(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2005-293585
(P2005-293585A)
(43)公開日　平成17年10月20日(2005.10.20)

(51)Int.Cl.[7]	F I	テーマコード（参考）
G06F 9/48	G06F 9/46 452Z	

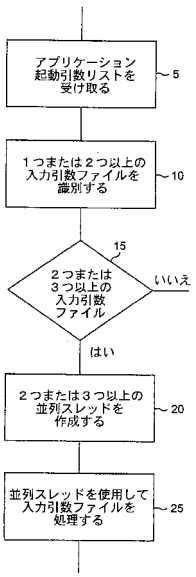
審査請求 有　請求項の数 14　ＯＬ　（全 20 頁）

(21)出願番号　特願2005-96692(P2005-96692)
(22)出願日　平成17年3月30日(2005.3.30)
(31)優先権主張番号　10/816086
(32)優先日　平成16年3月31日(2004.3.31)
(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　２０５５５　ステイト　ハイウェイ　２４９
(74)代理人　110000039
特許業務法人アイ・ピー・エス
(72)発明者　メーメット・ムサ
アメリカ合衆国カリフォルニア州サンノゼ　ベックドライヴ＃シー１６１３

(54)【発明の名称】コンピュータアプリケーションの性能を向上させる方法および装置

(57)【要約】
【課題】　コンピュータアプリケーションの性能を向上させる方法および装置を提供する。
【解決手段】　アプリケーション起動引数リストを受け取る（５）こと、該アプリケーション起動引数リストにおいて１つまたは２つ以上の入力引数ファイルを識別する（１０）こと、２つまたは３つ以上の入力引数ファイルが存在する（１５）場合に、２つまたは３つ以上の並列スレッドを作成する（２０）こと、および該並列スレッドを使用して前記入力引数ファイルを処理する（２５）こと、を含む、コンピュータアプリケーションの性能を向上させる方法とする。
【選択図】　図１



【特許請求の範囲】
【請求項１】
アプリケーション起動引数リストを受け取る（５）こと、
該アプリケーション起動引数リストにおいて１つまたは２つ以上の入力引数ファイルを識別する（１０）こと、
２つまたは３つ以上の入力引数ファイルが存在する（１５）場合に、２つまたは３つ以上の並列スレッドを作成する（２０）こと、および
該並列スレッドを使用して前記入力引数ファイルを処理する（２５）こと、
を含む、コンピュータアプリケーションの性能を向上させる方法。
【請求項２】
前記２つまたは３つ以上の並列スレッドを作成することは、
作成できる並列スレッドの最大数を決定する（３０）こと、および
該作成できる並列スレッドの前記最大数に従って多数の並列スレッドを作成する（３５）こと、
を含む、請求項１に記載のコンピュータアプリケーションの性能を向上させる方法。
【請求項３】
前記作成できる並列スレッドの最大数を決定することは、アクティブプロセッサの個数（４０）と、最大の並列スレッドについてのユーザごとのシステム制限（４５）と、ユーザ制御される最大並列スレッドの環境変数（５０）とに従って前記作成できる並列スレッドの最大数を決定することを含む、請求項２に記載のコンピュータアプリケーションの性能を向上させる方法。
【請求項４】
アプリケーション起動指令を受け取る（９０）こと、
該アプリケーション起動指令が向上対象の候補であるアプリケーションを指定するかどうかを判断する（９５）こと、
アプリケーション起動引数リストを受け取る（１００）こと、
前記アプリケーションが向上対象の候補であり、かつ、前記アプリケーション起動引数リストに複数の入力引数ファイルが含まれる（１０５）場合に、前記アプリケーションの２つまたは３つ以上の並列インスタンスを起動する（１１０）こと、および
各アプリケーションインスタンスに、前記アプリケーション起動引数リストに含まれる前記入力引数ファイルの対応する１つを含むインスタンティエーションアプリケーション起動引数リストを送る（１１５）こと、
を含む、コンピュータアプリケーションの性能を向上させる方法。
【請求項５】
前記アプリケーションが向上対象の候補であるかどうかを判断することは、前記アプリケーションが１つまたは２つ以上の候補アプリケーションの一覧に含まれるかどうかを判断する（１２５）ことを含む、請求項４に記載のコンピュータアプリケーションの性能を向上させる方法。
【請求項６】
前記アプリケーションの２つまたは３つ以上の並列インスタンスを起動することは、
作成できる並列スレッドの最大数を決定する（１５０）こと、および
該並列スレッドの最大数に従って前記アプリケーションの多数の並列インスタンスを起動する（１５５）こと、
を含む、請求項４に記載のコンピュータアプリケーションの性能を向上させる方法。
【請求項７】
前記作成できる並列スレッドの最大数を決定することは、
アクティブプロセッサの個数（４０）と、ユーザごとの最大並列スレッドシステム制限（４５）と、ユーザ制御される最大並列スレッドの環境変数（５０）との少なくとも１つに従って作成できる並列スレッドの個数を決定することを含む、
請求項６に記載のコンピュータアプリケーションの性能を向上させる方法。

【請求項８】
命令シーケンスを実行できるプロセッサ（６０５）と、
メモリ（６１５）と、
引数リストを受け取ることができるコンソール（６０１）と、
１つまたは２つ以上の入力ファイルを記憶でき、さらに、出力ストリームを記憶できるコンピュータ可読媒体（６１０）と、
前記メモリに記憶された命令シーケンスモジュールであって、
前記プロセッサによって実行されると、最低限、前記コンソール（６０１）が受け取った引数リストの１つまたは２つ以上の入力引数ファイル（２２０）を前記プロセッサに識別させる引数パーサモジュール（２１５）と、
前記プロセッサによって実行されると、最低限、前記プロセッサ（６０５）に、前記コンピュータ可読媒体（６１０）に記憶された入力ファイルに従って、前記コンピュータ可読媒体（６１０）へ出力ストリームを送信させる機能コアモジュール（２３０）と、
タスクマスタモジュール（２３３）であって、前記プロセッサによって実行されると、最低限、前記プロセッサに、
前記機能コアモジュールの１つまたは２つ以上のインスタンティエーションを作成させ、
前記プロセッサが前記引数パーサ（２１５）を実行すると、前記機能コアモジュールの対応するインスタンティエーション（２５０、２６０）へ、前記プロセッサによって識別された入力引数ファイルを送らせ、
代理プロセッサが前記機能コアモジュール（２３０）の各インスタンティエーションを実行することを行わせる、
タスクマスタモジュールと、
を含む命令シーケンスモジュールと、
を備えるファイル処理システム。
【請求項９】
前記タスクマスタモジュールは、最低限、前記プロセッサに、
作成できる並列スレッドの個数を決定させ（３０）、
該決定された並列スレッドの個数に従って、前記機能コアモジュールの多数のインスタンスを作成させる（３５）、
ことによって、最低限、前記プロセッサに、前記機能コアモジュールの１つまたは２つ以上のインスタンティエーションを作成させる、請求項８に記載のファイル処理システム。
【請求項１０】
前記タスクマスタモジュールは、最低限、前記プロセッサに、システムのアクティブプロセッサの個数（３１５）と、ユーザごとの最大並列スレッドシステム制限（３２５）と、ユーザ制御される最大スレッドの環境状態変数（３３０）との少なくとも１つに従って並列スレッドの個数を決定させることによって、最低限、前記プロセッサに、作成できる並列スレッドの個数を決定させる、請求項９に記載のファイル処理システム。
【請求項１１】
命令シーケンスを実行できるプロセッサ（６０５）と、
メモリ（６１５）と、
引数リストを含むアプリケーション起動指令を受け取ることができるコンソール（６０１）と、
１つまたは２つ以上の入力ファイルを記憶でき、さらに、出力ストリームを記憶できるコンピュータ可読媒体（６１０）と、
前記メモリに記憶された命令シーケンスモジュールであって、
コマンドラインパーサモジュール（２００）であって、前記プロセッサによって実行されると、最低限、前記プロセッサに、
受け取った起動指令（２０５）において、実行されるアプリケーションを識別させ、

該識別されたアプリケーションが向上対象の候補であるかどうかを判断させ、
前記アプリケーション起動指令に含まれる引数リストにおいて１つまたは２つ以上の入力引数ファイル（２２０）を識別させ、
前記識別されたアプリケーションが向上対象の候補であり、かつ、対応するインスタンティエーション引数リストが前記入力引数ファイルの１つを含む前記引数リストに２つまたは３つ以上の入力引数ファイルが存在する場合に、複数のロード指令（２７０）および対応するインスタンティエーション引数リストをタスクエグゼクティブ用に生成させる、
コマンドラインパーサモジュールと、
タスクエグゼクティブモジュール（４３５）であって、前記プロセッサによって実行されると、最低限、前記プロセッサに、
前記プロセッサによって実行されると、最低限、前記プロセッサに、前記コンピュータ可読媒体に記憶された入力ファイルに従って出力ストリーム（３８０）を前記コンピュータ可読媒体へ送信させるアプリケーションモジュール（２１０）を、前記複数のロード指令および対応するインスタンティエーション引数リストに従って、前記メモリにロードさせ、
前記コマンドラインパーサによって生成された対応するインスタンティエーション引数リストを前記アプリケーションモジュールへ送信させ、
代理プロセッサが前記アプリケーションモジュールを実行することを行わせる、
タスクエグゼクティブモジュールと、
を含む命令シーケンスモジュールと、
を備えるファイル処理システム。
【請求項１２】
前記コマンドパーサモジュールは、最低限、前記プロセッサに、前記識別されたアプリケーションが事前に設定された候補アプリケーションの一覧に含まれるかどうかを判断させることによって、前記識別されたアプリケーションが向上対象の候補であるかどうかを前記プロセッサに判断させる、請求項１１に記載のファイル処理システム。
【請求項１３】
前記コマンドパーサモジュールは、最低限、前記プロセッサに、
作成できる並列スレッドの最大数を決定させ（３０）、
該決定された並列スレッドの最大数に従って、多数のロード指令および対応するインスタンティエーション引数リストを生成させる（３５）、
ことによって、複数のロード指令および対応するインスタンティエーション引数リストを前記プロセッサに生成させる、請求項１１に記載のファイル処理システム。
【請求項１４】
前記コマンドパーサモジュールは、最低限、前記プロセッサに、システムのアクティブプロセッサの個数（３１５）と、ユーザごとの最大並列スレッドシステム制限（３２５）と、ユーザ制御される最大スレッドの環境状態変数（３３０）との少なくとも１つに従って並列スレッドの個数を決定させることによって、最低限、前記プロセッサに、作成できる並列スレッドの最大個数を決定させる、請求項１３に記載のファイル処理システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コンピュータアプリケーションの性能を向上させる方法および装置に関する。
【背景技術】
【０００２】
かつて、コンピュータアプリケーションは、大きなメインフレームシステムで実行され、パンチカード等の原始的な媒体からメモリにロードされていた。その早期の時代、コンピュータは、大部分が逐次的な方法で使用されていた。換言すると、１つのコンピュータ

アプリケーションがロードされ、次いで実行されることになっていた。そのアプリケーションが終了すると、次に、新たなアプリケーションをコンピュータにロードし、その後、実行することができた。このような古いコンピュータの歴史を持ち出すことはかなり奇異に見えるかもしれないが、多くのコンピュータアプリケーションは、依然として逐次的な方法で実行されていることは注目に値する。コンピュータプログラムのこの古風な実行方法は、今日、アメリカのほとんどすべての机上にコンピュータを見ることができる時代であっても続いている。

【０００３】

理解するのがさらに困難なことは、コンピュータアプリケーションの逐次実行の古いパラダイムが、多数のプロセッサが同時並列に動作する強力なサーバにおいても継続しているということである。これらの並列プロセッサマシンでは、コンピュータアプリケーションの実行は、依然としてシリアルな方法で行われる。

【０００４】

Ｕｎｉｘ（登録商標）ベースのコンピュータ環境では、コンピュータアプリケーションは、一般に、新たなコマンドラインがユーザコンソールから受け付けられるごとに実行される。この例では、ユーザがコマンドラインを入力する。それに応答して、オペレーティングシステム（Ｕｎｉｘのジャーゴンではシェルとして知られている）に含まれるコマンドエグゼクティブ（command executive）がそのコマンドラインを解析する。コマンドラインの解析が成功すると、オペレーティングシステムに含まれるコマンドエグゼクティブは、コンピュータアプリケーションの実行可能イメージをワーキングメモリにロードする。次に、コマンドエグゼクティブは、ワーキングメモリにそのようにロードされたコンピュータアプリケーションを計算システムのプロセッサに実行させる。実行アプリケーションは、多くの場合、アプリケーションのインスタンティエーション（instantiation）（またはインスタンス）と呼ばれる。

【０００５】

コンピュータアプリケーションの複数のインスタンティエーションを並列に実行できる例が存在する。例えば、Ｕｎｉｘオペレーティングシステムは、バックグラウンド実行モードを提供する。バックグラウンド実行モードを使用すると、ユーザは、コンピュータアプリケーションの数個のインスタンスをインスタンス化することができる。次いで、通常、計算システムに含まれる個々の並列プロセッサに、メモリにロードされた１つまたは２つ以上の実行可能イメージを実行させることによって、これらのインスタンスを並列に実行することが可能になる。非常に単純化した観点からすると、コンピュータアプリケーションの個々の並列インスタンスは、このタイプのバックグラウンド実行モードを使用して起動することができる。

【発明の開示】

【発明が解決しようとする課題】

【０００６】

バックグラウンド実行に関連する１つの問題は、コンピュータアプリケーションの実行が、依然として、このように起動された並列インスタンティエーションごとに、ユーザによる個々のコマンドラインの入力を必要とするということである。このタイプのバックグラウンド実行に関連した別の問題は、各インスタンティエーションの実行順序を制御できないということである。これは、コンピュータアプリケーションが入力ファイルに作用して、コンピュータアプリケーションが作成する出力ファイルをユーザの所望の順序で出力しなければならないときに大問題となる。例えば、出力デバイスに印刷出力を指示するためにバックグラウンドモードでプリント（ｐｒｉｎｔ）コマンドを起動することがある。コンピュータユーザが所望のシーケンスで印刷出力を配列したいが、バックグラウンドで実行される印刷アプリケーションの各インスタンスは、ユーザが入力した個々のバックグラウンド印刷コマンドの順序とは関係なく出力を印刷するようにオペレーティングシステムによって許可される場合に生じるフラストレーションを想像されたい。

【課題を解決するための手段】

【０００７】
コンピュータアプリケーションの性能を向上させる方法および装置は、アプリケーション起動引数リストを受け取ることを含む。１つまたは２つ以上の入力引数ファイルが、引数リストにおいて識別される。２つまたは３つ以上の入力引数ファイルが引数リストに存在する場合に、２つまたは３つ以上の並列スレッドが作成される。次いで、入力引数ファイルが、並列スレッドを使用して処理される。
【０００８】
以下では、添付図面および図と共にいくつかの代替的な実施の形態を説明する。添付図面および図において、同じ数字は同じ要素を示す。
【発明を実施するための最良の形態】
【０００９】
図１は、コンピュータアプリケーションの性能を向上させる方法の一代表的な実施の形態を示すフロー図である。この代表的な方法によると、アプリケーションが最初にアプリケーション起動引数リストを受け取ると（ステップ５）、コンピュータアプリケーションの性能が向上される。１つまたは２つ以上の入力引数ファイルは、通常、起動引数リストにおいて識別される（ステップ１０）。２つまたは３つ以上の入力引数ファイルが起動引数リストにおいて識別されると（ステップ１５）、２つまたは３つ以上の並列スレッドが作成される（ステップ２０）。次いで、入力引数ファイルが、並列処理スレッドによって処理される（ステップ２５）。例えば、個々の１つのスレッドが、通常、各入力引数ファイルについて作成される。本方法の一変形によると、入力ファイル引数は、任意の方法で並列スレッドの１つに割り当てられることに留意すべきである。例えば、入力ファイル引数は、ランダムに並列スレッドに割り当てることができる。
【００１０】
コンピュータアプリケーションの性能を向上させるこの代表的な方法は、コンピュータプログラムで具現化することができる。実際には、本方法を具現化するコンピュータアプリケーションは、通常、或る形式のタスクマネージャを含む。このタスクマネージャは、アプリケーションが受け取った起動引数リストにおいて識別された各入力引数ファイルについて処理スレッドを作成することができるものである。このようなコンピュータプログラムの一例は、コマンドラインアプリケーションである。このようなコンピュータプログラムのさらに別の例は、グラフィカルユーザインタフェースをサポートする汎用アプリケーションである。したがって、本方法は、コマンドラインアプリケーションおよびグラフィカルユーザインタフェース（ＧＵＩ）アプリケーションの双方に適用することができる。タスクマネージャは、必ずしも、処理スレッドを直接作成できない場合があることに留意すべきである。逆に、タスクマネージャは、オペレーティングシステムが提供する処理スレッド管理機構と相互作用することができる。
【００１１】
コマンドラインアプリケーションは、通常、コンピュータ可読媒体にファイルとして記憶される。オペレーティングシステム（例えば、ＵＮＩＸ）は、通常、コマンドラインパーサを含む。このコマンドラインパーサは、コンソールから英数字文字列を受け取ることによって人間のユーザと対話する。一例の実施の形態によると、コマンドラインパーサは、コンピュータ可読媒体に記憶された実行可能ファイルの名前を識別する。これは、受け取った英数字文字列をスキャンすることによって行われる。受け取った英数字文字列のスキャンは、通常、事前に設定された或る語彙フォーマットに従って行われる。コマンドラインパーサは、次に、コンソールから受け取った英数字文字列に含まれ得る付加引数を識別する。コマンドラインパーサは、タスクエグゼクティブに実行可能ファイルの名前を渡す。
【００１２】
一例の実施の形態によると、タスクエグゼクティブは、コンピュータ可読媒体に記憶されている識別されたファイルの実行可能イメージをメモリにロードする。実行可能イメージがメモリにロードされると、タスクエグゼクティブは、コンピュータシステムのプロセ

ッサに、実行可能イメージに含まれる命令を実行させる。また、オペレーティングシステムは、プロセッサが、メモリにロードされた実行可能イメージの実行を開始すると、コマンドラインパーサによって識別されたあらゆる付加引数にアクセスできるようにする。この説明は、ＵＮＩＸのようなオペレーティングシステムを表すが、本方法は、他の形式のコマンドラインプログラムの実行が利用される場合にも適用することができる。したがって、本明細書で提供されるこの説明の多くが、ＵＮＩＸのようなオペレーティングシステムに適用可能であっても、本明細書に添付の特許請求の範囲の真の範囲および精神は、さまざまな実施の形態を含むように意図されている。このようなさまざまな実施の形態は、必ずしも、ＵＮＩＸ互換性を有する必要はない。

【００１３】

多くのコマンドラインアプリケーションが、人間のユーザがコマンドラインの引数として指定した入力ファイルに対して作用する。例えば、「ｐｒｉｎｔ（印刷）」は、ユーザがファイルの内容を出力デバイス（例えば、プリンタ）に送信できる一般的なコマンドラインアプリケーションである。この説明中、パーセント文字（％）は、オペレーティングシステムのプロンプトを表すのに使用される。ユーザは、オペレーティングシステムのプロンプトに対してコマンドで応答することが予想される。ユーザが入力したコマンドは、次に、コマンドラインパーサによってスキャンされる。ｐｒｉｎｔコマンドラインアプリケーションについて、一般的なコマンドラインは以下のように表すことができる。

（ａ）　％ｐｒｉｎｔ　ファイル１

【００１４】

したがって、コマンド例（ａ）は、「ファイル１」の名前を有するファイルの内容の印刷を望んでいるユーザによって入力される。このｐｒｉｎｔコマンドが実行されると、ユーザは、次に、新たなコマンドの入力を促される。ユーザが数個の異なるファイルの内容の印刷を望んでいる場合、ユーザは、コマンド（ａ）として紹介したコマンドフォーマットに従って、対応する個数のｐｒｉｎｔコマンドを入力する必要がある。通常、ユーザ対話は、待ち状態のコマンドが完全に実行されるまで一時停止される。したがって、ユーザは、先に入力したコマンドが完全に実行されるとすぐに新たなコマンドプロンプトを待って、これらのコマンドを逐次入力する必要がある。

【００１５】

本方法によると、新たなコマンドラインフォーマットは、ｐｒｉｎｔコマンドを高度化したものによってサポートすることができる。例えば、新たなコマンドラインフォーマットは、以下のコマンド（ｂ）で紹介するように、複数の入力ファイル引数が続くコマンド名（例えば「ｐｒｉｎｔ」）を含む。したがって、この新たなフォーマットに対応する一般的なコマンドラインは、以下のように表すことができる。

（ｂ）　％ｐｒｉｎｔ　ファイル１　ファイル２　ファイル３

【００１６】

この新たなコマンドラインフォーマットで提供されるコマンドを解釈することによって、本方法によるｐｒｉｎｔコマンドが実行される。ｐｒｉｎｔコマンドは、コンピュータ可読媒体のファイルに記憶された１つまたは２つ以上の命令シーケンスとして具現化されて、メモリにロードされ、プロセッサにそのコンピュータプログラムを実行させる。プロセッサは、ｐｒｉｎｔコマンドを具現化した命令シーケンスの実行を開始すると、複数の入力ファイル引数を受け取る。コマンド（ｂ）によって表される例示の使用ケースによると、入力ファイルの引数は、「ファイル１」、「ファイル２」、および「ファイル３」となる。本方法によると、プロセッサは、ｐｒｉｎｔコマンドを具現化する命令シーケンスの実行を続けるにつれて、複数の並列処理スレッドを作成する。各入力ファイル引数は、次に、対応する並列処理スレッドによって処理される。「ｐｒｉｎｔ」コマンドは、本方法を具現化するコマンドラインアプリケーションの一例にすぎないことに留意すべきである。他のコマンドラインアプリケーションには、ａｓｍ（アセンブラ）、ｃｃ（「ｃ」言語コンパイラ）、ｐｃ（Ｐａｓｃａｌ言語コンパイラ）、ｌｐ（ラインプリンタユーティリティ）、およびｚｉｐ（ファイル圧縮ユーティリティ）が含まれ得るが、必ずしもこれ

らに限定されるものではない。本方法は広く適用することができ、コマンドラインアプリケーションのいずれの例も、本明細書では、例示のために提示されており、本明細書に添付の特許請求の範囲を限定することを意図するものでないことにさらに留意すべきである。また、本明細書に添付の特許請求の範囲は、例示の目的で本明細書に提示したコマンドラインのあらゆる特定の例（例えば、上記のコマンド（ａ）およびコマンド（ｂ））の構造によっても、また内容によっても限定されるものでもない。

【００１７】

図２は、２つまたは３つ以上の並列処理スレッドを作成する方法の一例の実施の形態を示すフロー図である。プロセッサは、コンピュータアプリケーションを具現化する１つまたは２つ以上のさまざまな命令シーケンスを実行すると、起動引数リストにおいて２つまたは３つ以上の入力ファイル引数が識別される場合に、２つまたは３つ以上の並列処理スレッドを作成する。本方法の代替的な一例の実施の形態によると、作成できる並列処理スレッドの最大数を決定することによって、２つまたは３つ以上の並列処理スレッドの作成が行われる（ステップ３０）。次に、多数の並列処理スレッドが、決定された最大数の並列処理スレッドに従って作成される（ステップ３５）。本方法を具現化するどの特定のコンピュータアプリケーションもコア機能を含む。このコア機能は、並列処理スレッドのそれぞれに導入され、処理スレッドのそれぞれがアプリケーションのコア機能を実行することを可能にする。処理スレッドが本方法を具現化するコンピュータアプリケーションの独立したインスタンティエーションであると誤ってみなされる可能性があることに留意することは重要である。より適切な観点は、各処理スレッドを、通常ならば複数の入力引数ファイルを受け付けることができないアプリケーションに含まれるコア機能のインスタンティエーションとみなすことである。

【００１８】

各処理スレッドは、一例の実施の形態によると、オペレーティングシステムにスレッド割り当てを要求することによって設定される。このようなスレッド割り当ては、この例によると、１つまたは２つ以上の命令シーケンスを記憶するのに使用できるメモリの割り当てを含む。したがって、コア機能のインスタンティエーションの作成は、割り当てられたメモリに、コア機能を具現化する１つまたは２つ以上の命令シーケンスをロードし、次いで、コア機能の処理資源をスケジューリングするようにオペレーティングシステムに要求することによって行われる。したがって、プロセッサがコア機能の命令シーケンスの実行を開始すると、コンピュータアプリケーションのコア機能が実現される。

【００１９】

図３は、並列処理スレッドの最大数を決定する方法の代替的な実施の形態を示すフロー図である。本方法の一代替的な実施の形態によると、作成できる並列処理スレッドの最大数は、計算システムで利用可能なアクティブプロセッサの個数に従って決定される（ステップ４０）。現代、ローエンドのワークステーションおよびサーバでさえも、複数の並列プロセッサを含むことができる。マルチプロセッサ計算システムでは、処理資源は、通常、オペレーティングシステムによって管理される。したがって、オペレーティングシステムに含まれるタスクマネージャは、計算システム内のプロセッサのアクティビティを認識する状態にある。一例示の実施の形態によると、計算システムで利用可能なアクティブプロセッサの個数は、オペレーティングシステムによって管理される環境変数を調べることによって決定される。さらに別の代替例の実施の形態によると、作成できる並列処理スレッドの最大数は、特定の環境内で作成できる並列スレッドの最大数についてのユーザごとのシステム制限に従って設定される（ステップ４５）。オペレーティングシステムの一例は、計算システムを使用できるユーザごとに個々のパーティションを提供する。これら個々のパーティションのそれぞれは、オペレーティングシステムが保持するさまざまな環境変数によって管理される。このような１つの環境変数は、特権ユーザ（ＵＮＩＸのようなシステムでは、これはルートとして一般に知られている）が設定するユーザごとのシステム制限を含むことができる。したがって、特権ユーザは、特定のパーティション内に作成できる並列処理スレッドの最大数を指定することができる。ユーザごとのシステム制限は

、一般に、不定期に変更されるだけである。通常、アプリケーションは、このようなユーザごとのシステム制限を変更することができない。さらに別の例示の代替的な実施の形態によると、ユーザ制御される最大並列スレッドの環境変数は、作成できる並列スレッド数を設定するのに使用される（ステップ５０）。オペレーティングシステムの一例の実施の形態は、最大並列スレッド（ＭＰＴ）環境変数として知られている環境変数を保持する。ＭＰＴ変数の実際の名前は、どのタイプのオペレーティングシステムが計算システムの制御に使用されるかに応じて変化し得る。例えば、ＵＮＩＸのようなオペレーティングシステムでは、ＭＰＴ変数は、スレッド最大数（ＭＡＸ＿ＮＵＭＢＥＲ＿ＯＦ＿ＴＨＲＥＡＤＳ）変数と呼ばれる。オペレーティングシステムのこの例の実施の形態によって、ユーザ（すなわち、特定のユーザパーティション内で実行されるアプリケーション）は、ＭＰＴ環境変数に記憶された値を変更することが可能になる。したがって、本方法のこの例示の代替的な実施の形態は、このようなＭＰＴ環境変数に従って作成できる並列処理スレッドの最大数を設定する。本方法のこれらのさまざまな代替的な実施の形態は、例示のために本明細書に提示されるものであり、本明細書に添付の特許請求の範囲を限定するためのものと意図するものではない。

【００２０】

図４は、並列処理スレッドの最大数を決定する方法の他の代替的な実施の形態を示すフロー図である。このような他の代替的な一実施の形態によると、オペレーティングシステムによって保持される１つまたは２つ以上の環境変数（ステップ５５）は、本方法を具現化するアプリケーションを作成することを可能にする最大数の並列処理スレッドを設定するのに利用される。オペレーティングシステムの任意の特定の実施の形態が保持できるさまざまなタイプの環境変数のすべてを列挙することは困難である。しかしながら、本明細書に添付の特許請求の範囲は、アプリケーションが作成できる並列処理スレッドの最大量が、特定のオペレーティングシステムが保持するさまざまなタイプの環境変数の１つまたは２つ以上に従って設定される派生的な方法を包含するように意図されている。別の代替的な実施の形態によると、本方法を具現化するアプリケーションは、アプリケーション起動時にアプリケーションが受け取るアプリケーション起動引数リストに含まれる最大スレッド指示子を引数として受け取る（ステップ６０）。

【００２１】

図５は、２つまたは３つ以上の並列処理スレッドを作成する方法の一例示の代替的な実施の形態を示すフロー図である。本方法のこの代替的な実施の形態によると、入力引数ファイル数が決定される（ステップ６５）。本方法を具現化する一アプリケーションは、一代替的な実施の形態によると、アプリケーション起動引数リストに含まれる入力引数ファイルの個数を単にカウントする。一代替的な実施の形態によると、アプリケーションは、オペレーティングシステムに依拠して、入力引数ファイル数のカウントを提供する（例えば、ＵＮＩＸは引数カウント変数「ａｒｇｃ」を提供する）ことができる。入力引数ファイルの個数が決定されると、この代替的な方法は、対応する個数の並列処理スレッドを作成する（ステップ７０）。並列処理スレッドは、この例示の代替的な実施の形態によると、図２に関して上記に提供した説明に従って作成される。

【００２２】

図６は、並列処理スレッドを使用して入力ファイルを処理する方法の一例の実施の形態を示すフロー図である。コンピュータアプリケーションのコア機能のインスタンティエーション（すなわち、複数の並列処理スレッドの１つ）が、コンピュータシステムの処理資源によって実行されると、そのインスタンティエーションは、通常、プロセッサに、入力ファイルから情報をフェッチさせる。

【００２３】

したがって、並列処理スレッドを使用して入力ファイルを処理するこの例の方法は、コンピュータアプリケーションが作成するさまざまな並列処理スレッドに入力引数ファイルを割り当てる（ステップ７５）。アプリケーションが受け取る引数リストに含まれる各ファイル引数は、並列処理スレッドの１つにディスパッチされる。並列処理スレッドは、ど

のファイルを入力として使用するかを決定する手段として、自身が受け取ったファイル引数を使用する。入力引数ファイルを処理スレッドに割り当てることは、本方法の一変形によると、集中プロセスによって行われる。例えば、本方法を具現化するコンピュータアプリケーションは、タスクマネージャプロセスを含むことができる。このようなタスクマネージャプロセスは、一代替的な実施の形態によると、特定の入力引数ファイルに作用するように特定の並列処理スレッドに指示する。本方法のさらに別の代替的な変形によると、入力引数ファイルの割り当ては分配方法で行われる。例えば、本方法を具現化するコンピュータアプリケーションは、未処理の入力ファイル引数を処理キューに置くことができる。したがって、特定の並列処理スレッドは、並列処理スレッドが処理キューから取り出すファイル引数に作用することができる。

【００２４】

さらに別の例の代替的な方法によると、特定の並列処理スレッドによって生成される出力は、起動時にアプリケーションが受け取ったアプリケーション起動引数リストに含まれる引数シーケンスと一致したシーケンスで生成されない場合がある。この問題を軽減するために、この代替的な方法は、複数の並列処理スレッドによって生成された出力を収集し（ステップ８０）、アプリケーション起動引数リストに含まれる入力ファイル引数の順序に従って出力を編成する（ステップ８５）。

【００２５】

図７は、コンピュータアプリケーションの性能を向上させる方法の一代替的な実施の形態を示すフロー図である。コンピュータアプリケーションの設計に組み込むことができる本方法の実施の形態をこれまで説明してきた。この代替的な方法は、より適切にオペレーティングシステムに組み込まれる。例えば、コマンドラインパーサが、コンピュータアプリケーションの性能を向上させるこの代替的な方法を具現化することができる。一例の方法によると、コンピュータアプリケーションの性能は、アプリケーション起動指令を受け取る（ステップ９０）ことによって向上される。一例の方法によると、起動指令は、コンソールから英数字文字列として受け取られる。アプリケーション起動指令は、本方法の一変形によると、コンピュータアプリケーションの名前（例えば、「ｐｒｉｎｔ」）および起動引数リストを含む。この代替的な方法によると、一定のコンピュータアプリケーションは、「候補アプリケーション」とみなされる。候補アプリケーションは、複数の並列インスタンティエーションにおいて実行できるアプリケーションを含み、アプリケーションの各並列インスタンティエーションは、特定の入力引数ファイルに作用することができる。したがって、本方法は、アプリケーション起動指令で指定されたアプリケーションがこのような候補アプリケーションであるかどうかを判断する（ステップ９５）。

【００２６】

図８は、アプリケーションが向上対象の候補であるかどうかを判断する方法の一例の実施の形態を示すフロー図である。この例の方法によると、アプリケーション起動指令で指定されたアプリケーションが、候補アプリケーションの一覧と比較される。指定されたアプリケーションが候補アプリケーションの一覧に発見されると（ステップ１２５）、指定されたアプリケーションは、候補アプリケーションであると宣言される（ステップ１３０）。すでに説明したように、候補アプリケーションは、複数の並列インスタンティエーションにおいて実行できるアプリケーションを含む。一例の実施の形態によると、コマンドラインパーサは、指定されたアプリケーションを候補アプリケーションの一覧と比較する機構を提供する。

【００２７】

図７は、さらに、コンピュータアプリケーションの性能を向上させるこの例の方法に従って、アプリケーション起動引数リストが受け取られる（ステップ１００）ことも示している。一例の実施の形態によると、コマンドラインパーサは、アプリケーション起動指令に含まれる入力ファイル引数を識別するために、アプリケーション起動指令をスキャンすることができる。アプリケーション起動指令に含まれる２つまたは３つ以上の入力ファイル引数が存在する場合（ステップ１０５）、この例の方法は、アプリケーションの並列イ

ンスタンティエーションを起動する（ステップ１１０）。引数リストの一部は、アプリケーションの各インスタンティエーションへ送られる（ステップ１１５）。アプリケーションの各インスタンティエーションは、特定の入力ファイル引数に作用することが可能になる。この説明から理解できるように、多種多様なコンピュータアプリケーションをこのようにして開始することができる。本方法を組み込むために、コンピュータアプリケーション自体を変更する必要はない。オペレーティングシステム（例えば、オペレーティングシステムに含まれるコマンドラインパーサ）は、コンピュータアプリケーションの性能を向上させるこの代替的な方法を具現化することができる。

【００２８】

図９は、コンピュータアプリケーションの並列インスタンティエーションを起動する一例の方法を示すフロー図である。コンピュータアプリケーションの並列インスタンティエーションを起動するこの代替例の方法によると、並列処理スレッドの最大数が決定される（ステップ１５０）。この並列処理スレッドの最大数は、インスタンス化されるコンピュータアプリケーションの並列インスタンティエーションの個数を制限するのに使用される（ステップ１５５）。並列処理スレッドの最大数の決定は、本明細書で前述したさまざまな方法および技法を使用して行うことができる。

【００２９】

図１０は、入力引数ファイルを処理するシステムの一例の実施の形態を示すブロック図である。この例の実施の形態によると、システム（例えば、計算システム）６００は、１つまたは２つ以上のプロセッサ６０５、コンピュータ可読媒体６１０、およびメモリ６１５を備える。一代替的な実施の形態によると、システム６００は、さらに、引数リストを受け取るコンソール６０１も含む。また、システム６００のこの例の実施の形態には、１つまたは２つ以上の機能モジュールも含まれる。機能モジュールは、通常、命令シーケンスとして具現化される。機能モジュールを実施する命令シーケンスは、一代替的な実施の形態によると、メモリ６１５に記憶される。用語「最低限、プロセッサにさせる（minimally causes the processor）」およびその変形したものは、プロセッサ６０５が特定の機能モジュール（すなわち、命令シーケンス）を実行する際に、プロセッサ６０５によって実行される機能を制限なく列挙したものとして機能するように意図されていると、読み手には助言をしておく。したがって、添付の特許請求の範囲に規定されたものに加えて、特定の機能モジュールがプロセッサ６０５に機能を実行させる一実施の形態は、本明細書に添付の特許請求の範囲に含まれることになる。

【００３０】

本方法に従って入力引数ファイルの処理を可能にする、これまでに説明した機能モジュール（すなわち、それら機能モジュールに対応する命令シーケンス）は、一代替的な実施の形態によると、コンピュータ可読媒体上に与えられる。このような媒体の例には、ランダムアクセスメモリ、読み出し専用メモリ（ＲＯＭ）、コンパクトディスクＲＯＭ（ＣＤＲＯＭ）、フロッピィ（登録商標）ディスク、ハードディスクドライブ、磁気テープ、およびデジタル多用途ディスク（ＤＶＤ）が含まれるが、これらに限定されるものではない。このようなコンピュータ可読媒体は、単独でまたは組み合わせて、スタンドアロン製品を構成することができ、本明細書で提示した技法および教示に従って入力引数ファイルを処理できるデバイスに汎用計算プラットフォームを変換するのに使用することができる。したがって、本明細書に添付の特許請求の範囲は、本方法および本明細書で説明した教示のすべての実行を可能にするような命令シーケンスで与えられるこのようなコンピュータ可読媒体を含むことになる。

【００３１】

図１０は、さらに、システム６００の一代替的な実施の形態に従って、アプリケーション２１０と呼ばれる機能モジュールがメモリ６１５に含まれることも示している。一代替的な実施の形態によると、アプリケーション２１０は、引数パーサ２１５、タスクマスタ２３３、および機能コア２３０と呼ばれる他の機能モジュールを含んでいる。

【００３２】

図１１は、入力引数ファイルを処理するシステムの一例示の実施の形態のオペレーションを示すデータフロー図である。この例示の実施の形態によると、プロセッサ６０５は、コマンドパーサ２００を含むオペレーティングシステム３００を実行する。コマンドパーサは、時に、シェルとして知られている。シェル２００は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に引数リストを受け取らせる。通常、シェル２００は、最低限、プロセッサ６０５にユーザからのコマンドも受け取らせる。このようなコマンドは、通常、ユーザが実行を望むアプリケーションの名前を含む。したがって、シェル２００は、特定のアプリケーション、例えば、この例示の実施の形態のアプリケーション２１０を識別すると、さらに、プロセッサ６０５に、オペレーティングシステム３００に含まれるタスクエグゼクティブ６３０も実行させる。シェル２００とタスクエグゼクティブ６３０との間のこの相互作用は、単なる例示の目的で本明細書に提示されるものであり、本明細書に添付の特許請求の範囲を限定するためのものでないことに留意すべきである。実際には、本システム６００は、コマンドの入力およびコマンドに従ったその後の特定のアプリケーションの実行をユーザに可能にするさまざまなメカニズムを含むことができる。この説明全体を通じて留意すべき重要なことは、この例示の実施の形態によると、プロセッサ６０５がメモリ６１５に記憶されたアプリケーション２１０を実行するということである。プロセッサ６０５は、アプリケーション２１０の実行を開始すると、当該アプリケーション２１０によって、最低限、引数リストを受け取る（２０５）ようにされる。プロセッサ６０５は、直接または参照（例えば、ポインタ）によって引数リストを受け取る。

【００３３】

アプリケーション２１０のこの例示の実施の形態には、引数パーサ２１５が含まれる。引数パーサは、プロセッサ６０５によって実行されると、最低限、プロセッサに、シェル２００から受け取った（２０５）引数リストにおいて１つまたは２つ以上の入力引数ファイルを識別させる。引数パーサ２１５は、アプリケーション２１０の一代替的な実施の形態によると、最低限、プロセッサ６０５に、シェル２００から受け取った（２０５）引数リストから個々の入力ファイル引数を抽出させる。これらの入力ファイル引数は、ファイルリストバッファ２２０に記憶することができる。引数パーサのさらに別の代替的な実施の形態によると、引数パーサ２１５は、さらに、最低限、プロセッサ６０５に、引数リストから最大スレッド指示子を抽出させる。この最大スレッド指示子は、次に、アプリケーション２１０のこの例示の実施の形態に含まれるタスクマネージャ２３３に提供される（２６５）。

【００３４】

アプリケーション２１０のこの例示の実施の形態に含まれる機能コアモジュール２３０は、プロセッサ６０５によって実行されると、最低限、プロセッサに、コンピュータ可読媒体（ＣＲＭ）に記憶された入力ファイルに従って、出力ストリームをコンピュータ可読媒体へ送信させる。一般に、機能コアモジュール２３０は、アプリケーション２１０のさまざまな実施の形態に応じて変化する。機能コアモジュール２３０のこのような変化は、通常、本明細書で説明した例示の実施の形態が表すタイプのアプリケーション２１０に対応する。例えば、アプリケーション２１０が印刷アプリケーションである場合、機能コアモジュール２３０は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、入力ファイルの表示を出力デバイス（例えば、プリンタ）へ送信させる命令を含む。さらに別の例によると、アプリケーション２１０が、アセンブリ言語の命令文をバイナリファイルに変換するアセンブラである場合に、機能コアモジュール２３０は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、入力ファイルに含まれるアセンブリ言語の命令文をバイナリ命令シーケンスファイルに変換させる命令を含む。機能コアモジュール２３０のさまざまな実施の形態で明らかにすることができる変形の例は２つだけである。この説明にこれらの例を導入することが本明細書に添付の特許請求の範囲に適用できる方法および装置を示すためであることを強調することは重要である。機能コアモジュール２３０のこれらの例および他のこのような例は、添付の特許請求の範囲を

限定するためのものではない。

【００３５】

この例示の実施の形態のタスクマスタモジュール２３３は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、機能コアモジュール２３０の１つまたは２つ以上のインスタンティエーションを作成させ、プロセッサ６０５が引数パーサ２１５を実行した時にプロセッサ６０５によって識別された入力引数ファイルを機能コアモジュール２３０の対応するインスタンティエーションへ送信させる（２５０、２６０）。さらに、タスクマスタモジュール２３３は、最低限、プロセッサ６０５（または別の代理プロセッサ（assignee processor））に、機能コアモジュールの各インスタンティエーションを実行させる。一代替的な実施の形態によると、タスクマスタモジュール２３３は、オペレーティングシステム３００に含まれるタスクエグゼクティブ６３０にスレッド要求２４０をディスパッチすることによって、プロセッサ６０５にインスタンティエーションを作成させる。これに応じて、タスクマスタ２３３は、通常、ロードポインタ２４５を受け取る。次に、タスクマスタ２３３は、機能コアモジュール２３０をコアイメージとして取り出し、このコアイメージをロードポインタ２４５に従ってメモリにロードする。タスクマスタ２３３とタスクエグゼクティブ６３０との間の他の通信により、システムに含まれ得るいくつかの並列プロセッサの１つを使用して特定のインスタンティエーションの実行が可能になる。

【００３６】

また、図１１は、さらに、タスクマスタ２３３が、一代替的な実施の形態に従って、最低限、プロセッサ６０５に、最大スレッド数値２３５を受け取らせるタスクマスタ２３３も示している。この代替的な実施の形態によると、タスクマスタモジュール２３３は、まず、最低限、プロセッサ６０５に、システムに作成できる並列スレッドの最大数を決定させ、次いで、最低限、プロセッサ６０５に、決定した並列スレッドの最大数に応じて、機能コアモジュール２３０の多数のインスタンティエーションを作成させることによって、機能コアモジュール２３０の１つまたは２つ以上のインスタンティエーションをプロセッサ６０５に作成させる。さらに別の代替的な実施の形態によると、タスクマスタモジュール２３３は、最低限、プロセッサ６０５に、引数パーサ２１５から受け取った最大スレッド指示子２６５に従って機能コアモジュール２３０の多数のインスタンティエーションを作成させる。

【００３７】

図１２は、最大スレッド決定機能の一実施の形態のオペレーションを説明するデータフロー図である。さらに別の代替的な実施の形態によると、タスクマスタモジュール２３３は最大スレッド決定機能３０５を含む。最大スレッド決定機能３０５は、プロセッサ６０５によって実行されると、さまざまなシステム変数および／または環境変数を調べることによって、最低限、プロセッサ６０５に、作成できる並列スレッドの個数を決定させる。例えば、最大スレッド決定機能３０５の一代替的な実施の形態は、最低限、プロセッサ６０５に、プロセッサカウント３１０をオペレーティングシステム３００に要求することを行わせる。これに応じて、オペレーティングシステム３００は、計算システムで利用可能なアクティブプロセッサの個数の表示を提供する（３１５）。最大スレッド決定機能３０５は、次いで、最低限、プロセッサ６０５に、オペレーティングシステム３００が提供するアクティブプロセッサの個数３１５の表示を最大スレッド数２３５として設定させる。さらに別の代替的な実施の形態によると、最大スレッド決定機能３０５は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、オペレーティングシステム３００が提供するユーザごとの最大並列スレッド制限３２５に保持される値を最大スレッド数２３５として設定させる。さらに別の代替的な実施の形態によると、最大スレッド決定機能３０５は、最低限、プロセッサ６０５に、ユーザ制御される最大スレッド３３０の状態変数を、作成できるスレッドの最大数２３５の基礎として使用させる。そして、さらに別の代替例の実施の形態によると、プロセッサ６０５が最大スレッド決定機能３０５を実行すると、最低限、プロセッサ６０５に、オペレーティングシステム３００が提供できる

１つまたは２つ以上の他の環境変数３４５を使用させることが行われる。

【００３８】

図１３は、出力オーガナイザを含むアプリケーションの代替的な例示の実施の形態を示すデータフロー図である。上述したように、機能コアのさまざまな並列インスタンティエーションからの出力は、必ずしも、所望のシーケンスで生成されないことがある。したがって、アプリケーション２１０のこの代替的な例示の実施の形態は、さらに、出力オーガナイザモジュール３６０を備える。出力オーガナイザモジュール３６０は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、タスクマスタ２３３がインスタンス化した機能コア２３０の１つまたは２つ以上のインスタンティエーション（２８０、２８５）からの出力ファイル（３６５、３７０）を収集させる。各インスタンティエーションからの出力は、アプリケーション２１０が最初に起動された時にアプリケーション２１０が受け取った引数リストに含まれる入力ファイル引数の順序に従って編成される。一代替例の実施の形態によると、出力オーガナイザ３６０は、プロセッサ６０５が引数パーサ２１５を実行する時にプロセッサ６０５によって作成されたファイルリストバッファ２２０で発見されたファイル引数のシーケンスに従って出力を順序付ける。

【００３９】

また、図１０は、さらに別の代替的な実施の形態に従って、入力引数ファイルを処理するシステム６００が、１つまたは２つ以上のプロセッサ６０５、コンピュータ可読媒体６１０、およびメモリ６１５を備えることも示している。この代替的な実施の形態によると、メモリには、オペレーティングシステム４３０を含む命令シーケンスとして具現化される機能モジュールが含まれる。この代替的な実施の形態によると、オペレーティングシステムは、コマンドパーサ４００およびタスクエグゼクティブ４３５を含む。さらに別の代替的な実施の形態によると、オペレーティングシステムは、さらに、ファイルマネージャ４４０を含む。さらに別の代替的な実施の形態によると、ファイルを処理するシステム６００は、さらに、ユーザからアプリケーション起動指令を受け取るのに使用できるコンソール６０１を含む。

【００４０】

図１４は、入力引数ファイル処理システムの一代替的な実施の形態のオペレーションを示すデータフロー図である。この代替的な実施の形態のコマンドパーサ４００は、時にシェルと呼ばれ、ユーザと対話することができる。したがって、コマンドパーサ４００は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、起動指令４０５を受け取らせる。起動指令４０５は、通常、コンピュータ可読媒体（ＣＲＭ）４５０に記憶されたアプリケーションの名前を含む。さらに、コマンドパーサ４００は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、引数リスト４１０を受け取らせる。コマンドパーサ４００の一代替的な実施の形態によると、引数リスト４１０は、起動指令４０５に含まれることに留意すべきである。これらは、コマンドパーサ４００のさまざまな実施の形態の単なる例にすぎない。これらの例の実施の形態は、本明細書に添付の特許請求の範囲を限定するためのものではない。

【００４１】

この例の実施の形態のコマンドパーサ４００は、プロセッサ６０５によって実行されると、最低限、プロセッサに、起動指令４０５に含まれるアプリケーションを識別させる。コマンドパーサ４００は、さらに、最低限、プロセッサ６０５に、引数リスト４１０の１つまたは２つ以上の入力引数ファイルを識別させる。コマンドパーサ４００は、さらに、最低限、プロセッサ６０５に、複数のロード指令４２０および対応するインスタンティエーション引数リスト４２５を生成させる。これらは、次に、オペレーティングシステム４３０に含まれるタスクエグゼクティブ４３５に利用可能にされる。起動指令４０５で指定されたアプリケーションが向上対象の候補であり、かつ、２つまたは３つ以上の入力引数ファイルが引数リスト４１０に存在する場合に、コマンドパーサ４００は、複数のロード指令４２０および対応するインスタンティエーション引数リスト４２５を生成することに留意すべきである。コマンドパーサ４００の一代替的な実施の形態によると、コマンドパ

ーサ４００は、最低限、プロセッサ６０５に、起動指令４０５に指定されたアプリケーションが候補アプリケーションリスト４１５に含まれるかどうかを判断させることによって、最低限、プロセッサ６０５に、アプリケーションが向上対象の候補であるかどうかを判断させる。図に提示する候補アプリケーションのどの特定の例も添付の特許請求の範囲を限定するためのものでないことに留意すべきである。

【００４２】

さらに別の例の代替的な実施の形態によると、コマンドパーサ４００は、プロセッサ６０５によって実行されると、生成できる並列処理スレッドの最大数をまず決定することによって、最低限、プロセッサ６０５に、複数のロード指令４２０および対応するインスタンティエーション引数リスト４２５を生成させる。多数のロード指令４２０および対応するインスタンティエーション引数リスト４２５は、次に、並列処理スレッドの決定された最大数に従って生成される。コマンドパーサ４００のさまざまな代替的な実施の形態は、本明細書で提供した他の教示に従って並列処理スレッドの最大数を決定する。例えば、図１２について説明したような並列処理スレッドの最大数の決定を行うこのコマンドパーサ４００の実施の形態がある。したがって、この代替的な実施の形態のコマンドパーサ４００は、当該実施の形態に含まれる最大スレッド決定機能３０５から最大スレッド数２３５を受け取る。コマンドパーサ４００のさらに別の例の実施の形態は、引数リスト４１０に含まれる最大スレッド指示子に従って並列処理スレッドの最大数を設定する。この代替的な例の実施の形態は、起動指令４０５および引数リスト４１０をコマンドパーサ４００に運ぶ際に最大スレッド指示子を指定することをユーザに可能にする。

【００４３】

さらに別の代替例の実施の形態によると、コマンドパーサ４００は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、引数リスト４１０に含まれる入力ファイル引数の個数を決定させることによって、最低限、プロセッサ６０５に、複数のロード指令４２０および対応するインスタンティエーション引数リスト４２５を生成させる。次いで、多数のロード指令４２０および対応するインスタンティエーション引数リスト４２５が、入力ファイル引数の決定された個数に従って生成される。

【００４４】

この例の実施の形態のタスクエグゼクティブ４３５は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、ロード指令４２０に指定されたアプリケーションの個々のインスタンティエーションを作成させる。これは、一代替的な実施の形態によると、最低限、プロセッサ６０５に、コンピュータ可読媒体４５０からアプリケーションの実行可能イメージを取り出させる（４５５）ことによって行われる。通常、プロセッサ６０５は、オペレーティングシステム４３０の一代替的な実施の形態に含まれるファイルマネージャ４４０を実行する。したがって、タスクエグゼクティブ４３５は、最低限、プロセッサに、ファイルマネージャ４４０からアプリケーションの実行可能イメージを受け取らせ（４６０）、続いて、そのアプリケーションをメモリにロードさせる（４６５、４７０）。タスクエグゼクティブ４３５は、さらに、最低限、プロセッサ６０５に、各アプリケーションインスタンス（４８０、４８５）の実行をスケジューリングさせる。このような実行は、一代替例の実施の形態によると、代理プロセッサによって実行される。一代替的な実施の形態によると、ファイルを処理するシステム６００は複数のプロセッサ６０５を含む。したがって、この代替的な実施の形態のシステム６００に含まれるプロセッサ６０５の１つは、アプリケーションのインスタンス（４８０、４８５）を実行するように割り当てられる。図示した一実施の形態によると、このようなアプリケーションは、代理プロセッサによって実行されると、最低限、代理プロセッサに、出力ストリームをコンピュータ可読媒体４５０へ送信させる。最低限、代理プロセッサに、コンピュータ可読媒体４５０に記憶された入力ファイルに従って出力ストリームを生成させることが行われる。

【００４５】

ファイル処理システム６００のさらに別の代替的な例の実施の形態は、オペレーティングシステム４３０に出力オーガナイザ４９０を含む。この代替的な例の実施の形態による

と、さらに、コマンドパーサ４００は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、順序リスト４９１を出力オーガナイザ４９０へ提供させる。この代替的な実施の形態のファイル処理システム６００で実行されるアプリケーションインスタンス（４８０、４８５）は、出力オーガナイザ４９０に出力（４８２、４８７）を提供するように指示される。出力オーガナイザ４９０は、プロセッサ６０５によって実行されると、最低限、プロセッサ６０５に、アプリケーションの複数のインスタンティエーションから受け取った出力（４８２、４８７）をコマンドパーサ４００から受け取った順序リスト４９１に従って編成させる。さらに、出力オーガナイザ４９０は、最低限、プロセッサ６０５に、順序付けられた出力ストリーム５００を生成させる。

【００４６】

本方法および本装置をいくつかの代替的な例示の実施の形態の観点で説明してきたが、当業者には、本明細書を読み図面を検討することにより、それらの代替物、変更、並べ替え、および等価物が明らかになると考えられる。したがって、本明細書に添付の特許請求の範囲の真の精神および範囲は、このようなすべての代替物、変更、並べ替え、および等価物を含むことが意図されている。

【図面の簡単な説明】

【００４７】

【図１】コンピュータアプリケーションの性能を向上させる方法の一代表的な実施の形態を示すフロー図である。

【図２】２つまたは３つ以上の並列処理スレッドを作成する方法の一例の実施の形態を示すフロー図である。

【図３】並列処理スレッドの最大数を決定する方法の代替的な実施の形態を示すフロー図である。

【図４】並列処理スレッドの最大数を決定する方法の他の代替的な実施の形態を示すフロー図である。

【図５】２つまたは３つ以上の並列処理スレッドを作成する方法の一例示の代替的な実施の形態を示すフロー図である。

【図６】並列処理スレッドを使用して入力ファイルを処理する方法の一例の実施の形態を示すフロー図である。

【図７】コンピュータアプリケーションの性能を向上させる方法の一代替的な実施の形態を示すフロー図である。

【図８】アプリケーションが向上対象の候補であるかどうかを決定する方法の一例の実施の形態を示すフロー図である。

【図９】コンピュータアプリケーションの並列インスタンティエーションを起動する一例の方法を示すフロー図である。

【図１０】入力引数ファイルを処理するシステムの一例の実施の形態を示すブロック図である。

【図１１】ファイルを処理するシステムの一例示の実施の形態のオペレーションを示すデータフロー図である。

【図１２】最大スレッド決定機能の一実施の形態のオペレーションを説明するデータフロー図である。

【図１３】出力オーガナイザを含むアプリケーションの一代替的な例示の実施の形態を示すデータフロー図である。

【図１４】ファイル処理システムの一代替的な実施の形態のオペレーションを示すデータフロー図である。

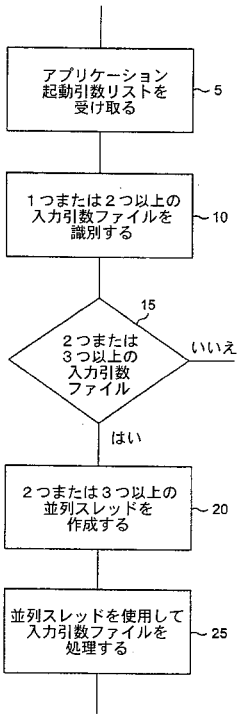
【符号の説明】

【００４８】

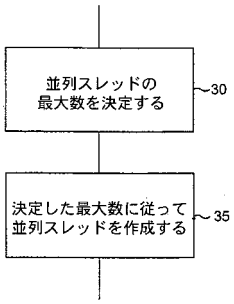
２１０　アプリケーション
２１５　引数パーサ
２３０　機能コア

２３３　タスクマネージャ（タスクマスタ）
４００　コマンドパーサ
４３５　タスクエグゼクティブ
４４０　ファイルマネージャ
６００　計算システム（ファイル処理システム）
６０１　コンソール
６０５　プロセッサ
６１５　メモリ

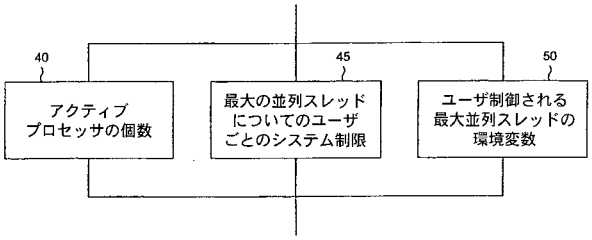
【図１】



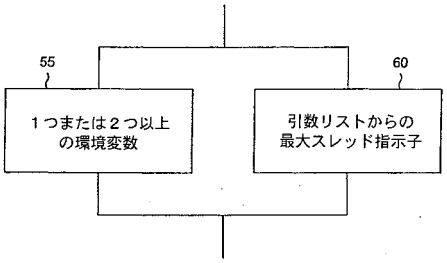
【図２】



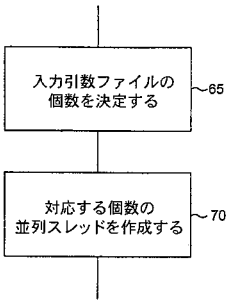
【図３】


40
アクティブ
プロセッサの個数
45
最大の並列スレッド
についてのユーザ
ごとのシステム制限
50
ユーザ制御される
最大並列スレッドの
環境変数

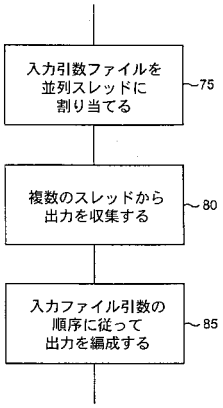
【図４】


55
１つまたは２つ以上
の環境変数
60
引数リストからの
最大スレッド指示子

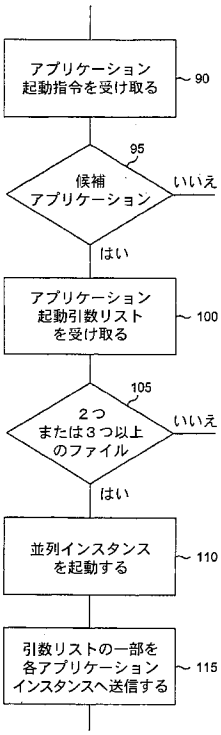
【図５】


入力引数ファイルの
個数を決定する
65
対応する個数の
並列スレッドを作成する
70

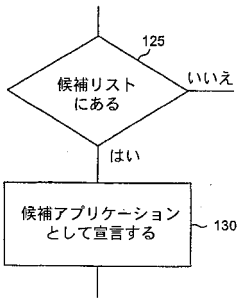
【図６】


入力引数ファイルを
並列スレッドに
割り当てる
75
複数のスレッドから
出力を収集する
80
入力ファイル引数の
順序に従って
出力を編成する
85

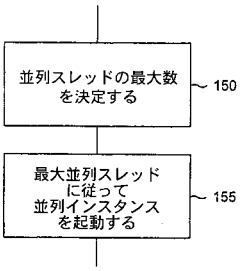
【図７】


アプリケーション
起動指令を受け取る
90
候補
アプリケーション
95
いいえ
はい
アプリケーション
起動引数リスト
を受け取る
100
2つ
または3つ以上
のファイル
105
いいえ
はい
並列インスタンス
を起動する
110
引数リストの一部を
各アプリケーション
インスタンスへ送信する
115

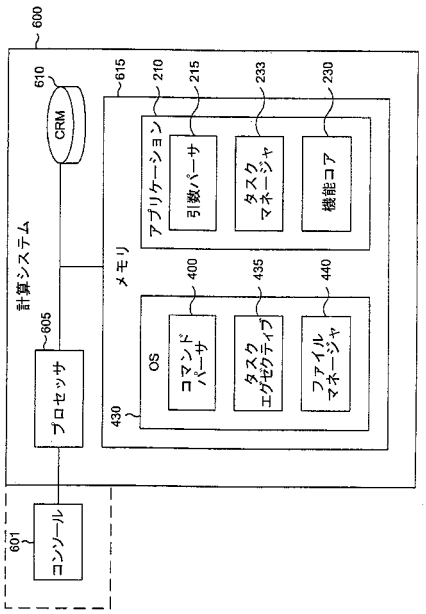
【図８】


候補リスト
にある
125
いいえ
はい
候補アプリケーション
として宣言する
130

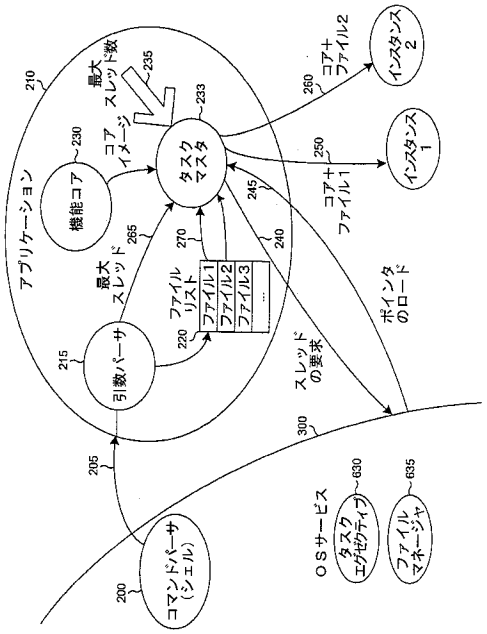
【図９】


並列スレッドの最大数
を決定する
150
最大並列スレッド
に従って
並列インスタンス
を起動する
155

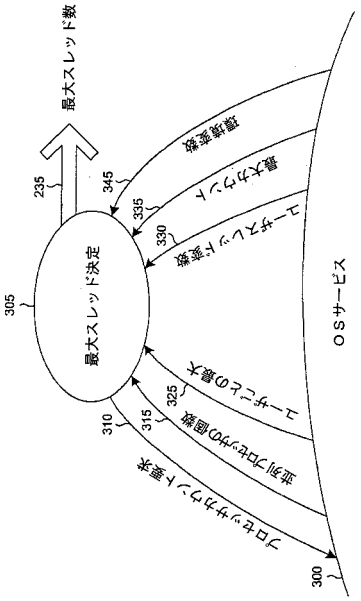
【図１０】


600
610
CRM
615
210
215
233
230
アプリケーション
引数パーサ
タスク
マネージャ
機能コア
メモリ
400
435
440
OS
コマンド
パーサ
タスク
エグゼクティブ
ファイル
マネージャ
430
計算システム
605
プロセッサ
601
コンソール

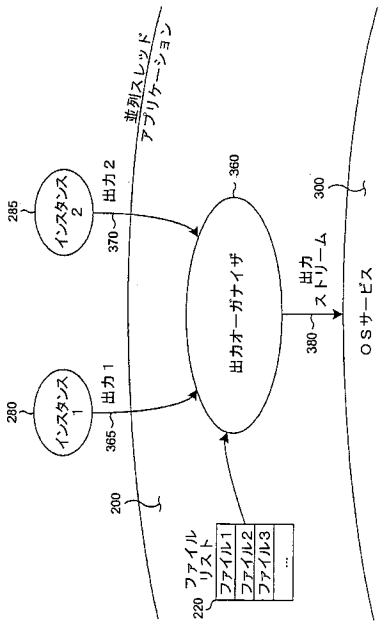
【図１１】



【図１２】



【図１３】



【図１４】