は、そのタイプのグラフのインスタンスに対するランタイムデータ構造が必要となる前に事前計算される。次に、ランタイムインスタンスの生成には、本質的に、ランタイムデータ構造３００全体に対するメモリの割り付けとその割り付けられたメモリへの適切なテンプレート３１０のコピーとが含まれる（ＦＩＦＯ方式待ち行列３６０の構造に依存するが、バッファセクション３５０に何らかの最小限の初期化が要求されることもある）。

【００５０】

３．プロセス群

上記で紹介したように、頂点に対する計算は、プロセス群を用いて実行される。多数の異なるタイプの頂点の計算毎に、そのタイプの計算を要求する計算グラフを用いたワークフローの処理の開始に先立って、プロセスの群が生成される。グラフインスタンスによるワークフローの処理期間中、特別なタイプの計算がグラフ頂点に対する計算の実行に必要となる場合、プロセス群の要素はその頂点と動的に関係付けられ、そのワークフローの処理中その頂点と関係付けられたままとなる。一般に、異なるプロセス群が数多くあり、それぞれが、対応する処理のタイプと関係付けられている。ある群のプロセスを、別のタイ

プのグラフ内の頂点、あるタイプのグラフの別のインスタンス内の頂点、および、あるタイプのグラフの複数の異なる頂点に用いてもよい。

【００５１】

プロセス群の各プロセスは、プロセス群を管理するＧＥＣモジュール２２０が呼び出す別々のプロセス（例えばＵＮＩＸ（登録商標）プロセス）である。ＧＥＣモジュール２２０は、プロセス群毎に別々のワーク待ち行列を維持する。ワーク待ち行列の各エントリは、プロセスが計算を実行する対象であるグラフインスタンスの特定の頂点を識別する。

【００５２】

説明した実施の形態では、群のプロセスが最初に生成されるとき、それはグラフインスタンスに対する共有メモリセグメントをプロセスのアドレス空間にマッピングすることを含む初期化手順を実行する。初期化手順の完了後、その群に対するワーク待ち行列内の要素と関係のある処理を実行するようにＧＥＣモジュール２２０が合図するまで、プロセスは待機する。代替メカニズムを幾つかを用いてプロセスに合図できる。システムの一つのバージョンでは、ＧＥＣモジュール２２０が、そのモジュールと群の各プロセスとの間で制御情報を渡すための別々の制御チャンネルを維持する。各群プロセスは、群プロセスが実行すべきことを指し示す制御プロセスからの入力を待つ間「ブロック」する。

【００５３】

プロセス群の幾つかは、固定されたリソースを確保または消費するプロセスで成り立っている。このようなプロセス群の例は、Ｏｒａｃｌｅ（登録商標）データベース等のデータベースへの接続を生成するプロセスの多数のインスタンスで成り立っている。各データベースへの接続を形成及び維持するためにリソースが消費されるので、アクティブの状態にあるそのようなプロセスの数を制限することが望ましい。頂点で処理されるべきワーク要素が、データベースにアクセスするためのプロセスを要求する場合、（データベースとの接続を既に確立した）群のプロセスのうち一つがその頂点と関係付けられる。このようにして、そのデータベースに接続することが要求されたであろうプロセスの初期化ステップのような、プロセスを起動するオーバーヘッドが回避される。

【００５４】

システム２００は、頂点が群プロセスと関係付けられるときと、頂点の計算を開始するときとで異なる、頂点に対するプロセスを構成するための種々の手法をサポートする。あるタイプの構成においては、プロセスは、その総ての入力ワーク要素における総てのデータが完全に利用可能となるまで、頂点と関係付けされない。ワーク要素が大きい場合、上流の頂点によってワーク要素全体が計算されて利用可能となるのに幾らかの時間がかかることもある。このタイプの構成は、入力が利用可能となるのを待つプロセスがブロックされることを回避する。

【００５５】

別のタイプの構成は、ストリーミングモードを用いる。プロセスは、各入力の少なくとも始めの部分が利用可能となったときに、頂点と関係付けられ始動させられる。その入力それぞれの残りの部分は、そのプロセスを実行しているうちに利用可能となる。その入力が十分速やかに利用可能となる場合は、プロセスは入力を待ってブロックされたりはしない。しかしながら入力が利用可能とならない場合は、プロセスはブロックされることになろう。

【００５６】

別の形式の構成は、ディスクバッファモードを用い、このモードでは、ディスクまたは他の記憶装置にバッファリングされているデータの流れが明示的に識別される。例えば、上流プロセスはその出力をディスクに書き込み、下流プロセスはその入力全体がディスクから読み出し可能となったとき通知されるにすぎない。このようなデータの流れに対して、対応するＦＩＦＯ方式待ち行列３６０の記載事項３６２は、データをＦＩＦＯ方式待ち行列に直接保持するのではなく、ディスク上のデータの位置を識別する。この構成は、入出力に対するメモリ空間バッファを用いるのではなくディスクを用いるので、ＦＩＦＯ方式待ち行列を保持する共有メモリセグメントを温存する。

【００５７】
様々な度合いのプロセス群特異性もある。一形式の群は、特定のタイプのグラフの特定の頂点に非常に良く仕立てられたプロセスで成り立っている。別の形式の群では、プロセスはもっと一般的であり、多数の異なる頂点に適用可能である。このようなプロセスは、特定の頂点と関係付けられる時点でカスタマイズされる。例えば、群内のプロセスは、幾つかの異なる種類のデータ変換にとって一般的なデータ変換器プロセスから成り立っている。特定の頂点と関係付けられる構成データ３３８は、その特定の頂点に対する変換器を構成するために用いられる情報を提供する。さらに一般的なプロセス群では、各プロセスは、Ｊａｖａ（登録商標）仮想マシン（ＪＶＭ）等の仮想マシンを実現してもよく、頂点に対する構成データ３３８は、仮想マシンを用いて実行するプログラムを識別する。
【００５８】
プロセスが通知を受けて、その群に対するワーク待ち行列のエントリを処理する場合、プロセスはワーク要素を処理する前に任意の構成データ３３８に作用する。そのプロセスは、その処理が関係付けられる頂点を最初に識別し、次いで、共有メモリセグメント内の対応する頂点記録にアクセスして構成データ３３８を探すことにより、構成データにアクセスする。次に、プロセスは、その頂点の入力リンクに対するＦＩＦＯ方式待ち行列３６０内の処理するべきワーク要素３６２を見つける。そして、処理が完了すると、その頂点の出力リンクに対するＦＩＦＯ方式待ち行列にデータを書き込む。
【００５９】
４．計算制御
図５から８を参照すると、システム２００は、ＧＥＣモジュール２２０によって調整されるイベント駆動型制御手法を用いる。
【００６０】
図５は、システム初期化のフローチャートである。ＧＥＣモジュール２２０は最初に、プロセス群と、それらに関係付けられるワーク待ち行列とを生成する（ステップ５１０）。このステップの一部として、ＧＥＣモジュール２２０は、各プロセス群に対して別々のワーク待ち行列を生成する。次いで、ＧＥＣモジュール２２０は、ワークフローの処理に必要となる計算グラフの各タイプに対するアドレス空間にグラフテンプレート３１０を生成し、グラフインスタンスに対するランタイムデータ構造が生成されることになる共有メモリセグメントを生成する（ステップ５２０）。
【００６１】
図６は各ワークフローを処理するためのフローチャートである。ＧＥＣモジュール２２０がワークフローを処理するリクエストを受けた場合、そのワークフローの処理に必要な計算グラフの形式のグラフインスタンスを最初に生成する（ステップ６１０）。このプロセスの一部として、ＧＥＣモジュール２２０は、そのグラフインスタンスに対するランタイムデータ構造３００のために共有メモリセグメントの一部を割り当て、そのタイプの計算グラフに対するグラフテンプレート３１０をランタイムデータ構造３００にコピーし、それにより、ランタイムデータ構造を初期化する。次いで、ＧＥＣモジュール２２０は、下記のようにグラフインスタンスを実行する（ステップ６２０）。ワークフロー全体が処理されたとき、ＧＥＣモジュール２２０は、割り当てられたリソースを開放し、グラフインスタンスのランタイムデータ構造を削除することが好ましく、その結果、共有メモリセグメントのその部分を他のグラフのインスタンス用に再利用できる（ステップ６３０）。
【００６２】
図７は、計算グラフのインスタンスを実行するためのフローチャートである。グラフインスタンスの実行（図６、ステップ６２０を参照）は、どれかが入力カウントゼロを有する（これは、実行前に任意のフロー上の入力を必要としないことを示すのであるが）ように初期化されるか否かを決定するために、最初にグラフの頂点をスキャンする（ステップ７１０）。入力カウントゼロの頂点は、実行可能であり、且つ、関係付けられるそれらのプロセス群に対するワーク待ち行列に加えられる（ステップ７１２）。本例の最初の頂点はいかなる入力リンクも備えていないので、グラフインスタンスの実行が始まり、それが

ワーク待ち行列に加えられると、実行準備が整う。これら頂点に対する計算を実行するのに利用できるプロセス群内のプロセスがある場合（ステップ７２０）、ＧＥＣモジュール２２０はその計算リソースをグラフインスタンスに割り当て、直ちに計算を実行処理するよう通知し、そしてワーク待ち行列からエントリを取り出す（ステップ７３０）。群からプロセスが利用できない場合、後に群内のプロセスが別のグラフインスタンス内の頂点に対する計算の実行を完了して、利用可能となる時まで、最初に実行可能な頂点がワーク待ち行列に留まる。

【００６３】

頂点に対する計算を実行するプロセス群内のプロセスは、入力ワーク要素を入力リンクに対するＦＩＦＯ方式待ち行列３６０から取り出し、出力ワーク要素をその頂点の出力リンクに対するＦＩＦＯ方式待ち行列３６０に入れる。好ましくは、ＦＩＦＯ方式待ち行列からワーク要素を取り出したり入れたりすることは、実行可能な頂点に関係付けられる入力カウント３３６が値ゼロになるように、頂点の入力カウント３３６を維持する。ときとして入力が処理に利用できない場合、プロセスは、上流の頂点が入力を作成し、ＦＩＦＯ方式待ち行列に入れるまで、ブロックする。プロセスが頂点でワークフローに対する計算を完了すると、プロセスはＧＥＣモジュール２２０にプロセスが完了したことを通知する。ＧＥＣモジュール２２０は次いで、プロセスを、そのプロセス群に対するワーク待ち行列に入れられた別の頂点に割り当てることができる。

【００６４】

図８は、頂点に対する処理を完了させるフローチャートである。プロセスが処理を完了し、従って、プロセスを別の頂点に割り当てることが可能であることをプロセスがＧＥＣモジュール２２０に通知すると、ＧＥＣモジュール２２０は、まず、何れかのグラフインスタンスにおいて実行可能な頂点があるかどうかチェックする（ステップ８１０）。上記のように、実行可能な頂点は、それの入力リンクそれぞれが読み出し準備が整っているワーク要素を持つ頂点であり、それは、その頂点に対する頂点記録３３２の入力カウント３３６がゼロであることにより示される。実行可能な頂点は、対応するプロセス群の適切なワーク待ち行列に追加される（ステップ８２０）。ワーク待ち行列にある頂点に対する計算の実行に利用できる何らかのプロセスがある場合（ステップ８３０）、総てのこのようなプロセスは、実行可能な頂点に対する計算を実行するよう通知される（ステップ８４０）。

【００６５】

最後に、もはや実行可能な頂点が無くなったグラフインスタンスは、そのワークフローの処理を完了し、ＧＥＣモジュール２２０はそのグラフインスタンスの実行を完了し（ステップ８５０）、その結果、そのグラフインスタンスは削除される（図６、ステップ６３０を参照）。

【００６６】

上記計算制御は、多くの別のワークフロー手法をサポートする。例えば、一つのワークフローは単一の取引と関係付けられてもよく、頂点は、入力リンクそれぞれについて多くても一つのワーク要素を処理し、その出力にゼロまたは一個のワーク要素を生成してもよい。ワークフローは、例えば、取引のバッチ全体を処理するための、ワーク要素のストリームと関係付けることもできる。このような場合、各頂点は入力ストリームを処理し、それの入力の各セットについてゼロまたは一個の出力を生成する。

【００６７】

入力がまだ利用可能でないのとは対照的に、頂点がそれ以上の入力を受け取らないことを検出するために、各上流頂点はオプションとしてそれの下流リンクに明示的な終止符号（explicit terminator）を送る。これらの終止符号は、ワーク要素と同じ方法で待ち行列に入れられ、入力カウントに影響を与える。従って、頂点がその入力の各々に終止符号を有する場合、頂点は、その処理を終了させる前に、出力の各々に終了符号を出力する。この終了符号を用いれば、処理を行うプロセスは単一取引または取引のストリームを処理するように予め構成される必要がない。

【００６８】
５．代案
上記の計算制御手法においては、プロセス群からのプロセスがグラフインスタンスの頂点に割り当てられた後、プロセスがワークフローのワーク要素の処理を完了するまでプロセスは自由に実行できる。代替案は、例えば、作成した未処理の出力の量や、作成した入力の量に従って、または処理時間に従って、頂点での処理量を制限することである。
【００６９】
別の代替案は、頂点が実行可能となった時点から同頂点がワークフローの処理を完了するまで、プロセスを頂点に割り当てるという要件を緩めることである。ワークフローに対するワーク要素のストリームにおける連続するワーク要素を処理する間の状態を頂点が維持しなくてよければ、プロセスは幾つかのワーク要素を処理した後、例えば単一のワーク要素を処理した後、群へ戻され得る。従って、同一のグラフインスタンスにあっても、同一のプロセスは多数の異なる頂点の処理を実行するために使用され得る。ひとつのストリーム内のワーク要素を処理する間の状態をプロセスが維持しなければならないならば、このような状態は、頂点に対して別々に維持されることができ、プロセスが頂点に割り当てられるときにプロセスに再ロード（reload）され得る。
【００７０】
計算制御の多くの最適化が使用され得る。最初の最適化では、グラフインスタンスを介するワークフロー終止符号の伝搬は、入力の総てが終止符号である場合、プロセスを頂点と関係付けることを回避する。入力上の終止符号は待ち行列から取り出され、一つの終止符号が各出力上において待ち行列に入れられる。別の最適化では、頂点に対する処理を完了するプロセスは、下流の頂点を処理するために、そのプロセスが適切かどうかチェックし、次いで、そのプロセス自体をその下流の頂点に関係付けることができ、それによりプロセス群にそのプロセスを戻して頂点に再割り当てする必要性を回避することができる。
【００７１】
別の代替案として、プロセス群のプロセス数は、随意的に増減可能である。例えば、あるオプションでは、群のメンバの最小数が存在する。その群に対する要求次第で、群の追加要素が生成され、続いて、群プロセスが活動していなくなるにつれて、これらの要素を次第に除去する。別のオプションは、群における要素の数を決定するためのスケジュールを使用することである。例えば、一日の内の様々な時間において、もっと多くの要素に対するより大きな必要性があり得る。例えば、システムが活発な取引を処理している場合、特定のタイプの取引は、一日の内のある時間より、別の時間の方が発生頻度が高くなるかもしれない。
【００７２】
代替または追加として、プロセス群の要素として「重い」処理を用いると、別のタイプのプロセスから群を作り上げることができる。例えば、代替の群の一つのタイプは、全体として単一のＵＮＩＸ（登録商標）プロセスを群に用いることができるが、要素「プロセス」は、事前に生成されて実行準備が整っている軽いスレッドであってもよい。
【００７３】
別の代替案は、あるグラフ群を必要とすることになるであろうワークフローが存在することを予測して、既に例示された計算グラフのグラフ群を事前に生成しておくことである。ワークフローがグラフインスタンスを必要とするとき、グラフ群から利用できるものがあれば、生成される必要はなく、むしろ、その群からあてがわれる。このようにして、ワークフローの起動に要する時間はさらに減少せしめられる。そのワークフローのための計算が完了されたとき、変数がワークフローに割り当てられることに先立ち、変数が初期値に戻され（例えば、入力カウント３３６をリセットし）、動的に割り当てられるいずれのメモリも開放することにより、グラフはリセットされる。グラフはリセットされた後に群へ戻る。
【００７４】
プロセス群と同様に、グラフ群におけるグラフインスタンスの数は、必要に応じて増加

させることができる。例えば、グラフ毎に最低数のインスタンスしかないこともあり、必要に応じてより多くのインスタンスが生成されてもよい。

【００７５】

システムの代替バージョンにおいては、各プロセス群に対するワーク待ち行列は必要ない。例えば、群プロセスの要素が新しいタスクに取りかかる準備が整ったときには、いつでも、ＧＥＣモジュール２２０はグラフインスタンスの各々における総ての頂点の総てのインスタンスをスキャンして、そのワークを引き受ける適切なプロセスがあるかどうか調べることができる。他の代替法には、実行可能な頂点を識別するのに、ワーク待ち行列以外のデータ構造を用いることが含まれる。例えば、ハッシュテーブルを用いて実行可能な頂点を識別できる。

【００７６】

上記の説明では、オン・デマンド方式でグラフの頂点に群プロセスを割り当ててもよく、その場合、その頂点への入力の総てが利用可能となるまでは、群プロセスはその頂点と関係付けられない。別の手法は、ワークフローがグラフインスタンスと関係付けられるときにプロセスを頂点に関係付け、ワークフロー全体が処理されるまでその関係を維持することである。

【００７７】

上記説明のように、ランタイムデータ構造は、計算グラフの全体を定義する。システムの代替バージョンにおいて、上述した手法は、計算グラフの頂点間の伝達のためのより伝統的な手法と組み合わせることができる。例えば、別々のランタイムデータ構造をグラフの別のサブグラフと関係付けることができる。次いで、メモリを共有しないプロセッサ上で別のサブグラフを実行でき、異なるプロセッサ上の頂点間の伝達には、ソケット等の伝達手法を用いることができる。

【００７８】

上述した手法を他のグラフ仕様に拡張できる。例えば、各種のネストされた計算グラフのためのテンプレートからグラフインスタンスを組み合わせて作ることにより、計算グラフの階層仕様を実現することができる。

【００７９】

上述したように、ＧＥＣモジュール２２０は、作業メモリ内においてグラフテンプレートを計算して、これを格納する。代替案として、これらグラフテンプレートを磁気ディスク等の外部メモリに格納できる。別の代替案として、グラフテンプレートは、必ずしも、グラフインスタンスを形成するために再生されるメモリイメージである必要はない。例えば、グラフテンプレートには、対応するグラフインスタンスを形成するために用いられる圧縮表現またはシンボル表現を含めることができる。

【００８０】

一般に、例えば、使用するオペレーティングシステムに応じて、各種共有メモリの代替形式を用いることができる。

【００８１】

６．用途

上記タイプの計算グラフの一用途は、銀行業の用途における金融取引処理に対するものである。一般に、取引形式が異なると別のタイプの計算グラフが必要となる。典型的な計算グラフは、顧客取引のタイプと、取引の処理に必要な「バックエンド」サービスと、の組合せに関係付けられる。例えば、取引は、ＡＴＭの要求、銀行窓口入力およびコンピュータ間またはウェブサーバ間のビジネス間取引のこともある。特に、銀行が合併し、顧客が元の異なる銀行から成り立っている場合、異なる顧客は異なるバックエンドシステムを持つこともある。全員が承継銀行の顧客達であっても、その顧客達の口座は、大きく異なるバックエンドシステム上で維持されているかもしれない。従って、グラフの異なる頂点が異なる取引を処理するように使用され得る。異なるサービスがグラフの頂点と関係付けられてもよい。例えば、収支の更新、口座への預金または資金を口座に保有するよう口座保有を実行する等の機能と幾つかの頂点を関係付けてもよい。本発明によれば、頂点への

オン・ザ・フライのプロセス割り当てにより、未使用の活動していない頂点に対する処理に伴うオーバーヘッドが回避される。

【００８２】

７．実装

本発明は、ハードウエアまたはソフトウエア、もしくは両者の組合せ（例えば、プログラマブルロジックアレイ）で実装できる。特に明示しない限り、本発明の一部として含まれるアルゴリズムは、何らかの特定のコンピュータまたは他の装置と本質的には関連していない。特に、本明細書の教示に従って書かれたプログラムとともに各種の汎用マシンが使用され得る。或いは、より特化した装置（例えば、集積回路）を構築して特定の機能を実行すると更に便利である。従って、本発明は、プログラム化された又はプログラム可能な一以上のコンピュータシステム（分散型、クライアント／サーバ又はグリッド型等の各種のアーキテクチャでもよい）上で実行する一つ以上のコンピュータプログラムに実装してもよい。そのコンピュータシステムは、それぞれ、少なくとも一つのプロセッサ、少なくとも一つのデータ格納システム（揮発性および不揮発性メモリおよび／または記憶素子を含む）、少なくとも一つの入力装置またはポート、および少なくとも一つの出力装置またはポートを備える。プログラムコードを入力データに適用して、本明細書で説明した機能を実行し、出力情報を生成する。出力情報は、既知の方法で一台以上の出力装置に出力される。

【００８３】

このような各プログラムは、所望の任意のコンピュータ言語（機械語、アセンブリ言語、または高度プロシージャ、論理型、もしくはオブジェクト指向プログラム言語を含む）で実装され、コンピュータシステムと通信してもよい。いずれの場合も、言語は、コンパイル型またはインタプリタ型言語のどちらであってもよい。

【００８４】

このような各プログラムは、汎用または専用プログラマブルコンピュータで読取可能な記憶媒体または装置（例えば、固体メモリもしくは媒体、磁気もしくは光媒体）に格納またはダウンロードするのが好ましい。本明細書で説明した手順を実行するために記憶媒体または装置がコンピュータシステムにより読み取られるとき、コンピュータが構成され操作される。発明性のある本システムは、コンピュータプログラムで構成されたコンピュータ可読記憶媒体として実装されると考えてもよく、その場合、そのように構成された記憶媒体は、本明細書で説明した機能を実行するように、コンピュータシステムを特定かつ所定の方法で動作させる。

【００８５】

言うまでもなく、これまでの説明は、本発明の範囲を例示するものであって、本発明の範囲を限定する意図はなく、本発明の範囲は付帯する特許請求の範囲により定義される。その他の実施の形態は特許請求の範囲に含まれる。

【符号の説明】

【００８６】

１００　計算グラフ
１１０　頂点
１２０　リンク
１３０　ワーク要素
２００　システム
２１０　グラフデータ構造
２２０　グラフ実行・制御（ＧＥＣ）モジュール
２２２　制御入力
２３０　グラフ計算処理
２３２　ワークフロー
２４０　外部データおよびプロセス

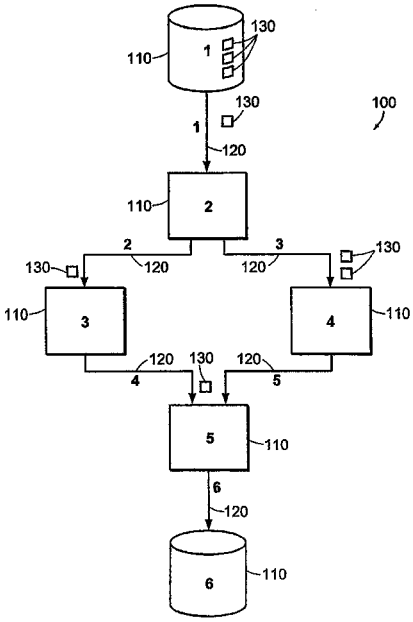
【００８７】

３００　ランタイムグラフデータ構造
３１０　テンプレート
３２０　ヘッダーセクション
３２２　頂点の数
３２４　リンクの数
３３０　頂点セクション
３３２　頂点記録
３３４　入力リンク
３３５　出力リンク
３３６　入力カウント
３３７　プロセス群識別
３３８　構成データ
３４０　リンクセクション
３４２　リンク記録
３４４　バッファ位置
３４５　構成データ
３４６　ソース頂点
３４７　宛先頂点
３５０　バッファセクション
３５２　バッファ領域
３６０　ＦＩＦＯ方式待ち行列
３６２　未処理ワーク要素

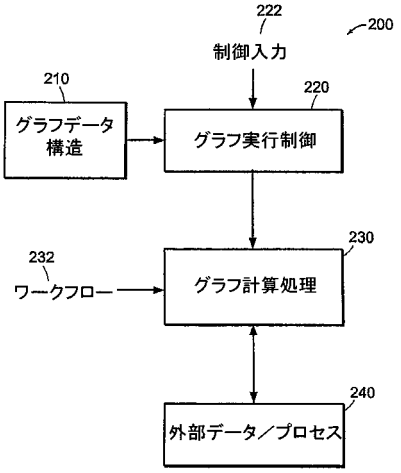
【００８８】

５１０　ステップ
５２０　ステップ
６１０　ステップ
６２０　ステップ
６３０　ステップ
７１０　ステップ
７１２　ステップ
７２０　ステップ
７３０　ステップ
８１０　ステップ
８２０　ステップ
８３０　ステップ

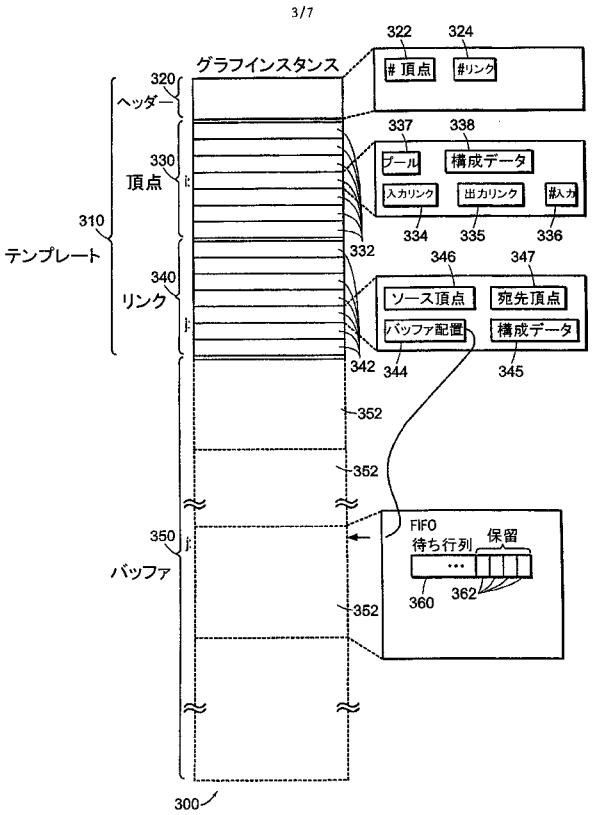
【図１】


100
110
120
130
1
2
3
4
5
6

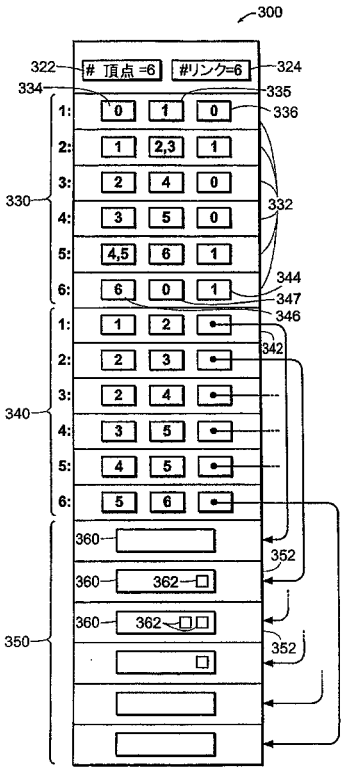
【図２】


200
210
グラフデータ構造
220
グラフ実行制御
222
制御入力
230
グラフ計算処理
232
ワークフロー
240
外部データ／プロセス

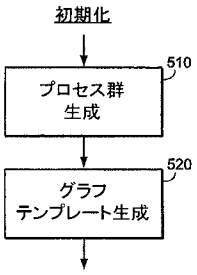
【図３】


3/7
グラフインスタンス
ヘッダ
頂点
リンク
テンプレート
バッファ
頂点
#リンク
プール
構成データ
入力リンク
出力リンク
ソース頂点
宛先頂点
バッファ配置
FIFO
待ち行列
保留
300
310
320
330
332
334
335
336
337
338
340
342
344
345
346
347
350
352
360
362

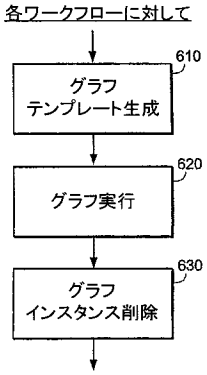
【図４】


300
頂点 =6
#リンク=6
322
324
330
332
334
335
336
340
342
344
346
347
350
352
360
362

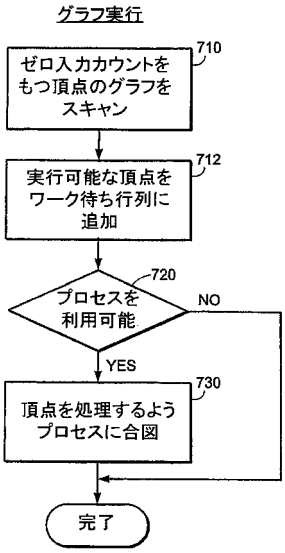
【図５】


初期化
510
プロセス群
生成
520
グラフ
テンプレート生成

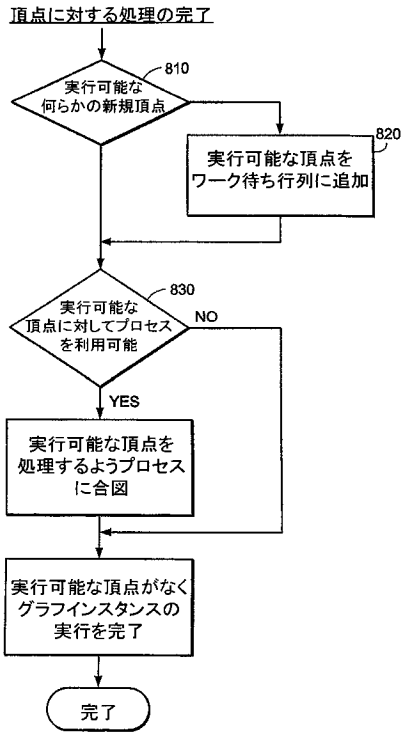
【図６】


各ワークフローに対して
610
グラフ
テンプレート生成
620
グラフ実行
630
グラフ
インスタンス削除

【図７】


グラフ実行
710
ゼロ入力カウントを
もつ頂点のグラフを
スキャン
712
実行可能な頂点を
ワーク待ち行列に
追加
720
プロセスを
利用可能
NO
YES
730
頂点を処理するよう
プロセスに合図
完了

【図８】


頂点に対する処理の完了
810
実行可能な
何らかの新規頂点
820
実行可能な頂点を
ワーク待ち行列に追加
830
実行可能な
頂点に対してプロセス
を利用可能
NO
YES
実行可能な頂点を
処理するようプロセス
に合図
実行可能な頂点がなく
グラフインスタンスの
実行を完了
完了