(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2010-508574
(P2010-508574A)
(43)公表日　平成22年3月18日(2010.3.18)

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

審査請求 有　予備審査請求 未請求　（全 24 頁）

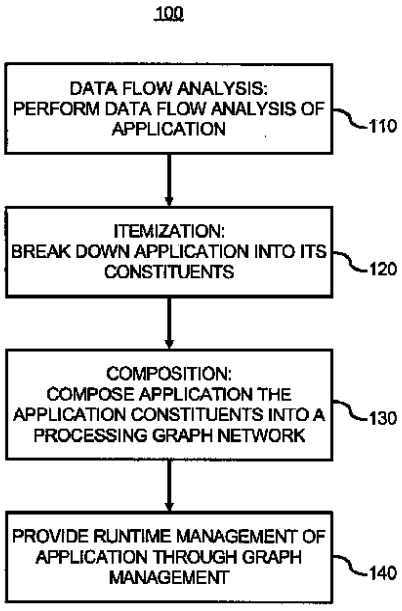
(21)出願番号　特願2009-534701 (P2009-534701)
(86)(22)出願日　平成19年10月30日 (2007.10.30)
(85)翻訳文提出日　平成21年6月24日 (2009.6.24)
(86)国際出願番号　PCT/US2007/022908
(87)国際公開番号　WO2008/054740
(87)国際公開日　平成20年5月8日 (2008.5.8)
(31)優先権主張番号　11/590,125
(32)優先日　平成18年10月31日 (2006.10.31)
(33)優先権主張国　米国 (US)

(71)出願人　503003854
ヒューレット－パッカード　デベロップメント　カンパニー　エル．ピー．
アメリカ合衆国　テキサス州　７７０７０　ヒューストン　コンパック　センタ　ドライブ　ウエスト　１１４４５
(74)代理人　110000039
特許業務法人アイ・ピー・エス
(72)発明者　タングアイ・ドナルド
アメリカ合衆国カリフォルニア州　サニーベール　アシュクロフトウェイ　１５４３
(72)発明者　ゲルブ・ダン
アメリカ合衆国カリフォルニア州　パロアルト　ページミルロード　１５０１

最終頁に続く

(54)【発明の名称】ミドルウェアフレームワーク

(57)【要約】
【課題】ミドルウェアフレームワークを提供する。
【解決手段】本発明に係る方法は、複数の処理ユニットを有するマルチプロセッシング環境で所望のアプリケーションを開発するためのミドルウェアフレームワークを提供する。本方法は、所望のアプリケーションを開発するための複数のタスクモジュールの選択を受け取ることと、選択されたタスクモジュール間の接続を受け取り、所望のアプリケーションを形成することと、形成されたアプリケーションを通じて処理するための複数の実行スレッドの入力を受け取ることと、少なくとも、ａ）複数の実行スレッドの少なくとも１つによる実行用の少なくとも１つのジョブのジョブリストを提供することと、ｂ）少なくとも１つの所定のポリシーに基づいて、複数の実行スレッドの１つによるジョブリストの各ジョブの実行を自動的にスケジューリングすることとによる、自動グローバルスケジューリングを提供することとを含む。
【選択図】図１Ａ



【特許請求の範囲】
【請求項１】
複数の処理ユニットを有するマルチプロセッシング環境において所望のアプリケーションを開発するためのミドルウェアフレームワークを提供する方法であって、
前記所望のアプリケーションを開発するための複数のタスクモジュールの選択を受け取ること（１３０）と、
前記選択されたタスクモジュール間の接続を受け取ること（１３０）であって、前記所望のアプリケーションを形成する、前記選択されたタスクモジュール間の接続を受け取ること（１３０）と、
前記形成されたアプリケーションを通じて処理するための複数の実行スレッドの入力を受け取ること（１３０）と、
前記複数の実行スレッドの前記ミドルウェアフレームワーク全体にわたる自動グローバルスケジューリングを提供すること（１４０）であって、少なくとも、
前記複数の実行スレッドの少なくとも１つによる実行用の少なくとも１つのジョブのジョブリストを提供すること（１４１）であって、前記少なくとも１つのジョブのそれぞれは、前記選択されたタスクモジュールの関連付けられるものによる１つまたは複数のデータオブジェクトの処理である、少なくとも１つのジョブのジョブリストを提供すること（１４１）と、
少なくとも１つの所定のポリシーに基づいて、前記複数の実行スレッドの１つによる前記ジョブリストの各ジョブの実行を自動的にスケジューリングすること（１４２）と
による自動グローバルスケジューリングを提供すること（１４０）と
を含む方法。
【請求項２】
前記形成されたアプリケーションのグラフネットワーク表現を表示すること（１４０）であって、前記選択されたタスクモジュール、前記タスクモジュール間の前記受け取った接続、および、前記形成されたアプリケーションのスループット統計量およびレイテンシの一方を示す、前記形成されたアプリケーションのグラフネットワーク表現を表示すること（１４０）
をさらに含む請求項１に記載の方法。
【請求項３】
前記複数の実行スレッドの１つによる実行用にスケジューリングされた前記少なくとも１つのジョブは、複数のジョブを含み（１４２）、
前記方法は、前記スケジューリングに基づいて、前記１つの実行スレッドが、前記選択されたタスクモジュールの少なくとも２つにおいて、および、前記複数の処理ユニットの少なくとも２つにわたり、前記複数のジョブを自動的に実行すること
をさらに含む請求項１に記載の方法。
【請求項４】
前記少なくとも１つの所定のポリシーは、前記ジョブのそれぞれに関連付けられる前記１つまたは複数のデータオブジェクトのそれぞれで見つけられる優先順位表示に基づいている
請求項１に記載の方法。
【請求項５】
前記データオブジェクトのそれぞれの前記優先順位表示は、
ａ）前記データオブジェクトのそれぞれのタイムスタンプ、および、
ｂ）前記データオブジェクトのそれぞれがその子孫である最も初期のデータオブジェクトのタイムスタンプ
の一方を含む請求項４に記載の方法。
【請求項６】
前記少なくとも１つの所定のポリシーは、
ａ）前記ジョブリストに実行用にスケジューリングされるジョブに関連付けられる前記

選択されたタスクモジュールの１つのタスクのタイプ、
ｂ）前記複数の実行スレッドの１つを実行している前記複数の処理ユニットの１つの識別情報、
ｃ）前記ジョブリストのジョブを最後に遂行した前記選択されたタスクモジュールの１つの識別情報、および
ｄ）前記ジョブリストのジョブが、遂行される前記ジョブに望ましい前記データオブジェクトの利用可能な１つまたは複数を有するとの判断
の１つに基づいている
請求項１に記載の方法。
【請求項７】
前記少なくとも１つの所定のポリシーは、前記選択されたタスクモジュールのうち他のいくつが、前記ジョブのそれぞれに関連付けられている前記選択されたタスクモジュールの出力に依存しているのかに基づいている
請求項１に記載の方法。
【請求項８】
前記ジョブリストを提供することは、
ａ）前記選択されたタスクモジュールの１つが、処理用に少なくとも１つのデータオブジェクトを受け取ることと、
ｂ）前記選択されたタスクモジュールの１つが、少なくとも１つのデータオブジェクトの生成を望むソースモジュールであることと
の１つに応答して、前記ジョブリストに各ジョブを動的に生成することと
を含む請求項１に記載の方法。
【請求項９】
複数の処理ユニットを有するマルチプロセッシングプラットフォーム上で所望のアプリケーションを開発するための、コンピュータ可読媒体のプログラムコードとしてコード化されるミドルウェアフレームワークであって、
前記所望のアプリケーションを構築し動作させるためのタスクモジュール及びメディアオブジェクトを生成する、前記コンピュータ可読媒体のプログラムコードとしてコード化されるフレームワークカーネル（５３０）であって、
前記フレームワークカーネルは、
グローバルスケジューラであって、前記フレームワークカーネルが、前記ミドルウェアフレームワーク全体にわたる複数の実行スレッドの自動グローバルスケジューリングを提供して、前記グローバルスケジューラによって保持されるジョブのリストおよび少なくとも１つの所定のポリシーに基づき前記生成されたタスクモジュールを通じて前記生成されたメディアオブジェクトを処理するために、前記プログラムコードの一部としてコード化され、前記ジョブのそれぞれは、前記生成されたタスクモジュールの関連付けられる１つによる１つまたは複数のデータオブジェクトの処理である、グローバルスケジューラ
を含むフレームワークカーネル（５３０）と、
前記フレームワークカーネルを前記マルチプロセッシングプラットフォームから隔離して前記フレームワークカーネルをプラットフォームから独立した状態にする、前記コンピュータ可読媒体のプログラムコードとしてコード化される抽象レイヤ（５４０）と
を備えるミドルウェアフレームワーク。
【請求項１０】
複数の処理ユニットを有するマルチプロセッシング環境において所望のアプリケーションを構築するためのミドルウェアフレームワークを提供するためのプログラムコードがコード化されるコンピュータ可読媒体であって、
前記所望のアプリケーションを構築するための複数のタスクモジュールの選択を受け取る（１３０）ためのプログラムコードと、
前記選択されたタスクモジュール間の接続を受け取る（１３０）ためのプログラムコードであって、前記所望のアプリケーションを形成する、前記選択されたタスクモジュール

間の接続を受け取る（１３０）ためのプログラムコードと、
前記形成されたアプリケーションを通じて処理するための複数の実行スレッドの入力を受け取る（１３０）ためのプログラムコードと、
前記複数の実行スレッドの前記ミドルウェアフレームワーク全体にわたる自動グローバルスケジューリングを提供する（１４０）ためのプログラムコードであって、少なくとも、
前記複数の実行スレッドの少なくとも１つによる実行用の少なくとも１つのジョブのジョブリストを提供する（１４１）ためのプログラムコードであって、前記少なくとも１つのジョブのそれぞれは、前記選択されたタスクモジュールの関連付けられるものによる１つまたは複数のデータオブジェクトの処理である、少なくとも１つのジョブのジョブリストを提供する（１４１）ためのプログラムコードと、
少なくとも１つの所定のポリシーに基づいて、前記複数の実行スレッドの１つによる前記ジョブリストの各ジョブの実行を自動的にスケジューリングする（１４２）ためのプログラムコードと、
を有することによる、自動グローバルスケジューリングを提供する（１４０）ためのプログラムコードと
を含むコンピュータ可読媒体。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明の実施の形態は、ミドルウェアフレームワークに関する。

【背景技術】
【０００２】
リアルタイムストリーミングマルチメディアアプリケーションは、性能及び応答性を維持すると同時に、複数のデータストリームの処理を必要とすることから、このようなアプリケーション用のロバストシステムの構築は困難である。
したがって、アプリケーション開発者は、１）システムの複雑性の隔離及び管理、２）マルチメディアアプリケーションの複数のデータフォーマットに対する同時実行のサポート、３）データストリーミングオペレーションのためのデータのシーケンスの処理、並びに４）リアルタイムアプリケーションの変動する負荷のもとでの可変強度プラットフォーム（variable-strength platform）上での応答性能の達成から成る少なくとも４つのタイプの課題を克服しなければならない。

【発明の概要】
【０００３】
実施形態が、以下の図（複数可）に、限定ではなく例として示される。
図において、同様の符号は、同様の要素を示す。

【図面の簡単な説明】
【０００４】
【図１Ａ】本発明の一実施形態によるソフトウェア開発の方法論を示す。
【図１Ｂ】本発明の一実施形態によるグローバルスケジューラのオペレーションを示す。
【図２】本発明の一実施形態によるアプリケーションのデータフロー解析のプロセスフローを示す。
【図３Ａ】本発明の一実施形態によるミドルウェアフレームワークにおけるアプリケーションの分解を示す。
【図３Ｂ】本発明の一実施形態によるミドルウェアフレームワークにおけるアプリケーションの組み立てを示す。
【図３Ｃ】本発明の一実施形態によるミドルウェアフレームワークにおけるアプリケーションのランタイムの管理を示す。
【図４】本発明の一実施形態による、データフローミドルウェアフレームワークを使用して所望のアプリケーションを構築する一例を示す。

【図５】本発明の一実施形態によるデータフローミドルウェアフレームワークの実施階層を示す。
【図６】本発明の一実施形態によるデータフローミドルウェアフレームワークを実施するためのコンピュータ化システム６００のブロック図を示す。
【発明を実施するための形態】
【０００５】
簡単にするために且つ例示を目的として、実施形態の例を主に参照することによって、実施形態の原理が説明される。
以下の説明では、実施形態の徹底した理解を提供するために、多数の具体的な詳細が述べられる。
しかしながら、これらの具体的な詳細に限定されることなく実施形態を実施することができることが、当業者には明らかであろう。
それ以外の場合には、実施形態を不必要に不明瞭にしないように、周知の方法及び構造は詳細に説明されていない。
【０００６】
リアルタイムのマルチメディアアプリケーション又は他の複雑なアプリケーションの開発を、オペレーティングシステムの依存関係を抽象化し、頻繁に使用されるコンポーネントの最適化された実施を提供するミドルウェアフレームワークの使用によって大きく加速することができる。
したがって、本明細書では、このようなミドルウェアフレームワークの方法及びシステムが説明される。
本発明の一実施形態では、ソフトウェアの設計及び構築を簡単にしてソフトウェア開発時間を減少させることにより、マルチメディアアプリケーション等の複雑なアプリケーションのソフトウェア設計を改良するように作動可能なマルチプラットフォームソフトウェアフレームワークであるデータフローミドルウェア（ＤＭ）フレームワークが提供される。
さらに、このミドルウェアフレームワークは、リアルタイム又はオフラインのいずれにおいてもランタイム中の複雑なオペレーションを効率的にサポートするように作動可能である。
【０００７】
ＤＭフレームワークの使用に対する従来のほとんどの解決法は、シングルスレッド又はスレッド毎モジュール（thread-per-module）であり、メディアパイプラインのマルチスレッド実行用のグローバルスケジューラを使用することはない。
シングルスレッドの解決法では、従来のフレームワークは、モジュール性の利益を達成するが、モジュールの実行を並列に行うことができないので、性能は低い。
スレッド毎モジュール解決法では、アプリケーションは、並列実行を使用する。
しかしながら、アプリケーションモジュールは、オーバーフローの状況及びスタベーション（starvation）の状況に個別に反応しなければならず、メディアのドロップ又はモジュールの動作速度の調整をいつ行うのかをローカルに決定しなければならない。
したがって、本発明の少なくとも１つの実施形態は、アプリケーションの性能を犠牲にすることなく、アプリケーションソフトウェアの簡単化したモジュール形式のデータフロースタイル設計を提供しようとするものである。
データフロー設計では、アプリケーションは、有向アークによって互いにリンクされた機能モジュールの接続ネットワークである。
データフロー設計は、モジュール性が複雑性を低減し、アークがデータのストリームを表し、アークが複数のデータフォーマットを送信することができるので、ストリーミングマルチメディアアプリケーション等の複雑なアプリケーションを表すのによく適している。
本発明の別の実施形態では、マルチコアプロセッサ内に又は複数のマルチコアプロセッサ全体にわたって複数のプロセッサを有するマルチプロセッシング環境全体にわたりアプ

リケーションタスクの並列実行を自動化又は組織化するグローバルスケジューラを含むミドルウェアフレームワークが提供される。
本明細書で参照されるように、処理ユニットは、単一のプロセッサ又はマルチコアプロセッサの１つのコアである。
したがって、複数のプロセッサ、マルチコアプロセッサ、又は複数のマルチコアプロセッサを有する環境は、複数の処理ユニットを有することになる。
【０００８】
アプリケーション開発者が直面することが多い上述した課題のうち最初の３つのものは、特に現代のオブジェクト指向型プログラミング言語の助けを借りることによって克服することが可能であるが、リアルタイム処理の第４の課題は、はるかに困難である。
したがって、どのアプリケーションも、オーバーパワーマシン（over-powered machine）上では常に応答するが（たとえば、サーバクラスのマシンを使用するウェブカメラビデオキャプチャ）、本発明の１つ又は複数の実施形態は、マシンの資源が限られていても、マシンのマルチプロセッシング能力を利用してアプリケーションの性能を達成しようとする。
【０００９】
本発明の別の実施形態によれば、アルゴリズム（たとえば、ビデオ処理又は解析）をランタイムシステム（たとえば、マルチスレッド化、同期）から隔離することにより、ＤＭフレームワークが、アプリケーションソフトウェアを設計してコーディングするのに用いられる。
これによって、アプリケーション開発者は、手元のアプリケーションに特有のアルゴリズム処理に集中することが可能になると同時に、そのフレームワークを利用して、アプリケーション開発者が直面することが多い上述した課題を克服することが可能になる。
加えて、このようなＤＭフレームワークは、（メンテナンスを簡単にする）改良された記述性及び可読性、他者の作業を利用するコード再利用、デバッグを簡単にし、且つソフトウェアロバスト性を確保するより良い試験方法論、及び他のプラットフォームへの増大された移植性のような他のソフトウェアエンジニアリングの利益も提供する。
【００１０】
方法論
データフローパラダイムに基づく本発明の実施形態では、アプリケーションからのデータは、ＤＭフレームワークの計算モジュールの有向グラフをフローする。
したがって、このパラダイムでアプリケーションを作成、構築、又は開発するために、次のフェーズ、すなわち、（１）信号及びそれらの信号に対する処理フェーズを決定するアプリケーションのデータフロー解析、（２）アプリケーションの、メディア表現及び処理モジュールへの分解、（３）モジュールの有向グラフネットワークへの組み立て、並びに（４）アプリケーショングラフネットワークのランタイム管理を含むソフトウェア開発の方法論が採用される。
図１Ａは、上述した方法論１００を示している。この方法論１００は、ＤＭフレームワークがこの方法論を援助するためにどのように作動可能であるのかに関して以下でさらに詳細に説明される。
【００１１】
１１０において、ストリーミングメディアアプリケーション等、構築又は開発される対象アプリケーションのデータフロー解析が行われる。
対象アプリケーションは、そのプロトタイプであってもよいし、テストフェーズであってもよく、アプリケーション開発者は、対象アプリケーションを完成させるために、さらなる修正又は高度化のためのアプリケーションのさらなる解析を望んでいる。
図２は、フェーズ１１０の詳細を示している。フェーズ１１０は、アプリケーション開発者が行うことができる。任意のアプリケーションの基本的な情報コンテンツは、時間と共に進展するデータ信号（たとえば、オーディオ又はビデオ）である。
したがって、２１０において、アプリケーションのデータフロー解析を行うために、ア

プリケーション開発者は、最初に、アプリケーションに供給を行う１つ又は複数の信号ソース（たとえば、マイクロホン、カメラ、又はファイル）を識別する。
次に、２２０において、アプリケーション開発者は、各信号が、信号ソースにおいて発生してからアプリケーションを通って進行する時に、各信号の所望の変換パスをたどる。
この変換パスに沿った各識別可能な信号フォーマット間で、信号は、異なった処理フェーズにかけられる。
たとえば、オーディオ信号は、マイクロホンソースにおいてＰＣＭフォーマットで発生し、次に、ＡＤＰＣＭへの変化を受け、次に、ＵＤＰパケットへの変換を受ける場合がある。
この例では、圧縮ステージが、ＰＣＭフォーマットとＡＤＰＣＭフォーマットとの間に位置し、ネットワークパケット化ステージが、ＡＤＰＣＭフォーマットとＵＤＰフォーマットとの間に位置する。
信号の変換は、信号のフォーマットを変更しない処理ステージ中に行われる場合もある。
たとえば、色補正ステージは、その画像入力と同じフォーマットを有するが、画像の内部に修正されたデータを有する画像出力を生成する場合がある。
したがって、２３０において、各信号をこの方法で解析することによって、アプリケーション開発者は、アプリケーションの信号フォーマット及び異なる処理フェーズの双方を識別することができる。

【００１２】

図１Ａを再び参照して、方法論１００の次のフェーズは、１２０におけるアプリケーション項目別分類である。
このアプリケーション項目別分類では、アプリケーション開発者は、データフロー解析に基づいて、構築されるアプリケーションをその構成要素又はコンポーネントにブレークダウンする。
これよりも先に識別された信号フォーマットは、メディアタイプであり、これよりも先に識別された処理フェーズは、それらのメディアタイプを処理する。
一実施形態では、ＤＭフレームワークは、この項目別分類フェーズをサポートする３つの抽象化、すなわち、メディアオブジェクト、タスクオブジェクト、及びジョブを提供する。
本明細書で参照されるように、メディアオブジェクトは、基本データ単位であり、各単位は、特定のメディアタイプを有する。
各メディアオブジェクトは、マルチメディアアプリケーションにおけるストリームベースの信号等、任意のタイプのデータ信号とすることができる。
ストリームベースの信号の例には、オーディオストリーム、ビデオストリーム、及び一連の画像のそれぞれにおける顔の２次元（２Ｄ）座標のストリームが含まれるが、これらに限定されるものではない。
これとは対照的に、本明細書で参照されるように、タスクオブジェクトは、基本処理単位である。
各タスクオブジェクトは、１つ若しくは複数のメディアオブジェクトを受け取るための入力をゼロ個以上、１つ若しくは複数のメディアオブジェクトを１つ若しくは複数の他のタスクオブジェクトへ送出するための出力をゼロ個以上、又は１つ若しくは複数のメディアオブジェクトを受け取り且つ送るための入力若しくは出力を少なくとも１つ有する。
少なくとも１つの出力を有するが入力を有しないタスクオブジェクトは、後に説明するプロダクションタスクモジュール等のメディアオブジェクト（複数可）のソースとして機能するソースタスクオブジェクトである。
少なくとも１つの入力を有するが出力を有しないタスクオブジェクトは、任意の入力メディアオブジェクトの終端点として機能するシンクタスクオブジェクトである。
たとえば、シンクタスクオブジェクトは、メディアオブジェクトをフレームワークの他のタスクオブジェクトへ送出しないので、出力を有しない。

その代わり、ファイルシンクタスクオブジェクト等のシンクタスクオブジェクトは、結果としてのメディアオブジェクト（複数可）をストレージ媒体に書き込むこともできるし、ネットワークタスクオブジェクト等のシンクタスクオブジェクトは、ネットワーク全体にわたって結果としてのメディアオブジェクト（複数可）を送信することもできる。

タスクオブジェクトは、ビデオ圧縮、ビデオ伸張、顔認識等の任意のタイプのメディア計算とすることができる。

タスクオブジェクトは、カメラからの画像取り込み、又はコンピュータサウンドカード及びスピーカによるオーディオ再生等のＩ／Ｏプロセスによるメディアの生成又は消費も含むことができる。

また、本明細書で参照されるように、ジョブは、このようなジョブに関連付けられるタスクオブジェクトによる１つ又は複数の必要なメディアオブジェクト（複数可）の処理である。

したがって、１つ又は複数のジョブを特定のタスク及び特定のメディアオブジェクト（複数可）に関連付けることができる。

さらに、複数のジョブを単一のタスクオブジェクトにより、タスクオブジェクトのタイプに応じて経時的にシーケンシャルに又は同時に処理することができる。

【００１３】

一意の信号フォーマットごとに、アプリケーション開発者は、タイムスタンプ記録、メモリ管理、及び自動シリアライズ（automatic serialization）のような振る舞いをオブジェクト指向型プログラミングのＭｅｄｉａ（メディア）基底クラスから継承するメディアオブジェクトの個別のメディアタイプを定義することができる。

同様に、処理フェーズごとに、開発者は、ＤＭフレームワークですでに利用可能な所定のタスクモジュールを使用することもできるし、ＤＭフレームワークにおけるオブジェクト指向型プログラミングの所定のＴａｓｋ（タスク）基底クラスの振る舞いを継承する新しいタスクモジュールを定義することもできる。

したがって、各タスクモジュールは、０個以上の入力ピン、０個以上の出力ピン、又は対応するタスクオブジェクトの入力（複数可）及び出力（複数可）に対応する少なくとも１つの入力ピン若しくは少なくとも１つの出力ピンを有する。

タスクモジュールの継承可能な振る舞いには、入出力（Ｉ／Ｏ）バッファ管理並びにマルチスレッド実行及びマルチスレッド同期が含まれる。

各タスクモジュールの内部のコードは、入力から出力へのアルゴリズム的マッピング（algorithmic mapping）であり、ＤＭフレームワークを単に使用することから可能になるスレッド化問題又は同期問題からは隔離されている。

【００１４】

図１Ａを再び参照して、ここで、メディアオブジェクト及びタスクオブジェクト並びに関連付けられるジョブはアプリケーション構築ブロックである。

したがって、１３０において、アプリケーション開発者は、アプリケーションの構成要素又はコンポーネントの組み立てを通じて、構築されるアプリケーションを形成することができる。

ここで、アプリケーション開発者は、多くのタスクモジュール（それぞれはタスクを表す）を互いに接続して、処理グラフネットワークを形成し、ＤＭフレームワークに、アプリケーション又は処理グラフネットワークの１つ又は複数のジョブの実行又は遂行の１つ又は複数のフレームワークスレッド（以下、「実行スレッド」としても参照する）を要求する。

各接続は、特定のメディアタイプの一方向転送であり、メディアストリームを表す。

すべてのタスクピンが接続されているという条件で、アプリケーションは、メディアオブジェクトを作成するプロダクションタスクをトリガすることによって処理グラフネットワークを開始することができる。

メディアオブジェクトの各プロダクションの後、プロダクションタスクは、自身をトリガして、次のメディアオブジェクトを作成することができる。

各メディアオブジェクトは、メディアプロダクションタスクモジュールから処理ネットワーク又はグラフの残りの部分へフローし、さらに、実行スレッドに基づいて、アプリケーションにおけるタスクの残りの部分の消費、プロダクション、又は処理振る舞いをトリガする。

【００１５】

一実施形態によれば、ＤＭフレームワークは、メディアオブジェクト内のメディアバッファの再利用を最適化する内部メモリマネージャも含む。後に、後述するようにＤＭフレームワークの一部であるグラフィカルディスプレイプログラムが、グラフに関連付けられるジョブの実行をフレームワークスレッドにホルト（halt）させる停止コマンドを発行することができる。

いくつかの実施形態では、停止コマンドは、ジョブ実行のホルトが効力を有する前に、現在スケジューリングされているすべてのジョブを実行させる。

グラフを停止することによって、タスクモジュールの内部データ状態が保持される。

動的アプリケーションでは、まずグラフを停止し、次に、タスクモジュールを追加又は除去し、次に、新しいグラフ及び除去されなかった任意のタスクモジュールの内部データ状態でアプリケーションを継続する開始コマンドを発行することによって、タスクをグラフに追加し又はタスクをグラフから除去することができる。

代替的に、グラフィカルディスプレイプログラムは、破棄コマンドを発行することもできる。

この破棄コマンドは、グラフ内のすべてのＴａｓｋを再帰的に破棄する。

ＤＭフレームワークの上述したコマンドは、グラフィカルディスプレイプログラムを通じて利用可能とすることができるフレームワークコマンドに関して説明されているが、フレームワークコマンドは、グラフィカルディスプレイプログラムの外部で利用可能なメカニズム又はコマンドを通じて、ＤＭフレームワーク内で自動的に又はＤＭフレームワークへのユーザ入力によって手動で、ＤＭフレームワークに発行することができることが理解されるべきである。

【００１６】

他の多くのアーキテクチャとは異なり、ＤＭフレームワークは、閉路を含む任意のグラフトポロジーをサポートする。

閉路は、フィードバックループを有するあらゆるアプリケーションで重要である。たとえば、ディスプレイタスクモジュールからのマウスの動きによって、別のタスクモジュールにおける新規なビュー合成の視点を求めることができ、この別のタスクは、次に、新しい画像をディスプレイタスクモジュールへ送信することができる。

メディアストリームのタイプについて一致するために、２つの接続されたタスクモジュールは、メディアタイプを交渉しなければならない場合がある。

たとえば、一般化されたＵＤＰ　Ｔａｓｋは、任意のメディアタイプを許容することができるが、そのメディアタイプを供給するビデオソースは、ＭＰＥＧ－４ビデオしか配信しない場合がある。

ＵＤＰ　Ｔａｓｋは柔軟性を有するので、２つのタスクモジュールは、単に、ＭＰＥＧ－４ビデオメディアタイプを送信／受信することに同意するだけである。

結局、完成したグラフ構造は、アプリケーションのタスク依存関係を直接表すものとなる。

【００１７】

図１Ａを再び参照して、１４０において、アプリケーショングラフネットワークのランタイム管理が、アプリケーション開発者等のユーザに提供される。

一実施形態では、ＤＭフレームワークは、たとえば、処理グラフネットワークのグラフィカルユーザインターフェース（ＧＵＩ）ソフトウェアプログラムを介して、リアルタイムグラフィカルディスプレイを提供する。

このようなディスプレイは、処理グラフトポロジー及びタスクのリアルタイム性能統計量を動的に視覚化したものをユーザに提供する。

リアルタイム性能統計量には、アプリケーションにおけるレイテンシ並びに１つのタスク当たり及びアプリケーション全体当たりのスループット統計量が含まれる。
このグラフィカルディスプレイによって、ユーザは、アプリケーションを表す処理グラフネットワークのグラフ管理を通じたアプリケーション構築又はアプリケーション開発を管理及び操作することが可能になる。
たとえば、ユーザは、修正されたタスクモジュールの内部状態もタスクモジュール内で修正し又は内部状態は修正せずに、処理グラフネットワークの１つ又は複数のタスクモジュールへの接続（複数可）及びそれらタスクモジュールからの接続（複数可）を修正することができる。
グラフ管理について、アプリケーションランタイムに、タスクモジュールが一旦接続され、メディアタイプが各接続について求められると、アプリケーションは、ＤＭフレームワークによる実行の準備が完了する。
次に、ＤＭフレームワークのグラフィカルディスプレイプログラムは、アプリケーショングラフの開始コマンドを発行する。
この開始コマンドは、ＤＭフレームワーク内のグローバルスケジューラのオペレーションをトリガする。
これに応答して、ＤＭフレームワークの内部スレッドは、処理グラフネットワークをトラバースして、グローバルスケジューラによって設定された所定のポリシーに従い、タスク接続、すなわちタスクモジュール間の接続にわたるメディアフローを方向づけ、１つ又は複数のタスクモジュールを使用して１つ又は複数のメディアオブジェクトを処理することにより、グローバルスケジューラにリストされた利用可能なジョブ（複数可）を遂行する。

【００１８】

一実施形態では、グローバルスケジューラは、所定のポリシーを自動的に用いて、タスクモジュール間の選ばれた接続に基づいて、処理グラフネットワークのジョブを実行するためのタスクモジュールをトラバースする定義された実行スレッドを管理する。
また、グローバルスケジューラは、個々のタスク及びタスクの全体グラフの平均レイテンシ及びスループット等の計算統計量の経過も追跡して、アプリケーション性能のボトルネックを識別する。
グローバルスケジューラは、実行スレッドによる実行用のジョブのリストを含む。
したがって、実行スレッドが、ジョブの遂行後にタスクモジュールを終了させるとき、その実行スレッドは、グローバルスケジューラを再び参照して、遂行される次のジョブをジョブリストにおいて識別し、関連付けられるタスクモジュールに進んで、関連付けられるメディアオブジェクトのセットに対してこのようなジョブを遂行する。

【００１９】

図１Ｂは、本発明の一実施形態によるグローバルスケジューラのオペレーションを示している。

【００２０】

１４１において、グローバルスケジューラは、各ジョブを動的に作成して自身のジョブリストに記憶する。
たとえば、タスクモジュールによって所望されるデータオブジェクト又は必要とされるデータオブジェクトが、タスクモジュールによって利用可能になると、このようなタスクモジュールによる実行用にジョブが作成されてリストされる。
別の例では、たとえば、ＤＭフレームワークの他のタスクモジュールによる処理用にメディアオブジェクト又は他のデータオブジェクトを生成するのに、ジョブがソースタスクモジュール（入力ピンを有しない）で望まれるとき、ジョブを動的に作成してリストすることができる。

【００２１】

１４２において、グローバルスケジューラは、１つ又は複数の所定のポリシーに基づいて、自身のジョブリストの各ジョブの実行を自動的にスケジューリングする。

この自動スケジューリングは、リストされた各ジョブの特定の実行スレッドへの割り当てを含む。

【００２２】

１４３において、ジョブが異なる実行スレッドに割り当てられることを回避するために、各ジョブが実行スレッドに一旦割り当てられると、グローバルスケジューラは、ジョブリストからそのジョブを自動的に除去する。

したがって、自身の所定のポリシーに基づくグローバルスケジューラのオペレーションは、自動的であり、アプリケーション開発者にトランスペアレントである。

グローバルスケジューラの所定のポリシーは、以下でさらに説明される。

【００２３】

図３Ａ～図３Ｃは、本発明の一実施形態による、アプリケーションが方法論１００の最後の３つのフェーズ（分解、組み立て、及びグラフ管理）を通過する時のアプリケーション及びその構成要素のグラフィカル説明図を提供する。

図３Ａでは、データフロー解析後に、アプリケーションは、５つの処理タスク３１０及び４つの信号フォーマット３２０によって示すように、その構成要素のタスク及び信号に分解される。

次に、図３Ｂでは、矢印３３０によって示すようなタスク依存関係が、タスクのタスク構造への組み立て中に明示的にされる。

最後に、図３Ｃでは、タスク接続３３０にわたる信号のフローの管理を通じて、完成されたタスクグラフを管理することにより、アプリケーションが実行される。

【００２４】

図４は、ＤＭフレームワークを使用して所望のアプリケーションを構築又は開発する一例を示している。

最初に、同期されたカメラ等の信号ソース４１０が識別される。

次に、さまざまなタスクモジュール４２０～４５０が、所望のアプリケーションについてインスタンス化される。

次に、信号ソース４１０及びタスクモジュール４２０～４５０が接続されて、単一のアプリケーショングラフ、すなわち処理グラフネットワークが形成される。

次に、このグラフは、開始、停止、破棄等の簡単なグラフコマンドを通じて管理される。

【００２５】

フレームワーク

本発明の一実施形態によれば、ＤＭフレームワークは、設計によるコンピューティングサービスであり、コンピュータ等の単一のコンピューティングマシンの実行環境でモデル化される。

このようなモデルでは、ＤＭフレームワークは、マシン上のコンピューティング資源の制御を有するものと仮定される。

換言すれば、ＤＭフレームワークは、マシンのオペレーティングシステム（ＯＳ）スケジューラの予測のつかない変動により、ＣＰＵ資源を得るために競合することはない。

もちろん、通常の非リアルタイムオペレーティングシステムでは、この仮定は、通常のＯＳオペレーションによる先取りに起因して満たされない。

しかしながら、ＤＭフレームワークを使用してすべての計算集約型アプリケーションを特定のマシン上で実施することにより、このような仮定は、合理的な近似であることが分かっている。

【００２６】

外部プロセスは、フレームワーク性能に影響を与えないので、フレームワークが外部プロセスにも影響を与えないことを予想することも合理的である。

この巧みな分離は、（アプリケーション処理フェーズの）プロセスを、コンピューティング及びＩ／Ｏの２つのカテゴリーに分割することによって可能である。

コンピューティングプロセスは、かなりの時間を要し、スループットの影響を受ける。

たとえば、ビデオコーデックは、かなりのレイテンシを有する場合があるが、ビデオコーデックが３０Ｈｚのフレームレートを維持することができる場合に、性能は良好である。

他方、Ｉ／Ｏプロセスは、ハンドリングに必要な時間がより少ないが、レイテンシの影響を受ける。

たとえば、新しいロケーションにウィンドウを描くことは、比較的高速に行われるが、このタスクの遂行に遅延があった場合、ユーザは気付くことになる。

同様の議論は、出力デバイスでのオーディオのプレイ又はキーボードでのストロークの取り込みにも当てはまる。

したがって、Ｉ／Ｏオペレーション（たとえば、カメラデバイスのリスン又はウィンドウイベントのハンドリング）は、マシン上のネイティブプラットフォーム又はＯＳに注意深く委ねられ、ＤＭフレームワークのタスクモジュールは、Ｉ／Ｏ処理能力を何ら有しない計算モジュールになる。

コンピューティングタスク及びＩ／Ｏタスクへの処理のこの分離は、他の２つの仮定に変換される。

すなわち、ＤＭフレームワークは、他の計算集約型アプリケーションと競合していないこと、及び、ネイティブプラットフォームは、Ｉ／Ｏ応答性についてＤＭフレームワークと競合していないことの２つの仮定に変換される。

【００２７】

一実施形態では、ＤＭフレームワークの上述した計算モデルを実施することは、フレームワーク実行スレッドの優先順位を人工的に下げることを含む。

これによって、マシン上のネイティブＯＳがＩ／Ｏ応答性を有することが確保される。

Ｉ／Ｏは高速であるので、マシンにおいてＩ／Ｏに必要とされないＣＰＵ時間の残りは、唯一の計算集約型アプリケーションであるフレームワークに与えられる。

換言すれば、ＯＳ及び他の標準的な優先順位のスレッドが、Ｉ／Ｏプロセスをハンドリングする間（及びその後に初めて）フレームワークは、計算をハンドリングするように作動可能である。

【００２８】

マシン（たとえば、単一のコアを備えた単一のプロセッサを有するマシン）のシングルプロセッサのシナリオでは、ＣＰＵは、信号ソースからの初期データ信号に対して処理を行い、信号の波が完全に消耗されるまで、当該データ信号及びその子孫信号を、処理グラフネットワークを通じて（データ依存関係により導かれる任意の有効な順序で）伝播する。

このプロシージャは、次の初期信号及びその後の初期信号に対しても同様に繰り返される。

新しい初期データ信号の平均到達率が、各データ波の平均完了率よりも大きい場合、アプリケーションが後れを取らないために（すなわち、増加の一途をたどるレイテンシが絶えず遅れることを回避するために）、いくつかの初期データ信号を破棄することができる。

【００２９】

マルチプロセッサ又はマルチコアのシナリオ（たとえば、マシンが複数のプロセッサ又は１つ若しくは複数のマルチコアプロセッサを有する）では、タスク並列性やデータ並列性等の潜在的な並列性が、アプリケーションの動的な振る舞いを大きく変化させる。

タスク並列性の場合、各タスクモジュールは、これまでの計算履歴（history previous computations）としてその内部状態の組を有する（with its internal state set）順序計算モジュールである。

この履歴の一例は手の追跡座標であり、この場合、先行フレームにおけるロケーションによって、現フレームにおけるロケーションの検索が不要になる。

したがって、所定の履歴の状態を可能にするために、各モジュールのコードは、シーケンシャルに実行される。

これは、どの所与の時刻においても、１つの実行スレッドしか特定のモジュールに常駐することができないことを暗に意味し、これは、「ライブ」の実行スレッドの最大数がグラフのモジュールの個数であることを暗に意味する。

換言すれば、順序モジュールの処理グラフネットワークで達成可能な最良の並列性は、タスク並列性である。

すなわち、モジュールの個数に等しいプロセッサを有するマシンは、使用可能なタスク並列性の限界に達している。

したがって、各順序モジュールは、本質的には、どの所与の時刻においても、１つのジョブを遂行するために、１つのプロセッサと同等のものを消費して、そのプロセッサで１つの実行スレッドしか動作させず、動作する他の実行スレッドが存在しないので、プロセッサが追加されても、性能はもはや改善されない。

実際には、全体のアプリケーションスループットは、この場合、最も遅いモジュールのレイテンシによって制限される。

この状況においても、本発明の実施形態は、所与のタスクモジュールに関連付けられるジョブの処理を、時間と共に、異なる処理ユニットにシフトすることができることに留意すべきである。

【００３０】

他方、データ並列性は、プロセッサの個数が増加するにつれて、線形の性能改善を享受することができる。

ＤＭフレームワークでデータ並列性を用いるために、少なくとも１つのタスクモジュールは、そのアルゴリズムコードが再入力される組み合わせ計算モジュールである。

その理由は、複数のスレッドが、コードを実行して、複数の処理ユニットにより同時に複数のジョブを遂行又は実行することができるからである。

スレッドは、多くの場合、実行時間が変化し、したがって、その出力は、順序どおりでない場合がある。

下流側モジュールが組み合わせモジュールである場合、スレッドは、より多くのデータ並列性を活用して、自由に動作し続ける。

一方、下流側モジュールが順序モジュールである場合、正確なシーケンスが下流側モジュールの入力バッファで獲得されるまで、スレッドの生成を阻止しなければならない。

【００３１】

上述したように、ＤＭフレームワークのタスクモジュールは、その時間依存関係によって組み合わせモジュール又は順序モジュールとしてカテゴリー化又は指定される。

組み合わせモジュールは、もっぱら現在の入力の関数である出力を生成する。換言すれば、これらのモジュールは、前の実行の内部履歴を何ら有しない。

したがって、組み合わせモジュールは、前の計算の関数ではない内部状態を有することができる。

他方、順序モジュールは、前の実行の内部メモリを有し、したがって、出力は、現在の入力及び前の入力の双方に依存することができる。

この状況では、データは、正しい順序で入力に到達しなければならない。

順序モジュールは、現在の状態を次の実行へ転送することによって組み合わせモジュールに変換することができる。

この転送は、追加の出力を追加の入力にリンクすることによって達成される。この変換は、より多くの並列性を明らかにするのに役立つ。

したがって、可能ならば、大きな順序モジュールは、小さな順序モジュール及び大きな組み合わせモジュールの組み合わせに分解される。

しかしながら、一定の順序モジュールは、決して組み合わせモジュールとすることはできず、それらの本来的な順序的振る舞いは、並列性の量を常に制限するので、本発明の一実施形態によれば、アプリケーションをモデル化するために、組み合わせモジュール及び順序モジュールの双方が、ＤＭフレームワークで指定され用いられる。

このようなモジュールは、通常、ソース（たとえば、カメラ等の本来的にシーケンシャ

ルな入力デバイスによってトリガされるモジュール）又はシンク（たとえば、会話データを出力バッファに書き込むオーディオモジュール）である。

【００３２】

各順序タスクモジュール又は各組み合わせタスクモジュールを、マルチプロセッシング環境の専用処理ユニットが動作可能又は実行可能であることが理解されるべきである。

たとえば、処理ユニット１がタスクモジュールＡを動作させ、処理ユニット２がタスクモジュールＢを動作させ、処理ユニット３がタスクモジュールＣを動作させる等である。

代替的に、１つ又は複数の順序タスクモジュールを、マルチプロセッシング環境の１つの処理ユニットが動作可能又は実行可能である。

たとえば、処理ユニット１がタスクモジュールＡ及びＢを動作させ、処理ユニット２がタスクモジュールＣを動作させ、処理ユニット３がタスクモジュールＤ、Ｅ、及びＦを動作させる等である。

さらに、各実行スレッドは、自身に割り当てられたジョブに応じて、時間と共に、マルチプロセッシング環境の異なる処理ユニットを用いて、１つ又は複数のタスクモジュールにおいて割り当てられたジョブを実行することができる。

たとえば、単一の実行スレッドは、時間と共に、プロセッサ間をホップすることもできるし、異なるタスクを実施することもできるし、その双方を行うこともできる。

【００３３】

順序モジュールの使用によるタスク並列性の限界内であっても、次にどのタスクを実行するのかを選ぶことには多くの選択肢がある。

一実施形態では、グローバルスケジューラは、最小限のエンドツーエンドレイテンシを優先する所定のポリシーをジョブについて実施することができる。

たとえば、ＤＭフレームワークで最も古い初期信号がまだアクティブであるのか、それとも、すでに使用又は破棄されているのかに関わらず、ＤＭフレームワークでまだアクティブである、最も古い初期信号の子孫を優先するポリシーを実施することができる。

したがって、それらの子孫信号を用いるジョブには、処理グラフネットワークのタスクモジュールによって処理するための実行スレッドが優先的に与えられる。

したがって、ＤＭフレームワークが、メディアオブジェクト及びそれらの子孫を含むジョブに優先順位付けすることができるように、メディアオブジェクトには、それらメディアオブジェクトがプロダクション（又はソース）タスクモジュールによって作成された時間を示すタイムスタンプを設けることができる。

代替的に、メディアオブジェクト又はタスクオブジェクトには、ＤＭフレームワークのグローバルスケジューラが、先に述べたような所定のポリシーに基づいて、関係するジョブを優先順位付けすることを可能にする優先順位タグ又は他のインジケータを設けることもできる。

たとえば、オーディオメディアオブジェクトには、ビデオメディアオブジェクトよりも高い優先順位を与えることができ、それらの優先順位は、優先順位タグ又はインジケータで示される。

ジョブの実行優先順位が、関連付けられるメディアオブジェクトの優先順位タグ又はインジケータから一旦決定されると、このようなタグ又はインジケータは、ジョブの優先順位付けにもはや使用されない。

【００３４】

別の実施形態によれば、グローバルスケジューラは、基礎となるタスクに基づいて、一定のジョブを他のジョブよりも優先する所定のポリシーを実施することができる。

たとえば、特定のジョブには、この特定のジョブの基礎となるタスクの出力にどれだけ多くの他のタスクが依存しているのかに基づいて、実行スレッドによる遂行用のより高い（又はより低い）優先順位が与えられる。

さらに、たとえば、或るジョブの基礎となるタスクの利用可能なすべての必要な入力のいくつかが、他のタスクモジュールによる出力にとって利用可能でないことから、そのジョブが、その基礎となるタスクの利用可能なすべての必要な入力をまだ有していないとき

、そのジョブには、遂行についてより低い優先順位が与えられる。
別の例では、最も長く待機していたタスク、したがってジョブ（複数可）を優先（又は劣後）するために、一定のジョブ（複数可）には、その基礎となるタスクが実行されてから又は実行用にスケジューリングされてからどれだけの時間が経ったかに基づいて、より高い（又はより低い）優先順位が与えられる。
【００３５】
さらに別の実施形態によれば、グローバルスケジューラは、同じ処理ユニット又は一定の選択された処理ユニット（複数可）によって実行される実行スレッドに与えられるプレファレンスに基づいて、一定のジョブを他のジョブよりも優先する所定のポリシーを実施することができる。
たとえば、グローバルスケジューラは、ＤＭフレームワークのすべての実行スレッドについて１つのジョブリストを有するが、実行スレッドのそれぞれは、このようなジョブリストについてグローバルスケジューラに別個のジョブ優先順位リストを保持する。
この別個のジョブ優先順位リストは、たとえば、各実行スレッドがジョブリストのジョブに設ける優先順位の或る重み付けとすることができる。
したがって、同じジョブリストの各ジョブに関連付けられる優先順位は、実行スレッドごとに異なる。
その結果、実行スレッドが、異なるプロセッサにジャンプする頻度を削減するために、特定の実行スレッドと同じプロセッサ上での実行が予定されているジョブリストの次のジョブの実行を優先するようにその特定の実行スレッドのジョブ優先順位リストを設定することによって、グローバルスケジューラは、キャッシュ使用量を改善し、キャッシュミスの回数を減少させることができる。
したがって、上記で述べたような順序制約条件のいくつかの取り除くことによって、グローバルスケジューラは、データ並列性も活用することができる。
これは、特に、性能ボトルネックであるモジュールにとって重要である。
したがって、グローバルスケジューラは、データ並列性を活用してアプリケーション性能を高めるポリシーを実施することができる。
データ並列性の利点の１つは、「ライブ」スレッドの個数が、タスクモジュールの個数にも、プロセッサ等の処理ユニットの個数にも、マルチコアプロセッサのコアの個数にも等しくなる必要がないことである。
【００３６】
したがって、エンドツーエンドレイテンシを削減し、破棄されるメディアオブジェクトに対する処理の浪費を回避するために、グローバルスケジューラは、すべての保留中の処理要求のそのグローバルな知識を使用して、オンザフライ（すなわち、リアルタイム）のベストエフォート型タスク優先順位決定を行うことができ、たとえば、どのメディアオブジェクトを次に処理するのかについての決定を行うことができる。
たとえば、グローバルスケジューラは、エンドツーエンドレイテンシを最小にするために、最も古い祖先（すなわち、最も古い初期信号）を有するメディアオブジェクトの実行を優先するタスクスケジューリングを提供することができる。
リアルタイムモニタツールによって、開発者は、アプリケーションのレイテンシを調べて、アプリケーション性能のボトルネックを識別するために、モジュール性能の統計量を調べることが可能になる。
【００３７】
実施態様
図５は、ＤＭフレームワークの実施階層５００を示している。
この実施階層５００は、抽象レイヤ５４０及びフレームワークカーネル５３０によって形成される。
フレームワークカーネル５３０は、アプリケーション５１０とアプリケーションを動作させるマシンのネイティブＯＳ５５０との間に位置する。
抽象レイヤ５４０は、ネイティブＯＳ５５０によって表されるようなホストプラットフ

ォームからフレームワークカーネル５３０を隔離して、フレームワークカーネルをプラットフォームから独立した状態にする。
コンポーネントライブラリ５２０は、一般に再利用可能な多くのモジュールを含む。アプリケーション開発者は、すべてのレベルにアクセスすることができる。
【００３８】
最下位レベルには、ネイティブＯＳ５５０によって表されるようなホストプラットフォームが位置する。
ネイティブＯＳ５５０は、マルチスレッドサポート、タイミングメカニズム、及びプログラミングコンパイラ（たとえばＡＮＳＩ　Ｃ＋＋コンパイラ）の３つの要素を含む。
マルチスレッドサポートは、計算を制御するためのスレッド抽象体（thread abstraction）だけでなく、スレッドを制御するのに必要な同期オブジェクトも含む。
ＤＭフレームワークは、単一プロセッサマシン上で作動することができるが、一実施形態では、基礎となるハードウェアは、並列実行又は並列処理から利益を受けるために、対称型マルチプロセッサ（ＳＭＰ）マシンである。
このような場合、抽象レイヤ５４０は、ＳＭＰ抽象レイヤとなる。
タイミングメカニズムは、レイテンシ測定等の性能解析に要求される。
プログラミングコンパイラは、実行可能ファイルを生成するのに要求され、このようなコンパイラは、ＤＭフレームワーク全体を通じてコンパイラに設けられた抽象体（たとえば、文字列、ベクトル、マップ（map）、集合、デキュー）の使用を可能にする標準的なテンプレートライブラリ（たとえば、Ｃ＋＋標準テンプレートライブラリ）を有するべきである。
【００３９】
ＳＭＰ抽象レイヤ５４０は、第１のミドルウェアレベルである。
これによって、ＤＭフレームワークを他のプラットフォーム及びオペレーティングシステムに移植することが容易になる。
Ｔｈｒｅａｄ（スレッド）抽象体は、ＤＭフレームワークに、ＯＳスレッドのネーミング、スポーニング（spawn）、及びデバッグの能力を提供する。実際のＴｈｒｅａｄ作成は、ネイティブプラットフォームコールで行われ、各スレッドは、その一意の名称に関連付けられるログファイルを有することができ、各スレッドの実行トレースの作成を可能にする。
Ｍｕｔｅｘ（ミューテックス）抽象体及びＳｅｍａｐｈｏｒｅ（セマフォ）抽象体によって、Ｔｈｒｅａｄの同期が可能になる。
Ｍｕｔｅｘは、２つの以上のスレッドが同時にコードを実行することを防止するための標準的な相互排除オブジェクトであり、Ｓｅｍａｐｈｏｒｅは、Ｔｈｒｅａｄ間でシグナリングを行うための標準的で効率的なメカニズムである。
ＳｔｏｐＷａｔｃｈ（ストップウォッチ）抽象体又はＴｉｍｅｒ（タイマ）抽象体は、プラットフォームタイミング機能を使用して時間を測定する能力をカプセル化する。
時間の測定は、性能解析に必須である。
【００４０】
第２のミドルウェアレベルであるフレームワークカーネル５３０は、すべてのフレームワークアプリケーションによって使用されるコアデータフロー機能を実施する。
これは、アプリケーションを構築するための拡張可能なＴａｓｋ抽象体及びＭｅｄｉａ抽象体を供給する。
フレームワークカーネルは、内部にも、自身の複雑度を管理するためのいくつかの抽象体を有する。
第１に、ＩｎｐｕｔＰｉｎ（入力ピン）オブジェクト及びＯｕｔｐｕｔＰｉｎ（出力ピン）オブジェクトが、メディアオブジェクトを転送するためのタスクオブジェクト間の接続を表す。
第２に、Ｇｒａｐｈ（グラフ）オブジェクトが、互いに接続されたタスクオブジェクトを管理し、start()及びstop()等のグラフ全体のコマンドのインターフェースとして機能

する。
Ｇｒａｐｈコマンドは、単一のＴａｓｋ引数を有するが、接続性を使用してアプリケーショングラフ全体をトラバースし、グラフの各モジュールにコマンドを適用する。
第３に、メモリマネージャオブジェクトが、メディアオブジェクトを記憶するためのメモリバッファを提供する。
メモリマネージャは、バッファ使用量を追跡し、事前に割り当てられたバッファを再利用するためのファシリティを有し、メモリ統計量を報告することができる。
先に述べたように、グローバルスケジューラは、フレームワークカーネル５３０に常駐して、処理グラフネットワークをトラバースする実行スレッドを管理し、平均レイテンシ及びスループット等の計算統計量の経過を追跡する。
【００４１】
コンポーネントライブラリ５２０は、アプリケーションとカーネルとの間に位置する再利用可能なコンポーネントの絶えず増大する集まりである。
アプリケーション開発者は、共通の機能（たとえば、オーディオ記録又は画像色空間変換）を再実装するのではなく、事前に構築された役立つタスクオブジェクトを再利用可能コンポーネントライブラリ５２０から見つけることができる。
事前に構築されたタスクオブジェクトの例には、カメラインターフェース、グラフィックス機能、オーディオコーデック、ビデオコーデック、ネットワーキングモジュール等が含まれるが、これらに限定されるものではない。
他者の作業を利用することは、迅速な開発の重要な態様である。
【００４２】
最後の実施レイヤは、ストリーミングメディアアプリケーション等のアプリケーション５１０である。
アプリケーション５１０は、先のすべてのレイヤにアクセスすることができる。
プラットフォーム独立性をさらに推進するために、アプリケーションは、ＤＭフレームワーク用に作成されたあらゆる下位レベルライブラリのすべての内部オブジェクトにアクセスすることができ、ＤＭネットワークが、異なるプラットフォーム上で実施されている下位ライブラリレベルのクラスについて気にする必要がないようにされる。
一方、フレームワークカーネル５３０では、ＤＭフレームワークインターフェースの複雑性を最小にするために、タスク抽象体及びメディア抽象体のみがアクセス可能である。
内部オブジェクトは、タスクオブジェクト及びメディアオブジェクトを通じて又は静的なフレームワークプロシージャを通じて間接的にアクセスされる。
【００４３】
本発明の一実施形態によれば、ＤＭフレームワークは、複数の特徴的な実施機能を有する。
第１に、入力ピン又は出力ピン上のデータが常に相互に関連付けられるべきであることがアプリオリに知られている場合、入力ピン又は出力ピンをグループ化するための便利なメカニズムがある。
たとえば、図４の結合モジュール４４０は、常に一対の画像に対して処理を行う。それら入力ピンを同じ入力グループにすることによって、モジュールは、双方の画像が到着した時にのみ作動し始め、開発者が入力画像を管理し関連付ける必要性が回避される。
この関連付けは、アプリオリに知られているので、静的同期と呼ばれる。第２に、タスクモジュールの出力ピンからの「ファンアウト」が利用可能である。
このファンアウトでは、タスクモジュールの出力ピンは、複数のタスクへの入力とすることができる。
したがって、或るタスクモジュールから出力されたメディアオブジェクトを、その後、他の複数のタスクが使用することができる。
さらに、他の複数のタスクがメディアオブジェクトを修正する必要はなく、単にそのメディアオブジェクトを用いて他のジョブを遂行するだけである場合に、メディアオブジェクトの複製コピーをフレームワークメモリで作成する必要がないように、出力メディアは

読み出し専用とすることができる。
したがって、メディアオブジェクトを含む同じメモリバッファをすべての受信タスクモジュールへ送信することができる。
【００４４】
第３の重要な実施機能は、自動シリアライズである。
Ｍｅｄｉａ基底クラスは、どのＭｅｄｉａオブジェクトも、その複雑性にかかわらず、平坦にすることができる強力なシリアライズプロシージャを有する。
Ｍｅｄｉａオブジェクトは、固定長フィールド（たとえば、画像サイズ、フォーマット仕様）及び可変長フィールド（たとえば、画像バイト、オーディオデータ）の双方を有することができる。
シリアライズプロシージャは、どの深さのＭｅｄｉａ構造もトラバースすることができ、そのＭｅｄｉａ構造を、ＤＭネットワークによる出力用の単一の平坦なバッファに変換して、ファイル又はネットワークストリーム等のシリアル表現にすることができる。
同様に、自動デシリアライズプロシージャは、平坦化された表現を読み出し、その表現を、ＤＭネットワークによる処理用にメモリで深いＭｅｄｉａ構造に変換して戻すことができる。
したがって、アプリケーション開発者は、このような出力メディアオブジェクトを効率的に記憶するために、メディアオブジェクトをＤＭネットワークによる処理用の適切なフォーマットに変換することにも、ＤＭネットワークによる処理後にメディアオブジェクトを変換して戻すことにも関与する必要がない。
【００４５】
一実施形態では、コンポーネントライブラリ５２０、フレームワークカーネル５３０、及びＳＭＰ抽象レイヤ５４０は、Ｃ、Ｃ＋＋、Ｃ＃、Ｊａｖａ（登録商標）等の任意に適したコンピュータプログラミング言語からのコードを含むコンピュータ実行可能プログラムを有する１つ又は複数のソフトウェアプログラム、アプリケーション、又はモジュールによって実施することができる。
これらコンピュータ実行可能プログラムは、コンピュータ化システムによって実行可能であり、コンピュータ化システムには、コンピュータ又はコンピュータのネットワークが含まれる。
コンピュータ化システムの例には、１つ若しくは複数のデスクトップコンピュータ、１つ若しくは複数のラップトップコンピュータ、１つ若しくは複数のメインフレームコンピュータ、１つ若しくは複数のネットワーク化コンピュータ、１つ若しくは複数のプロセッサベースのデバイス、又は同様の任意のタイプのシステム及びデバイスが含まれるが、これらに限定されるものではない。
図６は、図５の階層５００を実施するためのプラットフォームとして使用されるように作動可能なコンピュータ化システム６００のブロック図を示している。
より高度なコンピュータ化システムが使用されるように作動可能であることが理解されるべきである。
さらに、コンピュータ化システム６００にコンポーネントを追加し又はコンピュータ化システム６００からコンポーネントを除去して、所望の機能を提供することもできる。
【００４６】
コンピュータシステム６００は、プロセッサ６０２等の１つ又は複数のプロセッサを含み、ソフトウェアを実行するための実行プラットフォームを提供する。
したがって、コンピュータ化システム６００は、Intel、Motorola、AMD、及びCyrixからのプロセッサ等、複数のコンピュータプロセッサの任意のものの１つ又は複数のシングルコアプロセッサ又はマルチコアプロセッサを含む。
本明細書で参照されるように、コンピュータプロセッサは、中央処理装置（ＣＰＵ）又は他の任意の多目的プロセッサ若しくは多目的マイクロプロセッサ等の汎用プロセッサとすることができる。
また、コンピュータプロセッサは、グラフィックス処理装置（ＧＰＵ）、オーディオプ

ロセッサ、デジタル信号プロセッサ、１つ又は複数の処理目的に専用化された別のプロセッサ等の専用プロセッサとすることもできる。

プロセッサ６０２からのコマンド及びデータは、通信バス６０４により通信される。コンピュータシステム６００は、ソフトウェアがランタイム中に常駐するメインメモリ６０６、及び２次メモリ６０８も含む。

２次メモリ６０８は、階層５００の１つ又は複数のコンポーネントを実施するソフトウェアプログラム、アプリケーション、又はモジュールを記憶するのに使用することができるＣＲＭとすることもできる。

メインメモリ６０６及び２次メモリ６０８には、それぞれ、たとえば、ハードディスクドライブ、及び／又は、フロッピー（登録商標）ディスケットドライブ、磁気テープドライブ、コンパクトディスクドライブ等を表す着脱可能ストレージドライブ、若しくはソフトウェアのコピーが記憶される不揮発性メモリが含まれる。

一例では、２次メモリ６０８には、ＲＯＭ（読み出し専用メモリ）、ＥＰＲＯＭ（消去可能プログラマブルＲＯＭ）、ＥＥＰＲＯＭ（電気的消去可能プログラマブルＲＯＭ）、又はプロセッサ若しくは処理ユニットにコンピュータ可読命令を提供することができる他の任意の電子的な、光学的な、磁気的な、若しくは他のストレージデバイス若しくは伝送デバイスも含まれる。

コンピュータシステム６００は、ディスプレイ６１４及びユーザインターフェースを含む。ユーザインターフェースは、キーボード、マウス、スタイラス等の１つ又は複数の入力デバイス６１２を備える。

しかしながら、入力デバイス６１２及びディスプレイ６１４はオプションである。ネットワークインタフェース６１０は、他のコンピュータシステムとの通信用に設けられる。

【００４７】

コンポーネント５２０、５３０、及び５４０のそれぞれを別々のコンピュータ化システムで実施することができる代替的な実施形態、又は、このようなコンポーネントのいつくかは或るコンピュータ化システムによって実行され、それ以外のものは別のコンピュータ化システムによって実行されるか、若しくは、フレームワークカーネル５３０のタスクモジュールの少なくともいくつかは、異なるコンピュータ化システムによって実行される代替的な実施形態が考えられる。

したがって、ミドルウェアフレームワークは、マルチプロセッシング環境で作動することができる。

【００４８】

要約すれば、アプリケーション開発者は、最大限の並列性又は最も望ましい並列性を得るためにＤＭフレームワークの実行スレッドの個数を指定する能力を有する。

一実施形態では、スレッドの個数は、プロセッサの個数に等しくなるように設定される。

別の実施形態では、特に、スレッドを計算モジュールの内部で阻止することができるときは、スレッドの個数は、プロセッサの個数よりも多い。

一方、アプリケーションのデバッグ中は、実行スレッドを容易に追跡することができるように、単一の実行スレッドを使用することが極めて有益である。

【００４９】

本明細書で説明及び図示されたものは、その変形のいくつかと共に実施形態である。

本明細書で使用された説明及び図という用語は、例示としてのみ述べられ、限定を意図するものではない。

当業者は、主題の趣旨及び範囲内において多くの変形が可能であることを認識するであろう。

主題は、以下の特許請求の範囲及びその均等物によって確定されるように意図されている。

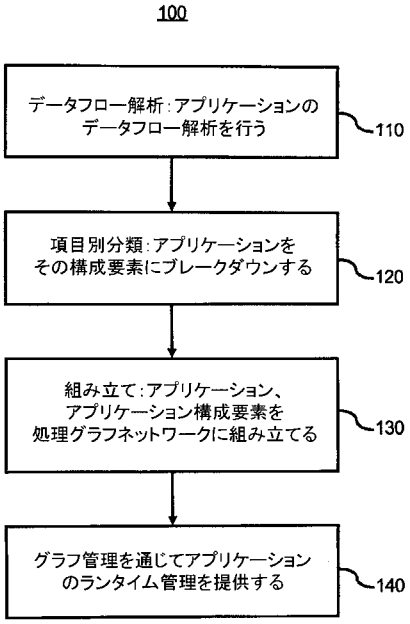
特許請求の範囲において、すべての用語は、別段の指定がない限り、その最も広い合理的な意味に意図されている。

【符号の説明】

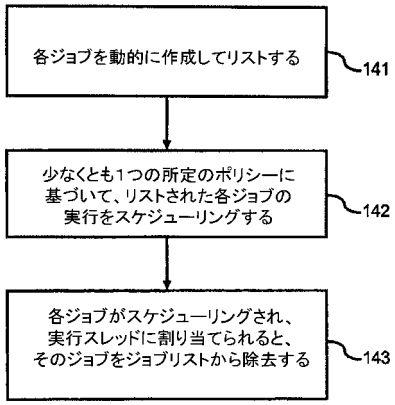
【００５０】

３１０・・・処理タスク，
３２０・・・信号フォーマット，
５１０・・・アプリケーション，
５２０・・・コンポーネントライブラリ，
５３０・・・フレームワークカーネル，
５４０・・・抽象レイヤ，
５５０・・・ネイティブオペレーティングシステム，
６０２・・・プロセッサ，
６０４・・・通信バス，
６０６・・・メインメモリ，
６０８・・・２次メモリ，
６１０・・・ネットワークインタフェース，
６１２・・・入力デバイス，
６１４・・・ディスプレイ

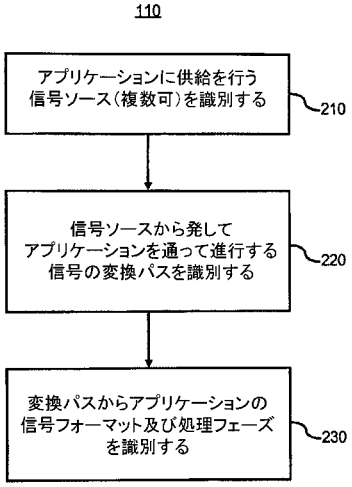
【図１Ａ】



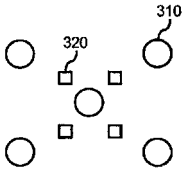
【図１Ｂ】



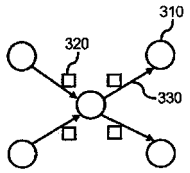
【図２】



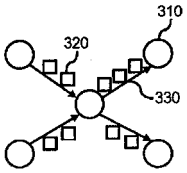
【図３Ａ】



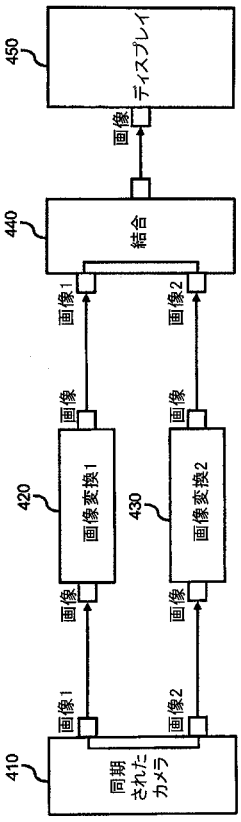
【図３Ｂ】



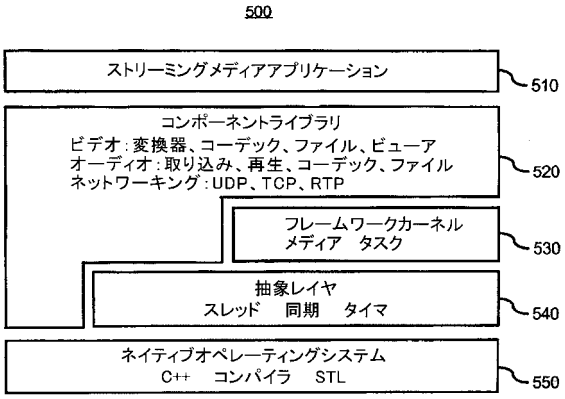
【図３Ｃ】



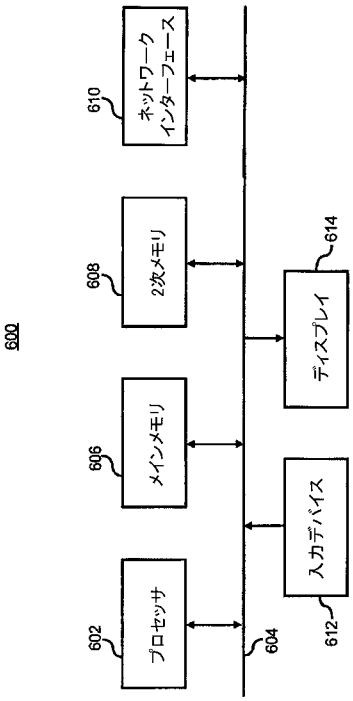
【図４】



【図５】



【図６】


600
602
プロセッサ
604
606
メインメモリ
608
2次メモリ
610
ネットワークインターフェース
612
入力デバイス
614
ディスプレイ

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2007/022908

A. CLASSIFICATION OF SUBJECT MATTER

G06F 9/44(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 8 : G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models since 1975
Japanese utility models and applications for utility models since 1975

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKIPASS(Kipo Internal), Google, YesKisti
keywords: schedul*, thead, dataflow, application, middleware, DM, dataflow middleware, components

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	TANGUAY, D. and et al. Nizza: A Framework for Developing Real-time Streaming Multimedia Applications. HP Technical Reports, HPL-2004-132, 02 August 2004. online! http://www.hpl.hp.com/techreports/2004/HPL-2004-132.pdf (acquired at 27 March 2008) See Section 3.	1-10
A	SCHRODER_PREIKSCHAT, W. PEACE - A Software Backplane for Parallel Computing. Parallel Computing. Vol. 20, no. 10-11, pp. 1471-1485. 1994. See sections 3–5.	1-10
A	HA, S. et al. 'Hardware-Software Codesign of Multimedia Embedded Systems: the PeaCE' In: Embedded and Real-Time Computing Systems and Applications, pp. 207-214, September 2006. See section 4.	1-10
A	THONG, J.V. et al. 'Multimedia Content Analysis and Indexing: Evaluation of a Distributed and Scalable Architecture' In: Proceedings of SPIE .Volume 5242, Internet Multimedia Management Systems IV 2003, pp. 137-145. See section 3.3	1-10

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
07 APRIL 2008 (07.04.2008)	**07 APRIL 2008 (07.04.2008)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer YOON, Hye Sook Telephone No. 82-42-481-8370

Form PCT/ISA/210 (second sheet) (April 2007)

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IS,IT,LT,LU,LV,MC,MT,NL,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BH,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IS,JP,KE,KG,KM,KN,KP,KR,KZ,LA,LC,LK,LR,LS,LT,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,MZ,NA,NG,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RS,RU,SC,SD,SE,SG,SK,SL,SM,SV,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,ZA,ZM,ZW

(72)発明者 ハービル・マイケル
アメリカ合衆国カリフォルニア州 パロアルト ピーオーボックス６０１８１

Ｆターム(参考) 5B376 BC07 BC16 BC45