(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2009-80583
(P2009-80583A)
(43)公開日　平成21年4月16日(2009.4.16)

| (51)Int.Cl. | F I | テーマコード（参考） |
|---|---|---|
| G06F 9/50 (2006.01) | G06F 9/46 465E | |

審査請求 未請求　請求項の数 9　ＯＬ　（全 11 頁）

(21)出願番号　特願2007-248025 (P2007-248025)
(22)出願日　平成19年9月25日 (2007.9.25)

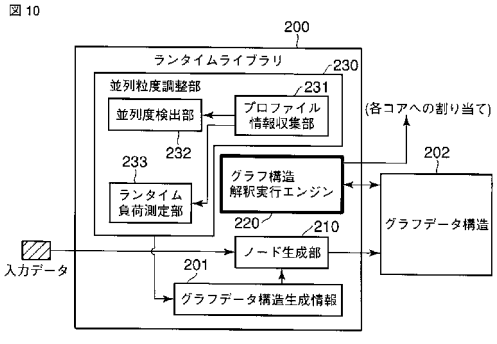
(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(74)代理人　100058479
弁理士　鈴江　武彦
(74)代理人　100091351
弁理士　河野　哲
(74)代理人　100088683
弁理士　中村　誠
(74)代理人　100108855
弁理士　蔵田　昌俊
(74)代理人　100075672
弁理士　峰　隆司
(74)代理人　100109830
弁理士　福原　淑弘

最終頁に続く

(54)【発明の名称】情報処理装置、並列処理最適化方法およびプログラム

(57)【要約】
【課題】プログラムの実行時に並列処理の粒度を適応的に最適化することを可能とする情報処理装置を提供する。
【解決手段】ランタイムライブラリ２００の並列粒度調整部２３０は、ランタイムライブラリの稼働時に、プロファイル情報収集部２３１、並列度検出部２３２およびランタイム負荷測定部２３３によるデータ収集を行い、この収集したデータから、実行ユニットの稼働率と並列制御のためのオーバーヘッドとを算出する。並列粒度調整部２３０は、実行ユニットの稼働率が高く、かつ、並列制御のためのオーバーヘッドが大きい場合、複数の処理単位をマージして、その負荷を削減する。
【選択図】　図１０

図10



【特許請求の範囲】
【請求項１】
複数の実行ユニットと、
他のモジュールと非同期に実行可能な複数の基本モジュールに分割され、当該複数の基本モジュールの時系列的な実行規則が定義されるプログラムを、前記複数の実行ユニットによって並列実行するために、前記実行規則に基づき、前記複数の実行ユニットに対する前記複数の基本モジュールの割り当てを制御するスケジューラと、
を具備し、
前記スケジューラは、
前記複数の実行ユニットによる前記プログラムの並列処理における並列化率を示す並列度を検出する並列度検出手段と、
前記複数のＣＰＵによる前記プログラムの並列処理における前記複数の基本モジュールの割り当て制御に関わる負荷を検出する負荷検出手段と、
前記並列度検出手段が検出した並列度の値が予め定められた値を越え、かつ、前記負荷検出手段が検出した負荷の値が予め与えられた値を越えていた場合、前記実行規則によって前後して実行される２以上の基本モジュールを、同一の実行ユニットに対して一組みとして割り当てられるように結合するモジュール結合手段と、
を有することを特徴とする情報処理装置。
【請求項２】
前記スケジューラは、前記モジュール結合手段が結合した２以上の基本モジュールを再分割するモジュール分割手段をさらに有することを特徴とする請求項２記載の情報処理装置。
【請求項３】
前記スケジューラの前記モジュール結合手段は、結合する２以上の基本モジュールの範囲において、当該２以上の基本モジュールに含まれる命令の再スケジューリングを行うことを特徴とする請求項１記載の情報処理装置。
【請求項４】
前記スケジューラの前記モジュール結合手段は、結合する２以上の基本モジュールの範囲において、当該２以上の基本モジュールに含まれる変数のレジスタへの再割り当てを行うことを特徴とする請求項１記載の情報処理装置。
【請求項５】
前記スケジューラの前記並列度検出手段は、前記複数の実行ユニットでの前記複数の基本モジュールの実行時間の総和を、実際の経過時間で除した値を並列度として算出することを特徴とする請求項１記載の情報処理装置。
【請求項６】
前記複数の実行ユニットは、１つのＣＰＵに内蔵されるＣＰＵコアであることを特徴とする請求項１記載の情報処理装置。
【請求項７】
前記複数の実行ユニットは、それぞれが個別に構成された複数のＣＰＵであることを特徴とする請求項１記載の情報処理装置。
【請求項８】
他のモジュールと非同期に実行可能な複数の基本モジュールに分割され、当該複数の基本モジュールの時系列的な実行規則が定義されるプログラムを、複数の実行ユニットによって並列実行する情報処理装置における並列処理最適化方法であって、
前記複数の実行ユニットによる前記プログラムの並列処理における並列化率を示す並列度を検出し、
前記複数のＣＰＵによる前記プログラムの並列処理における前記複数の基本モジュールの割り当て制御に関わる負荷を検出し、
前記並列度の値が予め定められた値を越え、かつ、前記負荷の値が予め与えられた値を越えていた場合、前記実行規則によって前後して実行される２以上の基本モジュールを、

同一の実行ユニットに対して一組みとして割り当てられるように結合する、
ことを特徴とする並列処理最適化方法。
【請求項９】
他のモジュールと非同期に実行可能な複数の基本モジュールに分割され、当該複数の基本モジュールの時系列的な実行規則が定義されるプログラムを、複数の実行ユニットによって並列実行する情報処理装置を、
前記複数の実行ユニットによる前記プログラムの並列処理における並列化率を示す並列度を検出する並列度検出手段、
前記複数のＣＰＵによる前記プログラムの並列処理における前記複数の基本モジュールの割り当て制御に関わる負荷を検出する負荷検出手段、
前記並列度検出手段が検出した並列度の値が予め定められた値を越え、かつ、前記負荷検出手段が検出した負荷の値が予め与えられた値を越えていた場合、前記実行規則によって前後して実行される２以上の基本モジュールを、同一の実行ユニットに対して一組みとして割り当てられるように結合するモジュール結合手段、
として機能させるプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
この発明は、例えばＣＰＵコアを複数内蔵するＣＰＵを搭載するコンピュータや、複数のＣＰＵを搭載するコンピュータ等に適用して好適なプログラムの並列処理制御技術に関する。
【背景技術】
【０００２】
近年、ノートブックタイプやデスクトップタイプ等、様々な種類の個人向けコンピュータ（パーソナルコンピュータ）が広く普及している。この種のコンピュータでは、例えば高精細動画像データをソフトウェアによって再生する等、その情報処理能力に対する要求はＣＰＵの性能向上の限界に迫るほどに高まる一方である。
【０００３】
このような事から、例えば複数のＣＰＵを搭載したり、また、最近では、ＣＰＵコアを複数内蔵するＣＰＵを搭載するコンピュータが登場してきている。即ち、プログラムを並列処理することで、所要時間の短縮化を図り、以て、コンピュータの性能を向上させるわけである。プログラムの並列処理を効率的に行うための仕組みについては、これまでも種々提案されている（例えば特許文献１等参照）。
【特許文献１】特開２００５－２５８９２０公報
【発明の開示】
【発明が解決しようとする課題】
【０００４】
プログラムの並列処理の１つの形態は、プログラム中の各処理単位を実行ユニットに割り当てる（複数のＣＰＵを搭載するコンピュータにおいては、各ＣＰＵへの割り当てを行い、ＣＰＵコアを複数内蔵するＣＰＵを搭載するコンピュータにおいては、各ＣＰＵコアへの割り当てを行う）スケジューラを含むランタイム処理と、各実行ユニット上で動作する処理単位との２つの構成要素から構成される。この時、処理単位の大きさを並列処理の粒度といい、粒度を細かくする方が、並列化の機会を増やすことが可能となるので、並列性能を向上できる。
【０００５】
一方で、この並列処理の粒度が細かすぎると、スケジューラが動作する回数が多くなるため、このオーバーヘッドによって、十分な性能を得られないという問題があった。
【０００６】
この発明は、このような事情を考慮してなされたものであり、プログラムの実行時に並列処理の粒度を適応的に最適化することを可能とする情報処理装置、並列処理最適化方法

およびプログラムを提供することを目的とする。

【課題を解決するための手段】

【０００７】

前述した目的を達成するために、この発明の情報処理装置は、複数の実行ユニットと、他のモジュールと非同期に実行可能な複数の基本モジュールに分割され、当該複数の基本モジュールの時系列的な実行規則が定義されるプログラムを、前記複数の実行ユニットによって並列実行するために、前記実行規則に基づき、前記複数の実行ユニットに対する前記複数の基本モジュールの割り当てを制御するスケジューラと、を具備し、前記スケジューラは、前記複数の実行ユニットによる前記プログラムの並列処理における並列化率を示す並列度を検出する並列度検出手段と、前記複数のＣＰＵによる前記プログラムの並列処理における前記複数の基本モジュールの割り当て制御に関わる負荷を検出する負荷検出手段と、前記並列度検出手段が検出した並列度の値が予め定められた値を越え、かつ、前記負荷検出手段が検出した負荷の値が予め与えられた値を越えていた場合、前記実行規則によって前後して実行される２以上の基本モジュールを、同一の実行ユニットに対して一組みとして割り当てられるように結合するモジュール結合手段と、を有することを特徴とする。

【発明の効果】

【０００８】

この発明によれば、プログラムの実行時に並列処理の粒度を適応的に最適化することが可能となる。

【発明を実施するための最良の形態】

【０００９】

以下、図面を参照して、この発明の一実施形態を説明する。

【００１０】

図１は、本実施形態に係る情報処理装置のシステム構成の一例を示す図である。この情報処理装置は、ノートブックタイプやデスクトップタイプ等のいわゆるパーソナルコンピュータとして実現されている。そして、図１に示すように、本コンピュータは、プロセッサ１、主メモリ２およびハードディスク駆動装置（ＨＤＤ）３を有しており、これらは内部バスを介して相互に接続されている。

【００１１】

プロセッサ１は、ＨＤＤ３から主メモリにロードされたプログラムを実行制御する中央演算処理装置（ＣＰＵ）であり、主要部の演算回路（ＣＰＵコア）であるコア１１を複数内蔵している。

【００１２】

主メモリ２は、プロセッサ１がアクセス可能な、例えば半導体で構成される記憶装置である。一方、ＨＤＤ３は、本コンピュータにおける補助記憶としての役割を担う、（主メモリ２と比較して）低速大容量の記憶媒体である。

【００１３】

また、図示していないが、プロセッサ１によるプログラムの処理結果等を表示するためのディスプレイや処理データ等を入力するためのキーボードなどの入出力装置が、例えばノートブックタイプの場合はさらに備えられ、また、例えばデスクトップタイプの場合はケーブル等により外部接続される。

【００１４】

コア１１を複数内蔵するプロセッサ１を搭載する本コンピュータは、複数のプログラムを並列実行することが可能であり、また、１つのプログラム中の複数の処理を並列実行することも可能である。ここで、図２を参照して、本コンピュータによって実行される並行処理仕様のプログラムの概略構成について説明する。

【００１５】

図２に示すように、本コンピュータによって実行される並行処理仕様の実行プログラム１００は、複数の直列基本モジュール１０１と、この複数の直列基本モジュール１０１を

どのような順序で実行すべきかを定義する並列実行制御記述１０２とから構成される。
【００１６】
いわゆるマルチスレッド処理では、一般的に、図３に示すように、他のスレッドとの間で（通信を含む）同期を取りながら、即ち、プログラム全体の整合性を保ちながら、各スレッドが処理を進行させていく。よって、同期の待ち合わせが多発すると、期待した並列性能が得られないことも考えられる。
【００１７】
そこで、本実施形態では、図４に示すように、他のモジュールとの同期を取る必要がない、非同期に実行可能な処理単位にプログラムを分割することで、複数の直列基本モジュール１０１を作成すると共に、この複数の直列基本モジュール１０１の時系列的な実行規則を定義する並列実行制御記述１０２を作成する。並列実行制御上、各直列基本モジュール１０１は、ノードとして表現される。このように、直列基本モジュールとは、他のモジュールと非同期に実行可能な処理単位のモジュールをいう。次に、図５を参照して、並列実行制御記述１０２について説明する。
【００１８】
図５（Ａ）は、ある直列基本モジュール１０１を表現したノードの概念図である。図示のように、各直列基本モジュール１０１は、先行ノードへのリンクと、後続ノードへの結合子とを有するノードとして捉えることができる。並列実行制御記述１０２は、各直列基本モジュール１０１それぞれについて、先行ノードへのリンク情報を記すことにより、複数の直列基本モジュール１０１の実行順序を定義する。図５（Ｂ）は、ある直列基本モジュール１０１に関する並列実行制御記述を例示する図であり、図示のように、それぞれの識別子である直列基本モジュールＩＤと、その直列基本モジュール１０１の先行ノードへのリンク情報とが記される。また、その他に、出力バッファタイプやコスト等の情報が併せて記される。
【００１９】
続いて、この複数の直列基本モジュール１０１と並列実行制御記述１０２とから構成されるという独自の構成をもつ実行プログラム１００を本コンピュータがどのように実行するのかについて説明する。
【００２０】
このような独自の構成をもつ実行プログラム１００を並列処理するために、本コンピュータでは、図６に示すランタイムライブラリ２００が用意される。このランタイムライブラリ２００は、スケジューラとしての機能を備えており、並列実行制御記述１０２がグラフデータ構造生成情報２０１として与えられる。並列実行制御記述１０２は、例えば関数型言語を用いて作成され、トランスレータによってグラフデータ構造生成情報２０１に変換される。
【００２１】
何らかのデータ入力が行われると、このデータを処理するための直列基本モジュール１０１をいくつか実行する必要が生じるが、その都度、ランタイムライブラリ２００は、グラフデータ構造生成情報２０１に基づき、グラフデータ構造２０２を動的に生成・更新していく。グラフデータ構造２０２は、その時々で適宜に実行されていくノード群の前後関係を示すグラフデータであり、ランタイムライブラリ２００は、追加対象のノード間での前後関係は勿論、実行待ちの状態にあるノードとの間の前後関係も考慮して、それらノード群のグラフデータ構造２０２への追加を行っていく。
【００２２】
また、ランタイムライブラリ２００は、あるノードの実行が完了すると、このノードをグラフデータ構造２０２から削除すると共に、このノードを先行ノードとし、かつ、その他に先行ノードがないか、または、その他の先行ノードがすべて完了している後続ノードの有無を調べて、この条件を満たす後続ノードが存在したら、そのノードをいずれかのコア１１に割り当てる。
【００２３】

このランタイムライブラリ２００の働きにより、並列実行制御記述１０２に基づく複数の直列基本モジュール１０１の並列実行が矛盾無く進められていく。また、このランタイムライブラリ２００は、プロセッサ１が内蔵するコア１１の数よりも多くの数のスレッドによって実行する（マルチスレッド）。その結果、図７に示すように、各コア１１（各コア１１のＯＳ３００配下の１スレッドであるランタイムライブラリ２００）があたかも自律的に次に実行すべき直列基本モジュール１０１を見つけ出してくるかのごとく本コンピュータを動作させることができる。スレッド間の排他制御は、ランタイムライブラリ２００による、グラフデータ構造２０２からのノードの選択と、当該グラフデータ構造の更新とのみに止まるので、図３に示した一般的なマルチスレッド処理と比較して、高い並列性能を得ることを実現する。

【００２４】

ところで、本コンピュータのプロセッサ１が内蔵するコア１１の数に対して、直列基本モジュール１０１の処理単位、つまり並列処理の粒度が細かすぎると、ランタイムライブラリ２００の稼働機会、いわゆるオーバヘッドが増えてしまい、実行効率を低下させることになりかねない。この点を考慮して、本実施形態のランタイムライブラリ２００は、この並列処理の粒度を適応的に最適化する機能をさらに備える。

【００２５】

より具体的に説明すると、いま、図８（Ａ）に示すような関係にあるノードＡ～ノードＥの５つのノードが存在するものと想定すると、ランタイムライブラリ２００は、例えば図８（Ｂ）に示すように、ノードＡとノードＢとを連結し、ノードＣ～ノードＥの先行ノードをノードＡからノードＢに変更する機能を備える。この連結を動的かつ効率的に行うことを可能とするために、ランタイムライブラリ２００は、各ノードについて、先行ノードへのリンク情報毎に図９に示す情報を保持する。

【００２６】

「リンク開放順位」は、グラフデータ構造生成情報２０１に基づき生成したノードに対して、当該ノードのリンク情報によってリンクされたノードが実行完了した順序を記録するフィールドである。「現ノード実行時の緊急度」は、あるノードに対応する直列基本モジュール１０１の実行時において、そのノードでのリンク開放順位が最後尾のノードがリンクされる（参照する）他のノードのうち、実行開始よりも進んだ状態にあるノードの数を記録するフィールドである。緊急度は、値が小さいほど高いものとする。即ち、実行開始よりも進んだ状態にあるノード数が０の場合が、最も高い緊急度ということになる。そして、「現ノード実行時の並列実行余裕度」は、現ノード実行時に実行可能状態にあるノードの数を記録するフィールドである。いずれの情報も、実行する度に変化する可能性があるため、最初の２つについては、とり得る値の配列を用意してカウントアップする。また、並列実行余裕度については、並列実行余裕度の最近Ｎ回分の平均か、期待度（たとえば、前の値×（１－α）＋今回の値×α）を使う等、データを圧縮して記録する。

【００２７】

図１０は、ランタイムライブラリ２００の機能ブロック図である。

【００２８】

ランタイムライブラリ２００は、図１０に示すように、ノード生成部２１０、グラフ構造解釈実行エンジン２２０および並列粒度調整部２３０を有している。

【００２９】

前述した、本ランタイムライブラリ２００による、グラフデータ構造生成情報２０１に基づくグラフデータ構造２０２の動的な生成・更新と、このグラフデータ構造２０２を用いたノードのコア１１への割り当て制御とは、ノード生成部２１０およびグラフ構造解釈実行エンジン２２０とによって実現されている。そして、以下にその詳細に説明する並列処理の粒度の適応的な最適化は、並列粒度調整部２３０によって実現され、並列粒度調整部２３０は、プロファイル情報収集部２３１、並列度検出部２３２およびランタイム負荷測定部２３３を有している。

【００３０】

プロファイル情報収集部２３１は、各直列基本モジュール１０１の実行完了による本ランタイムライブラリ２００の稼働時に、当該直列基本モジュール１０１に対応するノードのリンク情報における「リンク開放順位」、「緊急度」および「並列実行余裕度」の各情報を収集する。

【００３１】

並列度検出部２３２は、プロセッサ１による実行プログラム１００の並列処理における並列化率を測定する。この並列化率を示す値として、並列度検出部２３２は、すべてのコア１１での直列基本モジュール１０１の実行時間の総和を、実際の経過時間で割った値を算出する。

【００３２】

また、ランタイム負荷測定部２３３は、プロセッサ１による実行プログラム１００の並列処理に関する（オーバーヘッドである）当該ランタイムライブラリ２００の処理負荷を測定する。この負荷は、例えば各コア１１の実行クロックサイクルを記録するレジスタの値をランタイムライブラリ稼働前後で測定することで測定できる。ここで、この処理負荷時間には、実行可能なノード数が０、即ち、次に実行すべき直列基本モジュール１０１が無くなったことにより、待ち合わせている時間を含まないものとする。

【００３３】

このプロファイル情報収集部２３１、並列度検出部２３２およびランタイム負荷測定部２３３により取得される各種情報を用いて、並列粒度調整部２３０は、並列処理の粒度の適応的な最適化を以下のように実行する。

【００３４】

即ち、並列粒度調整部２３０は、単位時間当たりの本ランタイムライブラリ２００の処理負荷が、単位実行時間×（コア数－並列度）×αより大きいとき、性能ネックになっている可能性があると判断する。αは、予め定められる閾値である。

【００３５】

本ランタイムライブラリ２００の処理負荷が性能ネックになっている可能性があると判断した場合、並列粒度調整部２３０は、その処理負荷を軽減するために、ノードの結合を実施する。より具体的には、まず、並列粒度調整部２３０は、グラフデータ構造生成情報２０１のリンク情報のうち、最も緊急度が高く、かつ、並列実行余裕度の大きいものを選択する。そして、並列粒度調整部２３０は、このリンク情報を含んでいるノードと、リンク開放順位が最後尾で繋がる参照ノードとを結合する。

【００３６】

並列粒度調整部２３０は、この結合に伴い、結合するノードを属性情報として持つような新たなグラフデータ構造生成情報２０１を生成する。新たに生成する情報は、２つの直列基本モジュール１０１を同一のコア１１上で連続して実行させるためのノードに関する情報であり、そのリンク情報は、２つのノードのリンク情報をマージしたものとする。即ち、ここで新たに作成されるノードは、２つの直列基本モジュール１０１を１つのモジュールと見なす論理的な直列基本モジュール１０１に対応するものである。

【００３７】

この時、並列粒度調整部２３０は、これら２つのノードを参照していた別のノードのリンク情報を、新たに生成したノードに書き換える。これにより、次回以降、ノード生成部２１０は、この新たに生成されたグラフデータ構造生成情報２０１を使って、グラフデータ構造２０２へのノードの追加を行うことになる。

【００３８】

また、連結したノードに対応する直列基本モジュール１０１が同一のコア１１上で連続実行されることになることから、並列粒度調整部２３０は、この範囲で、命令のスケジューリングや変数のレジスタへの再割り当て、その他のコンパイラの最適化を行い、これらが効率よく処理されるための変換を併せて実行する。

【００３９】

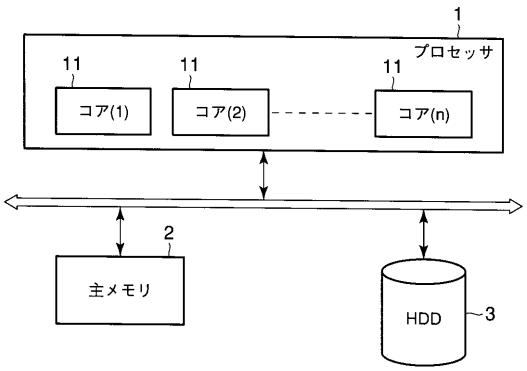
この並列粒度調整部２３０の働きにより、ランタイムライブラリ２００は、実行プログ

ラム１００の実行時に、適応的に、並列処理の粒度を自己調整することを実現する。
【００４０】
図１１は、本コンピュータが実行する並列処理最適化の動作の流れを示すフローチャートである。
【００４１】
ランタイムライブラリ２００の並列粒度調整部２３０は、ランタイムライブラリの稼働時に、プロファイル情報収集部２３１、並列度検出部２３２およびランタイム負荷測定部２３３によるデータ収集を行う（ステップＡ１）。
【００４２】
並列粒度調整部２３０は、この収集されたデータを使って、ノードを結合する必要があるか否かを判断し（ステップＡ２）、ノードを結合する必要があると判断したら（ステップＡ２のＹＥＳ）、並列粒度調整部２３０は、ノードの結合を行い（ステップＡ４）、これにより同一のコア１１で連続して実行されることになる２つの直列基本モジュール１０１の最適化を実行する（ステップＡ５）。
【００４３】
（並列処理の粒度が実行時に自己調整される）本手法によれば、実行プログラム１００の作成時には、粒度を意識することなく、各直列基本モジュール１０１を、ただ十分小さな処理単位に分割すれば良い。よって、コア１１の数毎に実行プログラム１００を作り直したり、チューニングする手間を一切無くすことができる。
【００４４】
なお、ここでは、十分細かく分割された直列基本モジュール１０１を並列実行する際のオーバーヘッド（ランタイムライブラリ２００の処理）の削減について説明したが、連結によって並列性が落ちてしまうこともありうる。例えば、処理するべき入力データの性質が変化して、モジュール毎の負荷バランスが変化する場合である。
【００４５】
これを考慮して、並列粒度調整部２３０は、連結したノードの属性情報として、元の２つのノードの情報を退避しておき、連結したノードを元の２つのノードに戻す機能を更に備えておくことが好ましい。この場合、初期状態を、いくつかのノードを連結したものにしておいて、並列度が十分でなく、オーバヘッドも小さい場合に、連結してあるノードを分割することで並列度を向上させる、という最適化も行えることとなる。
【００４６】
この粒度を大小いずれの方向にも自己調整する機能を備えれば、コア１１の使用可能状況が大きく変化する場合にも動的に対応することが可能となる。
【００４７】
また、ここでは、本コンピュータが、コア１１を複数内蔵するプロセッサ１を搭載する場合を例に本手法を説明したが、本手法は、複数のプロセッサ１を搭載するいわゆるマルチプロセッサコンピュータにおいても当然に適用できる。
【００４８】
つまり、本発明は上記実施形態そのままに限定されるものではなく、実施段階ではその要旨を逸脱しない範囲で構成要素を変形して具体化できる。また、上記実施形態に開示されている複数の構成要素の適宜な組み合わせにより、種々の発明を形成できる。例えば、実施形態に示される全構成要素から幾つかの構成要素を削除してもよい。さらに、異なる実施形態にわたる構成要素を適宜組み合わせてもよい。
【図面の簡単な説明】
【００４９】
【図１】この発明の実施形態に係る情報処理装置のシステム構成の一例を示す図
【図２】本実施形態の情報処理装置によって実行される並行処理仕様のプログラムの概略構成を説明するための図
【図３】一般的なマルチスレッド処理を示す図
【図４】本実施形態の情報処理装置によって実行されるプログラムを構成する直列基本モ

ジュールと並列実行制御記述との関係を示す図
【図５】本実施形態の情報処理装置によって実行されるプログラムの並列実行制御記述を説明するための図
【図６】本実施形態の情報処理装置上で動作するランタイムライブラリが行うプログラムの並列処理制御を説明するための図
【図７】本実施形態の情報処理装置上におけるランタイムライブラリの動作状態を示す図
【図８】本実施形態の情報処理装置上で動作するランタイムライブラリが実施するノードの結合を説明するための図
【図９】本実施形態の情報処理装置上で動作するランタイムライブラリがノードの結合を実施するために保持する情報を示す図
【図１０】本実施形態の情報処理装置上で動作するランタイムライブラリの機能ブロックを示す図
【図１１】本実施形態の情報処理装置上で動作するランタイムライブラリが実行する並列処理最適化の動作の流れを示すフローチャート
【符号の説明】
【００５０】
１…プロセッサ、２…主メモリ、３…ハードディスク駆動装置（ＨＤＤ）、１１…コア、１００…実行プログラム、１０１…直列基本モジュール、１０２…並列実行制御記述、２００…ランタイムライブラリ、２０１…グラフデータ構造生成情報、２０２…グラフデータ構造、２１０…ノード生成部、２２０…グラフ構造解釈実行エンジン、２３０…並列粒度調整部、２３１…プロファイル情報収集部、２３２…並列度検出部、２３３…ランタイム負荷測定部。

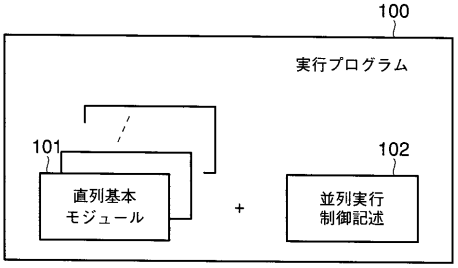
【図１】

図1



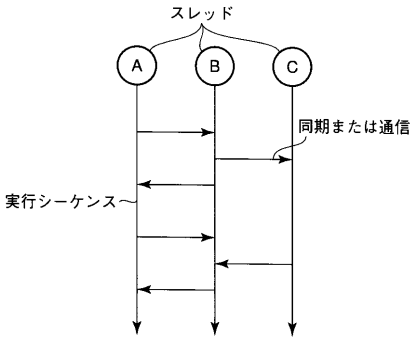

【図２】

図2



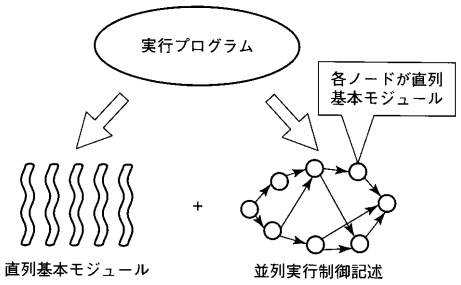

【図３】

図3



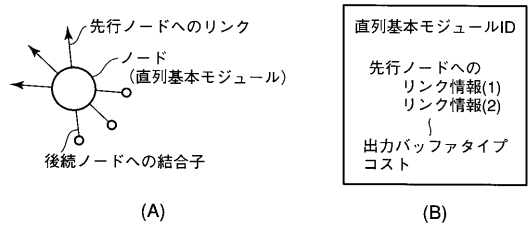

【図４】

図4



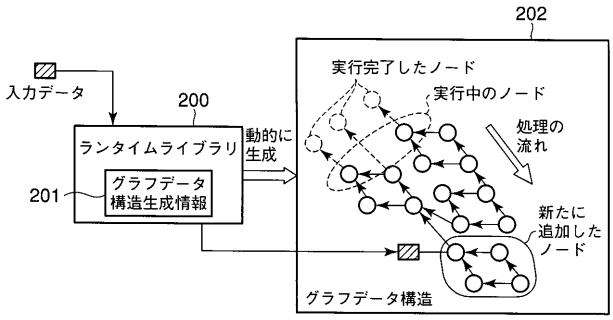
【図５】

図5



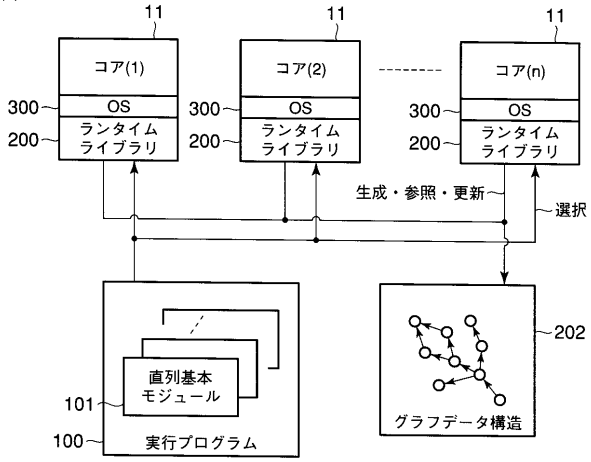

【図６】

図6



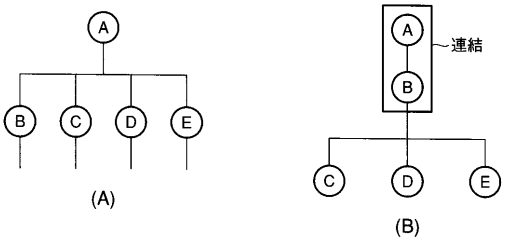

【図７】

図7



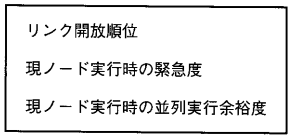

【図８】

図8



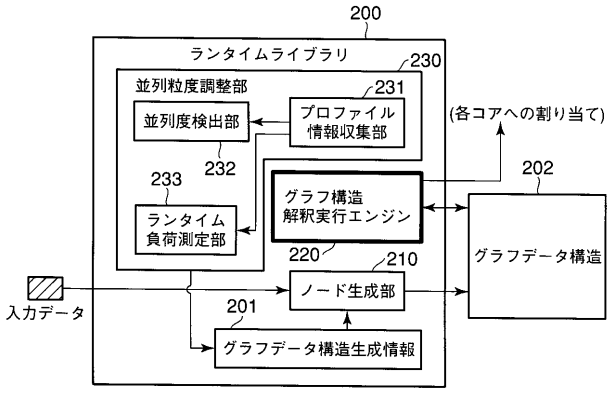

【図９】

図9



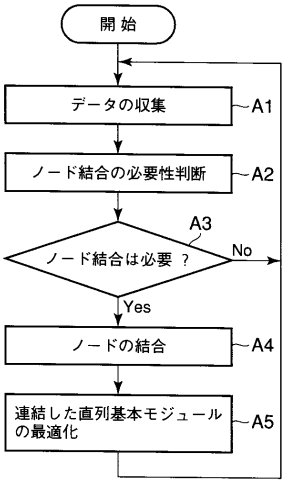

【図１０】

図10




【図１１】

図11



フロントページの続き

(74)代理人 100084618
弁理士 村松 貞男
(74)代理人 100092196
弁理士 橋本 良郎
(72)発明者 境 隆二
東京都港区芝浦一丁目１番１号 株式会社東芝内