(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4150518号
(P4150518)
(45)発行日　平成20年9月17日(2008.9.17)　(24)登録日　平成20年7月4日(2008.7.4)

(51)Int.Cl.　　F I
G06F 12/00 (2006.01)　　G06F 12/00 591
G06F 9/46 (2006.01)　　G06F 9/46 410

請求項の数 11　(全 8 頁)

(21)出願番号　特願2001-533523(P2001-533523)
(86)(22)出願日　平成12年10月27日(2000.10.27)
(65)公表番号　特表2003-513356(P2003-513356A)
(43)公表日　平成15年4月8日(2003.4.8)
(86)国際出願番号　PCT/SE2000/002096
(87)国際公開番号　WO2001/031455
(87)国際公開日　平成13年5月3日(2001.5.3)
審査請求日　平成18年12月7日(2006.12.7)
(31)優先権主張番号　9903890-3
(32)優先日　平成11年10月28日(1999.10.28)
(33)優先権主張国　スウェーデン(SE)

(73)特許権者　500105160
ビーイーエイ　システムズ，　インコーポレイテッド
ＢＥＡ　Ｓｙｓｔｅｍｓ，　Ｉｎｃ．
アメリカ合衆国　カリフォルニア　９５１３１，　サン　ノゼ，　ノース　ファースト　ストリート　２３１５
２３１５　Ｎｏｒｔｈ　Ｆｉｒｓｔ　Ｓｔｒｅｅｔ，　Ｓａｎ　Ｊｏｓｅ，　ＣＡＬＩＦＯＲＮＩＡ　９５１３１　Ｕ．Ｓ．Ａ．
(74)代理人　100059959
弁理士　中村　稔
(74)代理人　100067013
弁理士　大塚　文昭

最終頁に続く

(54)【発明の名称】未使用のプログラムセクションの不要データを収集するための方法

(57)【特許請求の範囲】
【請求項１】
仮想マシンのメモリー空間内にもともと分布していた複数のプログラムセクションの中で、再生しようとする複数のプログラムセクションを決定し、この決定段階は現在必要とされているプログラムセクションを決定するスレッドを分析する段階を含んでおり、
再生されたプログラムセクションを作成し、そして再生しようとする複数のプログラムセクションにおける各プログラムセクションの参照をその再生されたプログラムセクションの参照と置換え、そして
再生されなかったプログラムセクションを消去する
ことを特徴とする仮想マシンのメモリー空間内でソフトウエアプログラムセクションの分布を適正化する方法。
【請求項２】
スレッドを分析する段階は、現在使用されているプログラムセクションをスレッドスタックからスレッドによって決定することを含んでいる請求項１に記載の方法。
【請求項３】
再生されたプログラムセクションを作成する段階は、スレッドを停止し、そして必要とされるプログラムセクションのコピーを別のアドレスに当該スレッドにより再作成することを含んでいる請求項１に記載の方法。
【請求項４】
消去する段階は、再生されなかったプログラムセクションが占めていたメモリー空間の

その部分を解放すること含んでいる請求項１に記載の方法。
【請求項５】
決定する段階は、プログラムセクションの選択で使用すべきロッキング機構を決定することを含んでいる請求項１に記載の方法。
【請求項６】
メモリー空間内に複数のプログラムセクションを記憶している仮想マシン、そして
不要データ収集プロセス
を備え、このコレクタは
再生しようとする複数のプログラムセクションを決定し、それは仮想マシンが現在必要としているプログラムセクションを決定することを含んでおり、
再生されたプログラムセクションを作成し、それは現在必要とされているプログラムセクションを別のアドレスに再作成すること、そして再生しようとする各プログラムセクションの参照をその再生されたプログラムセクションの参照と置換ることとを含んでおり、そして
再生されなかったプログラムセクションをすべて消去する
ことを特徴とする仮想マシンのメモリー空間内でソフトウエアプログラムセクションの分布を適正化するシステム。
【請求項７】
現在必要とされているプログラムセクションの決定は、現在使用されているプログラムセクションをスレッドスタックからスレッドによって決定することを含んでいる請求項６に記載のシステム。
【請求項８】
再生されたプログラムセクションの作成は、スレッドを停止し、そして必要とされるプログラムセクションのコピーを別のメモリーアドレスに当該スレッドにより再作成することを含んでいる請求項６に記載のシステム。
【請求項９】
消去は、再生されなかったプログラムセクションが占めていたメモリー空間のその部分を解放することを含んでいる請求項６に記載のシステム。
【請求項１０】
決定する段階は、プログラムセクションの選択で使用すべきロッキング機構を決定することを含んでいる請求項６に記載のシステム。
【請求項１１】
仮想マシンのメモリー空間内にもともと分布していた複数のプログラムセクションの中で、再生しようとする複数のプログラムセクションを決定し、この決定段階は現在必要とされているプログラムセクションを決定するスレッドを分析する段階を含んでおり、
再生されたプログラムセクションを作成し、そして再生しようとする各プログラムセクションの参照をその再生されたプログラムセクションの参照と置換え、そして
再生されなかったプログラムセクションを消去する
ようにコンピューターに実施させる指令を記憶しているコンピューターが読める媒体。
【発明の詳細な説明】
【０００１】
【発明の背景】
本発明は、データ処理アプリケーションの有効性を改善する方法に関する。
【０００２】
より詳細に言えば、本発明は、特にJava（登録商標）プログラム言語に関して、仮想マシンにおけるデータ処理速度の増大に関する。
【０００３】
以下では主にJavaに関して本発明を説明するが、本発明はJavaに限定されず、多くのプログラム言語を用いて適用することができる。
【０００４】

この方法は、プログラムの適応可能な最適化と共に使用することを意図したものである。適応可能な最適化においては、プログラムが実行されるときにプログラムが再構成され、またプログラムの異なる部分が最適化される。プログラムがより長く実行されるほど、より多くのメモリースペースが必要とされるから、データ処理能力を増大させることの一般的な問題は、新たなメモリー部位の迅速な形成にある。

【０００５】

Javaおよび他のダイナミックなプログラム言語は、自動的メモリー管理を含んでいる。このことは、プログラマーが、メモリーの使用部分のアカウントを維持する必要がないことを意味する。仮想マシンは時々所謂不要データ（不要情報、ガーベッジ）収集を実行し、主に、仮想マシンが全体のメモリーを走査して、どのオブジェクトがメモリーに保存されたか、およびどのプログラムが最早アドレスできなくなっているかを見出す。メモリーのこれらの部分は後で使用するために戻される。

【０００６】

また、Javaは所謂スレッド管理法を含む。従って、Javaは、二以上のプログラムの同時処理をサポートし、またはシミュレートするための方法を組込んでいる。このスレッドの管理は二つの部分に分割することができる。一方の部分は、異なるスレッドを制御された方法で構築する方法に関する。もう一つの部分は、何れのスレッドが実行されるべきか、および何れのスレッドがパッシブであり、実行されるために待機しているかに関する。

【０００７】

有効性を更に増大し、占有されているメモリー空間をプログラムが自由に使用できるようにするためには、オブジェクトに関してメモリーを単独で最適化することは充分ではない。

【０００８】

本発明はこの問題を解決するものである。

【０００９】

従って、本発明は、プログラムが、使用されるコンピュータのメモリーに保存される多くのプログラムセクションを含み、また不要データ収集が前記プログラムにより使用される場合に、仮想マシンを使用するときのデータ処理アプリケーションの有効性を改善する方法に関するものであり、ここで、本発明の方法は、全ての所謂スレッドスタックがそれによって必要とされるプログラムセクションに関して分析される第一のステップと；必要とされるプログラムセクションの夫々が再生され、その場合に、再生の前に、プログラムセクションに対する現在の参照（発生中の（行われている）参照）が再生されるプログラムセクションに対する参照で置きかえられる第二のステップと；全ての非再生プログラムセクションが消去され、この場合に対応するメモリー空間は前記プログラムが自由に使用できるようになる第三のステップによって特徴付けられる。

【００１０】

次に、添付の図面に示された本発明の実施例を一部参照して、本発明を詳細に説明する。

【００１１】

図１は、オペレーティングシステムがWinNt、LINUX、Solaris（登録商標）（ソラリス）または他の或る種ののシステムであるかどうかに関係なく、Java仮想マシンであるJVMを使用して、異なるデータプログラム1，2，3を実行できることを示している。上記で述べたように、Javaは非常にポピュラーなプログラム言語であるが、本発明はこの言語に限定されず、オブジェクト指向性で且つプラットホームに依存しない全ての対応するプログラム言語に適用することができる。

【００１２】

従って、本発明は、仮想マシンを使用するときにデータ処理アプリケーションの有効性を改善する方法に関し、ここでプログラムは、使用するコンピュータのメモリーに保存される膨大な数のプログラムセクションを含み、また不要データ収集プロセスが前記プログラムによって使用される。

【００１３】
オブジェクトの不要部分を整理し、それによって現在は最早使用されないオブジェクトを消去することにより、対応するメモリー能力を使用できるようにすることは以前から知られている。
【００１４】
大きなシステムでは、多くのプログラムセクションが一回または数回使用され、または短期間だけ適用された後、使用されないで残される。
【００１５】
Javaおよび対応するプログラムの場合、新たなプログラムセクションがロードされ、古いプログラムセクションは使用されずに残される。
【００１６】
更に、適用できる最適化は、メモリーの中に配置されたプログラムセクションの最適化および再最適化をもたらし、ここでは古いプログラムセクションが使用されないまま残される。
【００１７】
ロック機構（ロッキング機構）の選択および不要データ収集の選択を最適化するときは、古い機構を使用する全ての未使用プログラムセクションを新しい機構で置きかえる必要がある。
【００１８】
本発明によれば、全ての所謂スレッドスタックが、本発明の方法の第一のステップにおいて、必要とされるプログラムセクションに関して分析される。第二のステップにおいて、必要とされる夫々のプログラムセクションが再生され、ここではプログラムセクションに対する現在の参照が、前記再生の前に、再生されるプログラムセクションに対する参照と置きかえられる。第三のステップにおいては、全ての再生されないプログラムセクションが消去され、対応するメモリー空間をプログラムが使用できるようにする。
【００１９】
この方法は、使用されないプログラムセクションを一掃するだけでなく、最早使用されない古いプログラムセクションを介してのプロシーディングの代りに、直接的に、再生されたプログラムセクションに対する当該プログラムセクションの参照を指令するように、再生されたこれらプログラムセクションの間での再構成を生じさせる。
【００２０】
これは図２および図３に示されており、図２は古いプログラムセクションを示し、図３は使用された再生されたプログラムセクションを示している。三つのプログラムセクション、即ち、foo、apaおよびbarが図２に示されている。fooはメモリーアドレス4711でスタートする。apaはアドレス4714でスタートし、またbarはアドレス4720でスタートする。
【００２１】
このスレッドスタックの分析は、プログラムセクションfooおよびbarのみが使用され、結果的に、fooおよびbarはプログラムセクションapaに対して参照されなかったことを示している。
【００２２】
プログラムセクションfooおよびbarは、図３に示したプログラムセクションに再生される。この場合、プログラムセクションfooおよびbarは正確に再生されるが、当該プログラムセクションは新たなアドレスを得ること、またfooはこの新規なbarアドレス4903に対するbar点を参照することにおいて異なっている。
【００２３】
全ての古いプログラムセクション、即ち、図２のプログラムセクションは消去され、以前はこれらプログラムセクションによって占められていたメモリー空間は、更なる使用のために空席にされる。
【００２４】
オブジェクトの不要データ収集が生じるとき、不要データ収集が起きる間、通常はプロ

グラムの実行は停止される。不要データ収集の後、および使用されていないオブジェクトの消去の後に、プログラムの実行が再スタートする。

【００２５】

このような方法は、本発明を適用するときに使用することができる。

【００２６】

しかし、その代りに以下の方法を使用するのが極めて好ましい。

【００２７】

本発明の方法を実施するときに、一つのスレッドは当該プログラムが実行されている間停止され、それと共に、停止されたスレッドのために使用されるプログラムセクションはリストに移され、次いで該スレッドが再スタートされる。次いで、リストの中のプログラムセクションが再生および保存される。全てのスレッドはこの方法で、即ち、問題のスレッドに関する全ての使用されたプログラムセクションが再生されるよう処理された後に、同時に停止される。再生されなかった全てのプログラムセクションは消去され、全てのスレッドは再生されたプログラムセクションで再スタートされる。

【００２８】

再生は間欠的に生じるから、この方法では、プログラムの実行を停止する必要性がない。

【００２９】

先に述べたように、Javaおよび対応の言語ではロック機構が使用される。異なるロック機構を選択することもできる。重要なことは、もう一つのスレッドと同時に、一以上のスレッドが与えられたオブジェクトにアクセスするのを防止する上で最も効果的なロック機構を選択することである。

【００３０】

幾つかのスレッドが一つの同じオブジェクトまたはソースにアクセスすることが望まれるときには、同期の問題が存在する。Javaにおいてこの問題を解決するために、夫々のスレッドは、ソースロックに到達しようと努力する。このソースロック機構は種々の方法で使用することができる。異なるロック機構の効果は、同期したソースに対するアクセスを得るように、スレッドが如何にして努力するかに依存するであろう。

【００３１】

好ましい実施例によれば、ロック機構が使用されるときには、最も有効なロック機構が前記第一のステップの前のステップで同定され、この同定されたロック機構を使用するプログラムセクションが再生される。

【００３２】

不要データ収集アルゴリズムに関しては、これらもまた選択される必要がある。多くのオブジェクト指向性言語は、不要データ収集を使用する。このことは、プログラマーが当該システムに対して、一定のオブジェクトが最早必要でないことを明瞭に指示する必要がないことを意味する。該システムは、この検出を実施することができ、前記オブジェクトによって占められていたメモリーの一部を再利用する。この検出および再利用の効果的な実施のために、多くの異なるアルゴリズムが提案されている。異なるアプリケーションについては異なるアルゴリズムが最適であることが分かった。実行されているプログラムアプリケーションのための、最良の不要データ収集アルゴリズムの選択は、問題のプログラムに関して最大の実行速度を達成するために極めて重要である。

【００３３】

本発明のもう一つの好ましい実施例によれば、異なる不要データ収集アルゴリズムが使用されるときに、前記第一の方法ステップの前のステップにおいて、種々のオブジェクトのアロケーションおよび寿命が決定され、その後に最も効果的な不要データ収集アルゴリズムが同定され、必要な不要データ収集アルゴリズムを構成するプログラムセクションが再生され、次いで残りの不要データ収集アルゴリズムが消去される。

【００３４】

この好ましい実施例の適用は、コード、スレッドおよびメモリー管理を最適化するため

の非常に効果的な方法を提供し、ここでの一般的特徴は、システムに未使用のプログラムセクションをロードしないように同定および再生することにある。

【図面の簡単な説明】

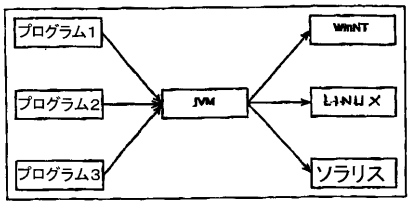
【図１】　図１は、ブロック図である。

【図２】　図２は、古いプログラムセクションを図示している。

【図３】　図３は、本発明によって再生されたプログラムセクションを図示している。

【図１】

Fig 1



【図２】

Fig 2

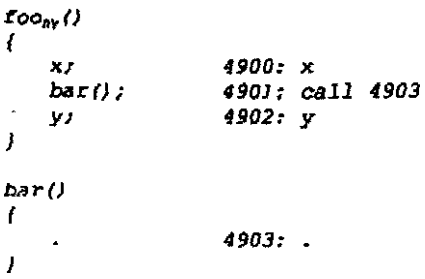
```
foo()
{
   x;              4711: x
   bar();          4712: call 4720
   y;              4713: y
}

apa()
{
                   4714: .
                   4715: .
                   4716: .
                   4717: .
                   4718: .
}                  4719: .

bar()
{
   .
}                  4720: .
```

【図３】

Fig 3



フロントページの続き

(74)代理人 100082005
弁理士 熊倉 禎男
(74)代理人 100065189
弁理士 宍戸 嘉一
(74)代理人 100084009
弁理士 小川 信夫
(74)代理人 100086771
弁理士 西島 孝喜
(74)代理人 100084663
弁理士 箱田 篤
(72)発明者 ダールステット ホアキム
スウェーデン エス－１１８ ４２ ストックホルム フリントバッケン １０

審査官 田中 秀人

(56)参考文献 国際公開第９９／０４５４８１（WO，Ａ１）
国際公開第９９／０１８７３０（WO，Ａ１）
特開昭６３－０６２０５３（ＪＰ，Ａ）
国際公開第９９／０３２９７８（WO，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 12/00 - 12/06
G06F 9/46