(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5980916号
(P5980916)
(45)発行日　平成28年8月31日(2016.8.31)　(24)登録日　平成28年8月5日(2016.8.5)

(51)Int.Cl.　　F I
G06F 11/36 (2006.01)　　G06F 11/36 104
G06F 12/02 (2006.01)　　G06F 12/02 540
G06F 9/50 (2006.01)　　G06F 9/46 462A

請求項の数 17　(全 27 頁)

(21)出願番号　特願2014-517071 (P2014-517071)
(86)(22)出願日　平成24年6月18日(2012.6.18)
(65)公表番号　特表2014-523022 (P2014-523022A)
(43)公表日　平成26年9月8日(2014.9.8)
(86)国際出願番号　PCT/US2012/043033
(87)国際公開番号　WO2012/177577
(87)国際公開日　平成24年12月27日(2012.12.27)
審査請求日　平成27年5月25日(2015.5.25)
(31)優先権主張番号　13/163,745
(32)優先日　平成23年6月20日(2011.6.20)
(33)優先権主張国　米国(US)

(73)特許権者　314015767
マイクロソフト　テクノロジー　ライセンシング，エルエルシー
アメリカ合衆国　ワシントン州　９８０５２　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100107766
弁理士　伊東　忠重
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介

最終頁に続く

(54)【発明の名称】コンピュータにより実行される方法及びコンピュータシステム

(57)【特許請求の範囲】
【請求項１】
メモリ割当情報を提供するように明確には設計されていないアプリケーションを分析し、メモリ割当情報を取得するために、コンピュータの少なくとも１つのプロセッサにより実行される方法であって、
アプリケーションを検出するステップであって、該アプリケーションにより行われるメモリ割当を示す情報は利用可能でない、ステップと、
該アプリケーションにより行われるメモリ割当を判別するために該アプリケーションについて分析を実行するステップと、
該アプリケーションをフックし、該アプリケーションがメモリを割り当てる該アプリケーション内のロケーションにおける割当情報を提供するステップと、
メモリを割り当てるための該アプリケーションからのリクエストとフックされたアプリケーションコードにより提供される関連割当情報とを受信するステップと
を有する方法。
【請求項２】
前記アプリケーションを検出するステップが、前記アプリケーションが改善されたメモリモデルに加わっているか否かを指定する情報を前記アプリケーションのバイナリモジュールにおいて検出することを含む、請求項１に記載の方法。
【請求項３】
前記アプリケーションを検出するステップが、以前にキャッシュしたアプリケーション

割当情報が利用可能であるか否かを判断することを含む、請求項１に記載の方法。
【請求項４】
前記分析を実行するステップが、アプリケーションバイナリモジュールにおいてメモリ関連動作を特定する静的バイナリ分析を前記アプリケーションについて行うことを含む、請求項１に記載の方法。
【請求項５】
前記分析を実行するステップが、前記アプリケーションの実行時においてメモリ関連動作を特定する動的分析を前記アプリケーションについて行うことを含む、請求項１に記載の方法。
【請求項６】
前記分析を実行するステップが、メモリを割り当てること、メモリにアクセスすること又はメモリを解放することのための１つ以上の特定のホストアプリケーションプログラミングインタフェース（ＡＰＩ）の呼び出しを特定することを含む、請求項１に記載の方法。
【請求項７】
前記分析を実行するステップが、前記アプリケーションコードのメモリ関連エリア付近で割当メタデータを特定することを含む、請求項１に記載の方法。
【請求項８】
前記アプリケーションをフックすることが、前記アプリケーションコードの或るポイントにおいて、アプリケーションの通常のメモリ関連動作をインターセプトする又は強化するためにアプリケーションバイナリコードのリダイレクションを行うことを含む、請求項１に記載の方法。
【請求項９】
前記アプリケーションをフックすることが、前記アプリケーションにより本来呼び出されるメタデータなしに、標準的な割当機能によらず、割当メタデータを提供する割当機能を起動するアプリケーションフックを提供することを含む、請求項１に記載の方法。
【請求項１０】
前記受信するステップが、アプリケーション情報は該アプリケーション情報を提供するように明確には設計されていないアプリケーションに由来していることを確認することが可能なホストにより前記リクエストを受信することを含む、請求項１に記載の方法。
【請求項１１】
割当情報を提供するように設計されていないソフトウェアアプリケーション内でのメモリの割当及び利用について、より多くの制御をアプリケーションホストに提供する、以下の要素に組み込まれるソフトウェア命令を実行するように形成されたプロセッサ及びメモリを有するコンピュータシステムであって、
メモリ割当の管理の仕方を交渉するためにアプリケーション及びホストの間に通信インタフェースを提供するアプリケーションインタフェース要素と、
前記アプリケーションがメモリをどのように使用するかを判断するためにアプリケーションバイナリコード又は他のアプリケーションコードを静的に分析する静的分析要素と、
前記アプリケーションによるメモリの利用の仕方に関連する追加的な情報を収集するために、動作しているアプリケーションを動的に分析する動的分析要素と、
前記アプリケーションが動作している環境を含み、当該コンピュータシステムにより提供されるメモリマネジャにアクセスするホスト要素と
を有し、前記静的分析要素及び動的分析要素はアプリケーションメモリ利用情報を前記ホストに提供する、コンピュータシステム。
【請求項１２】
前記アプリケーションインタフェース要素が、メモリ割当を要求しかつ該メモリ割当に関する情報を指定するために前記アプリケーションにより提供される１つ以上の機能又はベースクラスに加えて、前記アプリケーションにより提供されるユーザ定義機能を利用して前記アプリケーションのメモリとやり取りを行うために前記ホストにより使用される機

能又はベースクラスを有する、請求項１１に記載のコンピュータシステム。
【請求項１３】
前記アプリケーションインタフェース要素が、静的及び／又は動的分析により判定された使用されるアプリケーションコード間でやり取りを行い、割当情報を提供するために前記ホストとやり取りを行う、請求項１１に記載のコンピュータシステム。
【請求項１４】
前記静的分析要素が、アプリケーションのバイナリコード、中間コード又は他のコンパイルされた若しくは実行可能なバージョンを分析する、請求項１１に記載のコンピュータシステム。
【請求項１５】
前記静的分析要素が、前記アプリケーションのバイナリコードを利用して、情報を受信する、或いは前記アプリケーションの特定のメモリ関連処理をインターセプトする、請求項１１に記載のコンピュータシステム。
【請求項１６】
請求項１～１０のうち何れか１項に記載の方法を前記少なくとも１つのプロセッサに実行させるコンピュータプログラム。
【請求項１７】
請求項１６に記載のコンピュータプログラムを記憶するコンピュータ読み取り可能な記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
開示される実施の形態はコンピュータにより実行される方法及びコンピュータシステム等に関連する。
【背景技術】
【０００２】
コンピュータシステムにおけるメモリ管理又はメモリマネジメントは、複数のアプリケーションと1つのオペレーティングシステムとがメモリを利用することについて合意する方式又は形式に言及している。各々のコンピュータシステムは一定量の物理的なランダムアクセスメモリ(RAM)その他のメモリを有するが、オペレーティングシステムは、物理メモリとは異なるメモリサイズを表現する仮想メモリを、アプリケーションやオペレーティングシステム要素に提示する。場合によっては、仮想メモリは、メモリの特定の部分にアプリケーション各々がアクセスするのをオペレーティングシステムが制限することを許容したり、或るアプリケーションが偶発的又は意図的に別のアプリケーションのメモリを修正することで該別のアプリケーションの動作を妨げてしまうことをオペレーティングシステムが禁止することを許容する。一般にオペレーティングシステムはアプリケーションやオペレーティングシステム要素の要求に応じてメモリを割り当てたり解放したりする1つ以上の機能を提供する。オペレーティングシステムはアプリケーションにメモリプール(memory pool)を提供し、そのメモリプールの中でアプリケーションはメモリのチャンク(chunk)を割り当てることができる。アプリケーション又は一群のアプリケーションが、インストールされている量の物理メモリより多い仮想メモリを利用する場合、オペレーティングシステムは、ページング又はディスクスワッピング(paging or disk swapping)と呼ばれる手順(例えば、メモリのページをディスクに保存して取り出す手順)におけるスワップファイル(swap file)により、低頻度にディスクストレージを利用してメモリの見かけ上のサイズを拡張する。
【０００３】
提供される機能の割当及び解放とは別に、オペレーティングシステムはアプリケーションがどのようにメモリを利用しているかをほとんど把握できない。多くのコンピュータ装置はメモリに関する特定の制約を有する。例えば、モバイルコンピュータ装置はデスクトップコンピュータで通常利用可能なメモリよりもかなり少ないメモリしか含んでおらず(

或いは、システムはエネルギ消費を減らすためにいくらかのメモリへの給電を抑制することを希望するかもしれない)、同時にいくつのアプリケーションを実行させることができるか、アプリケーション各々がどの程度多くのメモリを要求/消費できるか等について制限が生じる。特定のコンピュータシステムの中でアプリケーションコードを使用する他のコンピュータ環境は、その環境におけるメモリの利用について制限或いは上限を課すかもしれない。VMウェア(VMware)及びマイクロソフトTM仮想PC(MICROSOFT TM Virtual PC)、ハイパーバイザオペレーティングシステム等のようなホスト等に限られたリソースが指定されるかもしれない。これら全ての状況において、効率的なメモリ管理が強く望まれている。

【先行技術文献】

【特許文献】

【０００４】

【特許文献１】米国特許出願公開第2010-0162233号明細書

【特許文献２】米国特許出願公開第2008-0063354号明細書

【特許文献３】米国特許出願公開第2002-0083112号明細書

【特許文献４】米国特許出願公開第2003-0061402号明細書

【発明の概要】

【発明が解決しようとする課題】

【０００５】

新たなコンピュータプラットフォームは、新たな技術を導入し或いは既存の技術を設計し直し、アプリケーション同士の間で共有される限られたメモリに関する問題に対処する。例えば、移動電話オペレーティングシステムはそのようなアプリケーションのメモリスナップショットを形成し、アプリケーションが最前面(foreground)(例えば、アクティブに使用されている画面)になかった場合にオペレーティングシステムがそれをシャットダウンしてアプリケーションメモリの画像を低速メモリ(例えば、フラッシュメモリ又はその他のメモリ)に保存するようにする。アプリケーションが選択されると、オペレーティングシステムは保存した画像をロードし直して(リロードして)アプリケーションを開始する。アプリケーションはシャットダウンされていたことに気付くことさえないかもしれない。このような技術は有用ではあるが、オペレーティングシステムは、メモリを利用するアプリケーションの曖昧な又は予測困難な(opaque)要求に依然として委ねられる。動的なメモリ利用に関して現在なされている多くの判断は、実行時間中に見出された情報に基づいてなされる。そのような情報の具体例は割り当てられるメモリセグメントに対するリファレンスのサイズ及び数を含む。この情報は、何れの割り当てがディスクに通知(ページング)されるか、何れの割り当てがハイパフォーマンスメモリにキャッシュされるか、或いは何れの割り当てが何らかの自動メモリ管理システムにより解放されるか等を決定するために利用できる。不都合なことに、従来のアプリケーションでは長い間何れのプラットフォームも制限されており、メモリ管理と同様に広範な領域で新たなモデルを採用することは困難である。

【０００６】

開示される実施の形態の課題は、効率的なメモリ管理等を可能にすることである。

【課題を解決するための手段】

【０００７】

実施の形態により提供される方法は、

メモリ割当情報を提供するように明確には設計されていないアプリケーションを分析し、メモリ割当情報を取得するためにコンピュータの少なくとも１つのプロセッサが実行する方法であって、

アプリケーションを検出するステップであって、該アプリケーションにより行われるメモリ割当を示す情報は利用可能でない、ステップと、

該アプリケーションにより行われるメモリ割当を判別するために該アプリケーションについて分析を実行するステップと、

該アプリケーションを利用して、該アプリケーションがメモリを割り当てる該アプリケーション内のロケーションに割当情報を提供するステップと、
利用されたアプリケーションコードにより提供されるメモリ及び関連割当情報を割り当てるためのリクエストを該アプリケーションから受信するステップと
を有する方法である。
【図面の簡単な説明】
【０００８】
【図１】一実施形態によるメモリ管理システムの要素を示すブロック図。
【図２】一実施形態によるメモリ管理システムの動作環境を示すブロック図。
【図３】メモリの割当及び利用を要求するためにソフトウェアアプリケーションにおいて行われるメモリ管理システムの処理を示す一実施形態によるフローチャート。
【図４】メモリの割当及び利用を要求するためのアプリケーション要求を受信するホストにおいて行われるメモリ管理システムの処理を示す一実施形態によるフローチャート。
【図５】メモリ割当情報を提供するように明確には設計されていないアプリケーションを分析するメモリ管理システムの処理を示す一実施形態によるフローチャート。
【図６】アプリケーションを静的に分析し及び改善されたメモリ情報のマニフェストを提供するメモリ管理システムの処理を示す一実施形態によるフローチャート。
【図７】検出されたメモリの逼迫に応じてメモリに関して処理を行うメモリ管理システムの処理を示す一実施形態によるフローチャート。
【図８】メモリがホストによって事前に修正されているアプリケーションをアクティブ化するメモリ管理システムの処理を示す一実施形態によるフローチャート。
【発明を実施するための形態】
【０００９】
＜実施の形態の概要＞
本願において説明されるメモリ管理システムは、メモリがどのように使用されているかを記述する情報をアプリケーションから受信し、メモリを利用するためのアプリケーション要求をアプリケーションホストがよりいっそう制御できるようにする。現在のアプリケーションホストは、アプリケーションがどの程度多くのメモリ要求を行っているか、及びどの程度のメモリサイズが個々の要求で要求されているかということ以外に、アプリケーションによるメモリの利用についてほとんど何も知らない。すなわち、メモリアプリケーション各々の目的、何れのアプリケーションが間もなく使用されることになるか、アプリケーションホストが更に多くのメモリを必要とする場合に何れのメモリアプリケーションが容易に作成し直せるか、何れのメモリアプリケーションがしばらくの間使用できなくなるか、何れのアプリケーションメモリがアプリケーションのパフォーマンスに影響を与えることなくディスクにページングできるか等をアプリケーションホストは把握していない。不都合なことに、アプリケーションホストはこの種の判断を行うタスク又は任務を担っているが、この種の判断を効率的に行うことに関するほとんどの情報をアプリケーションが保持している。
【００１０】
本メモリ管理システムはこのような問題を様々な方法で克服する。第1に、本システムはアプリケーションメモリ管理アプリケーションプログラミングインタフェース(API)を提供し、メモリ管理APIは、後にメモリを管理するのに役立つメモリ割当に関する豊富な情報をアプリケーションが特定できるようにする。APIは、メモリが何時必要とされるかをアプリケーションに通知し、アプリケーションによるやり取りを行うことなく必要に応じてメモリの割り当てを能動的に解放及び再構築する機能をアプリケーションホストにもたらす。第2に、本システムが提供する能力(又は発揮する機能)は、従来のアプリケーション(レガシーアプリケーション)を静的に及び/又は動的に分析し、本システムと共に動作するように修正されていないアプリケーションに、より効率的なメモリ管理に参加できる何らかの能力(機能)を与える。第3に、本システムは、アプリケーションにより提供される情報を活用するための変更であってその情報を利用していっそう効率的にメモリを管

理するための変更を、アプリケーションホストにもたらし、かつメモリのアプリケーションによる利用を把握できるようにする。かくて本メモリ管理システムはアプリケーションホストの動作を改善しかつアプリケーションがコンピュータリソースをいっそう効率的に利用できるようにする新たなメモリ管理モデルを提供する。

【００１１】

この概要の欄は以下の詳細な説明で更に説明される形態についての簡易な概念を選択的に導入しているにすぎない。この概要の欄は特許請求の範囲に記載された重要な事項或いは本質的特徴を特定するようには意図されておらず、特許請求の範囲を限定するためにしようされることも意図されていない。

【００１２】

＜実施の形態の詳細な説明＞

本願において説明されるメモリ管理システムは、メモリがどのように使用されているかを示す又は記述する情報をアプリケーションから受信し、メモリを利用するためのアプリケーション要求をアプリケーションホストがよりいっそう適切に制御できるようにする。現在のアプリケーションホストは、アプリケーションがどの程度多くのメモリ要求を行っているか、及びどの程度のメモリサイズが個々の要求で要求されているかということ以外に、アプリケーションによるメモリの利用についてほとんど何も知らない。すなわち、メモリアプリケーション各々の目的、何れのアプリケーションが間もなく使用されることになるか、アプリケーションホストが更に多くのメモリを必要とする場合に何れのメモリアプリケーションが容易に作成し直せるか、何れのメモリアプリケーションがしばらくの間使用できなくなるか、何れのアプリケーションメモリがアプリケーションのパフォーマンスに影響を与えることなくディスクにページングできるか等をアプリケーションホストは把握していない。不都合なことに、アプリケーションホストはこの種の判断を行うタスク又は任務を担っているが、この種の判断を効率的に行うことに関するほとんどの情報をアプリケーションが保持している。この矛盾又は問題は、現在のところ、基本レベルの機能を提供しかつ何れの処理がなされるかを推測するアプリケーションホストにより解決されている。多くの場合、アプリケーションホストはアプリケーションホストがメモリを必要とする直前にディスクに対するメモリを呼び出す、或いはアプリケーションホストはアプリケーションにとって重要でない多くのメモリ管理労力を費やしてしまうかもしれない。

【００１３】

本メモリ管理システムはこのような問題を様々な方法で克服する。第1に、本システムはアプリケーションメモリ管理アプリケーションプログラミングインタフェース(API)を提供し、メモリ管理APIは、後にメモリを管理するのに役立つメモリ割当に関する豊富な情報をアプリケーションが特定できるようにする。APIは、メモリが何時必要とされるかをアプリケーションに通知し、アプリケーションによるやり取りを行うことなく必要に応じてメモリの割り当て(又は割り振り又は指定)を積極的に解放及び再構築する機能をアプリケーションホストにもたらす。第2に、本システムが提供する能力(又は発揮する機能)は、従来のアプリケーション(レガシーアプリケーション)を静的に及び/又は動的に分析し、本システムと共に動作するように修正されていないアプリケーションに、より効率的なメモリ管理に参加できる何らかの能力(機能)を与える。第3に、本システムは、アプリケーションにより提供される情報を活用するための変更であってその情報を利用していっそう効率的にメモリを管理するための変更を、アプリケーションホストにもたらし、かつメモリのアプリケーションによる利用を把握できるようにする。かくて本メモリ管理システムはアプリケーションホストの動作を改善しかつアプリケーションがコンピュータリソースをいっそう効率的に利用できるようにする新たなメモリ管理モデルを提供する。アプリケーションホストは、本願で使用されているように、アプリケーションを実行するオペレーティングシステム又は他のタイプのホストに関し(例えば、オペレーティングシステム又は仮想サブシステムで自ら動作するアプリケーション)、例えば、シルバーライトTM(SILVERLIGHT TM)により提供されているランタイム(runtimes)、ドットネット(.NET)、ネイティブウィン32ホスト(native Win32 host)、VMウェア及び仮想PCにより提供される

その他のホスト又は仮想マシン等である。上記に言及された3つの形態の各々が以下において詳細に説明される。

【００１４】

＜＜修正アプリケーション＞＞

多くの場合、ソフトウェア開発者にとって、メモリ管理システムとやり取りを行うようにソフトウェアを修正することが可能である。実際に開発されるアプリケーションに関し、ソフトウェア開発者はメモリ管理システムがもたらす利点の観点からメモリ管理システムを使用することを選択するかもしれないし、或いはアプリケーションがメモリ管理システムを使用するように動作する特定のプラットフォームにより決められるかもしれない。多くの場合、アプリケーションは、そのアプリケーションが使用しがちな仕方で割り振られたメモリを維持する。例えば、アプリケーションのユーザインタフェースの或る部分から別の部分へユーザが遷移した場合に、アプリケーションはユーザがそのインタフェースを再び訪れる場合に備えてかつてのインタフェースからの情報を保持するかもしれない。今日、この情報は他のアクティブに使用されるメモリと同程度にホストにとってもまさに必要であるように見えている。本メモリ管理システムはホストにこの種の状況を通知する方法をアプリケーションに提供し、そのようなメモリが優先順位を落とせるようにする。これに応じて、ホストは、そのようなメモリをページングする有望な候補として選択し、或いはアプリケーションからの追加的な情報に起因していっそう効果的な他のメモリ管理判断を可能にする。

【００１５】

一実施形態において、メモリ管理システムはアプリケーションプログラミングモデル又はフレームワーク又は手段を提供し、これはメモリマネジャが実行時間の最中にメモリの利用方法を最適化することに関する優れた判断を可能にする。これはアプリケーションプログラミングモデル/フレームワークを利用することで達成され、アプリケーションプログラミングモデル/フレームワークは、割当に使用される処理及びメタデータの双方を受信し、かつ任意の所与のメモリオブジェクトについて指定されたメモリ割当でアプリケーション要求を満たす。メタデータは、アプリケーションがホストと通信することを希望している情報であってメモリ割当の種類又は目的を記述している情報を提供し、例えば、メモリの優先度、割り当てられるメモリの量、抹消状態からメモリの内容を再構築することの容易性(例えば、内容又はコンテンツはファイルからロードされてもよいし、或いはアルゴリズムにより計算されてもよい)、アクセスの頻度、アプリケーションが後どのくらいメモリを使用してよいか等を示す。メモリを割り当てて満たすのに使用される処理は、アプリケーションの要求の際にメモリを解放しそして解放されたメモリを再構築できる程度に充分な情報をホストに提供する。メモリを満たすのに使用されるメタデータ及び処理を開発者が特定できるようにすることで、メモリ管理ステムはアプリケーションにより指定された所望の用途に合わせてメモリの利用の仕方を最適化できる。

【００１６】

メモリ管理システムによりアプリケーションに提供されるAPIは、アプリケーションフレームワーク又はプログラミングモデルにより、メモリの割り当ての利用の仕方を記述するメタデータを特定又は指定する手段をアプリケーション開発者に提供する。更に、APIは、開発者が標準的な手段を用いて従来の機能によりメモリを満たす又は修正することを、アプリケーションフレームワークが指定できるようにする。これは、メモリ管理システムが、パフォーマンスの要因に応じてメモリを満たすことや、メモリの利用度が低い期間にメモリを解放することや(すなわち、メモリを解放するための機会コスト(opportunity cost)が、メモリを再配分又は再構築するのを遅らせるコストより大きかった場合)、その他の処理を可能にする。

【００１７】

メモリの利用を最適化することは、従来技術の多くを包含するが、一般に(a)パフォーマンス又は(b)サイズの最適化を意味することになるであろう。(a)パフォーマンスの場合、最適化は、メモリの割当を遂行する手段が利用可能であるならば、未だ必要とされてい

ないメモリの割当を実行できるようにすることを意味してもよい。これは、現在のCPUの利用度が低くかつアプリケーションがアイドルである場合に望ましいかもしれない。場合によっては、メモリの割当を求めるアプリケーションの要求(リクエスト)は、将来の時点で何かを割り当てることなく参照するためにホストが保存する留意事項(note)となるかもしれない。後に、アプリケーションがAPIを通じてメモリの利用を要求する場合、或いはホストが要求を満たす適切な時点を決定した場合に、メモリ管理システムは要求されたメモリを実際に割り当てる。(b)サイズの場合、最適化は、メモリのフットプリント(footprint)を減らすこと、或いは現在割り当てられているメモリ及び将来的なメモリの必要性に基づいて判断を行うことを意味してもよい。

【００１８】

アプリケーションとホストとの間の実際のインタフェースは、当業者に認められる様々な形態をとってよい。例えば、アプリケーションは、各々のタイプの割当について割当機能を発揮してもよく、割当要求の際に割当に対するポインタ又はリファレンスをホストに渡してもよい。ホストが割当を実行する準備を整えた場合、ホストは提供又は発揮される割当機能を呼び出し、アプリケーションが適切なメモリ割当機能を用いてメモリを作成する。同様に、アプリケーションはリファレンスを他の機能に渡し、メモリを解放すること、メモリを移動させること、メモリ内容を別のタイプのストレージに切り替えること等をホストが要求できるようにしてもよい。同様な概念を割当(割り振り又は指定)に利用することも可能であり、アプリケーションがオペレーティングシステムのメモリを要求する場合、オペレーティングシステムは様々な要因に応じて割当を遅延させてもよい。オペレーティングシステムが割当の準備を整えた場合、機能リファレンス(function reference)が(割り当てられたメモリと共に)呼び出される、或いはイベントが生じる(或いは類似する処理が行われる)。メモリサイズ(要求されたサイズとは異なっているかもしれない)、優先度又は優先順位、キャッシュする好み、ページ処理機能(pageability)、どの程度多くのメモリが満たされているか、メモリに対する依存性又はリファレンス、メモリが更新されているか否か等のようなメタデータを、インタフェースは受信してもよい。一実施形態では、システムはメモリインタフェースクラスを提供し、そのメモリインタフェースクラスからアプリケーションはメモリ割当の各々のタイプを決定又は導出する。クラスは、アプリケーション固有のアプリケーション機能をもたらすゲットポインタ関数(GetPointer function)、或いはその他のメモリ処理タスクを実行する機能をもたらすゲットエックス関数(GetX function)を含んでもよい。代替的又は追加的に、アプリケーションは従来の方法で割当を行いその後に登録ポインタ関数(RegisterPointer function)を呼び出し、登録ポインタ関数は、割り当てられたメモリをホストに登録し、かつ割り当てられたメモリに関連付けるために本願で説明される追加的な情報を指定する。以下の擬似コードはアプリケーションが使用できる或るメモリクラスの一例を示す。

【００１９】

【数１】

```
CMemChunk // base class for all memory allocations
{
  <Global List of Allocations>
  …
};
CMyMemory : public CMemChunk
{
  Allocate ( size );
  Fill () { <how to fill memory> } // override
  Attributes <e.g., priority, pageability, etc.>
};
```

代替的又は追加的に、ソース注釈言語(SAL)又はその他のマークアップを利用して、存在するメモリ割当を特定し、割当、アクセス又はその他のメモリのやり取りの各々に関するメタデータ及び追加的なパラメータを指定することで、開発者は本願で説明されているフレームワーク又は手段をアプリケーションコードに導入してもよい。

【００２０】

一実施形態において、メモリ管理システムは、1つのアプリケーションの中で動作し、カーネル又はその他のホストと共有されていなくてもよい。アプリケーションは改善されたメモリ管理による恩恵を享受でき、それ自身の内部のメモリマネジャは、本願で説明されるフレームワーク又は手段を利用することで提供される追加的な情報と共に動作することが可能である。場合によっては、ホストはアプリケーションが呼び出せる登録機能(registration function)を提供し、アプリケーション相互の恩恵を享受し、適切に決定されたメモリの割当及び利用法をホストが利用できるようにする。具体例として、システムはマルウェア(malware)をスキャンしてしまう前にアプリケーションに通知を行い、アプリケーションが関連するメモリを全くロードしないようにし、スキャンを高速化できるようにしてもよい。別の例として、アプリケーションはCPUがアイドル状態に移る前に先手を取ってメモリを割り当て、ユーザが何かを行うためにCPUが実行を再開するイベントにアプリケーションが迅速に応答できるようにしてもよい(電力状態変化に応答する機能を強化する)。

【００２１】

＜＜非修正アプリケーション＞＞

メモリ管理システムとやり取りを行うようにソフトウェアアプリケーションを修正することがソフトウェア開発者にとって不可能であった場合、或いはシステムが未修正のアプリケーション(すなわち、本システムと共に動作するように明確には設計されていないアプリケーション)と共に実現される場合であっても、システムはメモリ管理の恩恵を享受することが可能である。これを行うため、システムは、アプリケーションがどのようにしてメモリを使用するかを示す情報を収集する(例えば、静的な分析及び/又はアプリケーションの評価に基づいてもよい。分析又は評価は、アプリケーションを動作させること、メモリの割当の情報を取得すること、アプリケーションの実行時間の間に利用度を監視すること等を含む)。この情報は、パフォーマンス特性を判定する際に有用であり、アプリケーションホストのメモリマネジャにより有益に(intelligently)使用される。この情報の利用の仕方の具体例は、ゴミ又は不要データの高度な収集、ディスクに対する高度なページング、パフォーマンスの高いメモリキャッシュの高度なキャッシュ管理等を含み、アプリケーションが遭遇するかもしれない潜在的なメモリの制約をユーザに警告することさえ

も含んでもよい。

【００２２】

バイナリ又は二進状態による静的な分析(又は解析)、二進状態による動作時間の分析、及び二進状態を利用することで、二進状態によるメモリの割当及びメモリ割当の利用度に関する任意の追加的な情報を収集することが可能である。この情報は、ロードすること/ロードしないことに関する更に詳細な動作、及び物理メモリ内の割り当て場所を導出するのに利用されてもよい。メモリ管理システムは、メモリの割当に関する潜在的な又は実際の利用の仕方を記述するメタデータを、アプリケーションメモリの割り当て方法(の情報)に自動的に注釈として付する手段を提供する。この分析は、存在するアプリケーションとやり取りを行う又は作成し直すことを如何なる開発者にも要求することなく、二進状態で静的に又は実行時間中に動的に、自動的に実行できる。実行されると、分析(結果)はシステムによりキャッシュされ、将来アプリケーションを如何にして処理するかをホストオペレーティングシステムが分かるようにする。分析(結果)は、単にローカル又は局所的にキャッシュされるのではなく、ディスカバリ(discovery)のために他のクライアントにより公表されてもよい。更に、選択的な編集を行うユーザに情報を提示し、ユーザ、管理者又はアドミニストレータが、アプリケーションのメタデータを(特定の目的に)合わせることや、アプリケーションホストがアプリケーションのメモリ割当をどのように処理するかを調整すること等を許容することも可能である。

【００２３】

静的な分析及び動的な分析を行って、アプリケーションのメモリ割当に関する追加的な情報を導出することが可能であり、この情報は、その後、アプリケーションの全体的な管理やシステムメモリ割当等の制御を支援するのに使用されてもよい。一例として、メモリが指定されている場合に、別のメモリセグメントがこのメモリ割当を満足させるのに使用されるか否かを判定することが可能である。このメモリ割当が別の割り当てに依存している場合、この依存性が通知されてもよいし、或いはこのメモリ割当が或るビットにより単に通知され、入力なしには生成されていないことを通知してもよい。静的な分析は、ソフトウェアコードがメモリを指定している場所、メモリがどのように使用されているか、どの程度メモリが満たされているか、コードパスはどのように共通しているか(例えば、ライトワンスリードメニー(write once/read many(WORM)である場合)、及びどの程度頻繁にメモリが使用されているか等を判別することが可能である。動的な分析は、全ての割り当て及び/又はアクセスを把握することが可能であり(プロファイラ[profiler]に類似している)、コードの動作によるシステム環境への影響、動作に影響するユーザの設定、及び/又は静的に決めることが適切でない又は困難であるその他のデータを取得することが可能である。

【００２４】

一実施形態において、メモリ管理システムは、アプリケーションが分析された後に、アプリケーションのメモリの利用に関するマニフェスト(manifest)又はその他の説明を出力する。これは、システムが、分析の結果をキャッシュし、将来的なアプリケーションの実行の際にその結果を再利用できるようにする。オペレーティングシステムは、バイナリ又は二進状態のアプリケーションが実行する際に(或いは一連のアプリケーションの仮想化に類似する手順の際に)この分析を最初に実行し、その後にバイナリ又は二進状態が実行する毎に利用するためにその分析を保存するように本システムに関して設計されていてもよい。カーネル又はその他のホストは、マニフェストデータを読み取り、アプリケーションがメモリをどのように使用するかを示す追加的な情報に従って適切な処理を実行することが可能である。アプリケーションの利用の仕方は時間と共に変化するかもしれないので、マニフェスト又はその他のキャッシュは刻々と動的に更新される。一実施形態において、本システムは割り当ての各々についてメモリのページ全体を割り当て、メモリアクセスの各々がページフォールト(page fault)の契機又はトリガをもたらし、メモリがどのように使用されるかをカーネルが制御できるようにし、後述されるように実際のメモリ割当とメモリに対するアプリケーションリファレンスとの間の不一致(indirection)のタイプ

を提供するようにする。

【００２５】

一実施形態において、システムはアプリケーションメモリの動作について報告を行ってもよい。これは、分析(結果)の別の利用法であり、管理者又はアドミニストレータがアプリケーションに関する判断を行うこと(例えば、いずれのサーバ又は何れの仮想マシンが適切に動作しているかについての判定)を支援する。システムは、メモリの利用に関する市場や業界の等級付け、格付け又はレーティング(rating)を提供し、例えば、移動電話アプリケーションのユーザがアプリケーションをダウンロードする前にそのアプリケーションが移動電話の利用可能なメモリをどのように利用することになるか(例えば、高頻度の利用、低頻度の利用等)を知ることができるようにする。アドミニストレータは、メモリ消費に関して収集したメタデータに基づいて行う様々なシステムに対するソフトウェアのIT開発に、この情報を利用することができる。

【００２６】

メモリ管理システムは、メモリ利用度の分析から導出した情報を利用して、システムがアイドル状態である又は十分に活用されていない場合に、先手を取って又は潜在的にメモリを割り当てることが可能である。アプリケーションによるメモリの将来的な利用の仕方を知ることは、メモリをいっそう効果的に割り当てること、及びシステムの稼働率が低い時間を活用し、システムの負荷が軽い場合に利用可能なメモリを割り当てることを、システムが実行できるようにする。これは、早期に算出された処理により、より重い負荷が、より多くの処理を行うこと又は他のリソースにアクセスすることを可能にする。

【００２７】

＜＜ホスト修正＞＞

メモリ管理システムは、アプリケーションにより提供された又はアプリケーションの分析により判定されたメモリの利用の仕方についての追加的な情報を受け取るようになされたカーネル又はアプリケーションホストに対する修正を含む。従来のソフトウェアメモリの管理とは異なり、本願により説明されるアプリケーションにより通知されるメタデータ及びその他の情報により提供されるメモリの利用の仕方に関する更なる情報又は洞察に起因して、カーネルはアプリケーションの情報や処理によらずメモリを一層効率的に管理できる。後にカーネルは如何なる処理がなされたかをアプリケーションに通知する又はメモリを管理し、アプリケーションがメモリを必要とする時点で使用できるようにし、アプリケーションがカーネルのメモリ関連処理を気にしない又は考慮しないで済むようにする。カーネルは、(例えば、重要でない又は使用される見込みがないメモリを低速/低頻度のディスクストレージに移す又はオフロードすることで)より優れたページングを実行できるようにし、メモリの要請が高まった場合にメモリを解放できるようにし、カーネルにより提供されるオペレーティングシステムを利用して1つ以上のアプリケーションに代わってメモリを管理する他の処理を実行できるようにする。例えば、カーネルは少ないフラグメンテーション(fragmentation)でメモリを適切に割り振ることが可能である。

【００２８】

メモリはオペレーティングシステムに制限されたリソースである；従って、メモリが割り当てられる場合にカーネルが適切に追跡でき、必要に応じてアプリケーションからメモリを復元できることが重要である。解決手段の1つはメモリを割り振る際にアプリケーションがメモリに優先度を指定することである。そして、メモリの利用は少ないことがシステムにより判断された場合、カーネルは、最低の優先度のメモリが何処にあるかを判別し、高い優先度のメモリを使用する他のアプリケーションのパフォーマンスに影響を与えることなく、低優先度のメモリの割り当てを解除する或いはページングを行う。

【００２９】

従来、メモリの優先度及びアプリケーションに対する優先度の割り振りはカーネルにより決定される。システムが低頻度にしかメモリを使用していなかった場合、カーネルはそのメモリを勝手に解放し、高い優先度のアプリケーションにより使用できるようにし、その結果、アプリケーションのパフォーマンスを妨げてしまう又は劣化させてしまうおそれ

がある。そうではなく、優先度の低いメモリは最初に解放されるべきである。アプリケーションが何れのメモリを解放するかをカーネルが如何にして決定するかに関し、重要な特徴がある。解決手段の1つは、メモリリソースに関し、それらの間で優先順位が何であるかをアプリケーションに決定させることである。このメモリ優先度方式は、アプリケーションが、割り当てられた又は解放されたメモリの優先度を、割り当て又は解放の時点でそれら(メモリ)に指定することを要求する。従って、メモリの要請(pressure)が高まってきた場合に、カーネルは、優先度により格付けされたメモリマップを使用して、優先度が低いメモリを最初に解放又は指定してもよい。或いは、カーネルは、解放される必要のあるアプリケーションのリストを伴う通知をアプリケーションに送付してもよい。メモリ管理システムは、オペレーティングシステムに完全に収容されていてもよいし、或いはオペレーティングシステム及びアプリケーションの間で協働していてもよい。

【００３０】

優先度モデルは、メモリ要求がなされた場合は常にアプリケーションをカプセル化する(encapsulate)メモリ割当APIを作成することで実現されてもよい。これは登録していないアプリケーションにとっても有益である。このAPIを利用することで、アプリケーション及び優先度は、アプリケーションがメモリ優先度を積極的に管理する必要なしに、サブシステムにより自動的に追跡される。カーネルのオブジェクト(kernel object)は登録、演算、処理、シグナリング、通知及びメモリ管理の機能の全てを包含し、アプリケーションの各々がそのオブジェクトを呼び出しさえすればよいようにする(或いは、修正されていないアプリケーションの場合、アプリケーションを代理するものにより呼び出されてもよい)。

【００３１】

代替的な手段はカーネルの外部にあるマスタアプリケーションを有することであり、これは現在動作している様々なアプリケーションを追跡し続けるために使用される。アプリケーションがメモリを要求すると、それらはマスタアプリケーションに登録される。そして、あるアプリケーションが応答しなくなると、マスタアプリケーションは、メモリを再利用するために、不活性なアプリケーション又はゾンビアプリケーション(zombie application)を抹消するか、或いはメモリが必要になった場合にアプリケーションに復帰できるように維持するかを判断する。

【００３２】

カーネル又はホストは、改善されたメモリ情報を利用して、様々なメモリ管理の判断を効率的又は効果的に行うことができる。例えば、低電力装置のカーネルはバッテリ寿命を延ばすためにメモリのバンク(bank of memory)をオフにすることを選択してもよい。システムはバンクに属するメモリをオフにするプロセスを不使用にする(kill)又は交換してもよい(swap)。システムは複数のバンクに分散したメモリを最適化し(defragment)、遮断できる空きバンクを形成してもよい。別の例として、カーネルは或るメモリ割当を他の装置(クラウド方式のストレージ装置を含む)に適用することを選択してもよい。多くの小さなフットプリント、低電力オペレーティングシステムはページングの概念を有しておらず、その場合、メモリ割当は他の装置/サービスにより提供される表現状態遷移(Representation State Transfer：REST)インタフェースによりサポートされてもよい。データベーススナップショットによるスクロールと同様な方法により、アプリケーションは、メモリのデータベース全体を有してはいないが、データベースサービス/サーバから導出された一部分を有する。アプリケーションはデータベース全体ではなくスナップショットを求めている情報を有していない。メモリを秘密裏に動かす又はメモリを秘密裏に解放できることは、カーネル/ホストがメモリを適切に管理できるようにしかつメモリの利用に関するアプリケーションの期待に応じられるようにする。

【００３３】

一実施形態において、システムは、メモリに加えて他のホストレベルのオブジェクト又は要素と共に使用される。例えば、システムは、ディスプレイがオフになっている場合にグラフィカル処理ユニット(GPU)を遮断し、或いは関連するハードウェアが使用されてい

ない場合、若しくは或る時間の間使用されそうにない場合に個々のドライバを遮断してもよい。システムは、システムのハイバネーション(休止状態)の間に先ずメモリを最適化し、システムが起動した際に容易に再ロードできるディスクにメモリ割当の線形ストリームを流すことも可能である。物理メモリ全体とサイズが等しいファイルを保存する従来のハイバネーション(休止状態)とは異なり、メモリ管理システムは、休止状態の前に、アプリケーションが不使用のメモリ又は容易に復元される(又は容易に復元できる)メモリを解放することを促し、休止状態で少ないデータ量しか保存しなくてよいようにする。メモリに加えて管理される他の事項は、ファイルの処理、タイマ、カーネルオブジェクト等を含んでもよい。システムはこれらの事項についての利用情報(或いは、静的及び動的な分析からそれらを判断してもよく、その場合はメモリの利用情報のみ)を受信することが可能である。

【００３４】

一実施形態において、メモリ管理システムはメモリの割当各々に関する電力状態(の情報)を受信する。例えば、電力レベルXの場合、システムは、メモリ割当はもはや必要ないと判断してもよい。これは、その電力レベルにおいて必要とされていない何らかのメモリをオフロード又は解放することで、バッテリに制約のある装置が低電力モードに切り替わるようにすることが可能である。例えば、移動電話機は、着信呼に応答する又は新規電子メールをユーザに通知する程度に十分なアプリケーションデータをメモリに維持するが、他の低優先度の機能又はアプリケーションを解放してもよい。一実施形態において、アプリケーションは、低電力状態ならばメモリを利用する前にポインタの有効性を確認する必要があるにすぎないが、より高い電力状態ならばメモリに対して通常のアクセスを行うかもしれない。これは、所与の何らかのプラットフォームに関し、アプリケーションとホストとの間の取り決めによる保証として実行されることが可能である。

【００３５】

＜＜システム構成要素及びオペレーティング環境＞＞

図1は一実施形態によるメモリ管理システムの要素を示すブロック図である。システム100は、メタデータ受信要素105、充足条件要素110、割当リクエスト要素115、メモリ参照要素120、アプリケーションインタフェース要素125、静的分析要素130、動的分析要素135、ホスト要素140、リクエスト受信要素145、リクエスト保存要素150、割当要素155、メモリ処理要素160及びデータ保存要素165を含む。以下、これらの要素の各々を詳細に説明する。

【００３６】

メタデータ受信要素105、充足条件要素110、割当リクエスト要素115、及びメモリ参照要素120は、共に、システム100がアプリケーションに提供するメモリフレームワーク又はメモリ手段を形成する。より効率的なメモリ管理を行うために開発者各自のアプリケーションコードを修正しようとしている開発者は、各自のアプリケーションからこれらの要素を利用して、ホスト又はカーネルがアプリケーションの代わりに一層効率的なメモリ管理を実行できるようにするアプリケーションを作成することが可能である。

【００３７】

メタデータ受信要素105は、各々のメモリ割当に関連する情報を受信し、メモリがアプリケーションによりどのように使用されるかを示す情報をアプリケーションホストに提供する。例えば、メタデータは、アプリケーションが如何にして容易にアプリケーションにアクセスすべきか、或いはアプリケーションが割当に関するメモリにどの程度頻繁にアクセスすることを計画又は予定しているか等を示す。メタデータは、メモリコンテンツを生成することがどの程度困難であるか、及びコンテンツが開放された場合又はディスクにページングされた場合に、アプリケーション又はホストがメモリコンテンツを置換することがどの程度困難であるか等を示していてもよい。メタデータ受信要素105は、メモリ割当API(follow up API)を呼び出す際にメタデータを受信してもよいし、或いはメモリが既に割り当てられた後にフォローアップAPIによりメタデータを提供してもよい。

【００３８】

充足条件要素(fill specification component)110は、特定のメモリ割当の内容、仕様又は条件がどのようにして満たされるか(メモリ割当に応じるか)を示す情報を受信する。メモリの内容は様々なソースから到来し、例えば、ディスクに保存されたファイルの内容を読み取ることで得られてもよいし、入力データについて1つ以上の演算を実行することで得られてもよいし、ユーザ入力から得られてもよいし、ネットワークアクセス情報(例えば、データベース又はインターネット)等から得られてもよい。一実施形態において、ホストが指定した時点でホストが関数又は機能を呼び出してメモリ内容を満たせるように、アプリケーションはメモリ充足関数(memory filling function)をホストに渡す又は通知する。処理リソース及びその他のリソースを効率的に利用するため、ホストは、リソースが低い稼働状態又はアイドルになるまでメモリを満たすことを遅延させることを選択してもよい。更に、ホストは、他の用途のためにかつて満たされたメモリをリリース又は解放する自由度(又は権限)を有し、以後アプリケーションがメモリの利用を予想する前にメモリを再度割り当てて充足し直してもよい(割当に応じてもよい)。受信したメタデータは、アプリケーションがメモリを使用することになる時をホストが知るために利用する情報を提供し、或いはアプリケーションはホストがメモリの現在の状態を検査できるようにメモリを利用する各々の試みの前にホストに通知してもよい。

【００３９】

割当リクエスト要素115は、受信したメタデータ及び充足仕様に基づいて、アプリケーションからのリクエスト又は要求を、メモリを割り当てるホストに通知する。ホストはリクエスト又は要求を受信するが、リクエストに応じて速やかにサービスを実行するか或いは他の適切な時点まで待機するかはホスト次第である点に留意を要する。極端な場合、ホストはアクセスを受ける準備が整うまで如何なるメモリも割り当てず、メモリがアプリケーションによって必要とされかつ処理を実行できるアプリケーションにメモリが割り当てられなければならない最終時点に至るまで、ホストが限られたリソースを節約できるようにしてもよい。割当リクエスト要素115は、通知されたリクエストをホストが管理しているデータ保存部に保存し、受信したメタデータ及び使用する充足条件を後のメモリ管理処理に含める。

【００４０】

メモリ参照要素120は、以前に通知された割当リクエストに従っているメモリにアプリケーションがアクセスする前に、アプリケーションから通知を受信する。ホストはメモリを利用できないようにしている或いは選択されるまでメモリを実際に割り当てることを遅らせているので、アプリケーションがメモリを利用できる準備が整った場合にメモリが利用可能になることを保証する手段を、アプリケーションは必要とする。メモリ参照要素120はこの目的に沿って機能し、特定のメモリ割当にアクセスする準備が整っていることをアプリケーションに通知できるようにする。これに応じて、ホストは(メモリが既に利用可能であった場合)実際のメモリロケーションを指すポインタをアプリケーションに通知する、或いは(メモリが今のところ割り当てられていなかった場合)受信した充足仕様及びメタデータに基づいてメモリを作成し及び充足してもよい(割当に応じてもよい)。アプリケーションはメモリを解放する通知を送信し、ホストがメモリ管理の判断にそのメモリを再び自由に組み入れることができるようにしてもよい。

【００４１】

アプリケーションインタフェース要素125は、利用するメモリの割り当て方を交渉するためにアプリケーションとホストとの間に通信インタフェースを提供する。インタフェースはアプリケーションにより使用される1つ以上の機能(関数)又はベースクラス(base class)を含み、メモリ割当を要求し、ホストにより使用される機能又はベースクラスに加えて割当についての情報を指定し、アプリケーションにより提供されるユーザ定義機能(ユーザが定義した機能又は関数)を利用してアプリケーションのメモリとやり取りを行う。アプリケーション及びホストの間の改善された通信は、アプリケーションのメモリ利用において、通常のレベルよりもかなり優れた可視性(visibility)をホストが有することを可能にし、有限のリソースを共有する様々なアプリケーションの代わりにホストがいっそう

効果的にメモリを管理できるようにする。

【００４２】

メモリ管理システム100を利用するように明確には構築されていないアプリケーションの場合、アプリケーションインタフェース要素125は、静的及び/又は動的な分析により判定された任意のアプリケーションコード手段の間の通信機能を提供し、上記と同様な方法でホストと通信できるようにする。システム100を利用するように明確には構築されていないアプリケーションの場合、システム100は、メモリ割当の目的又は他の仕様(条件)に関して多くの情報を有しておらず、システム100がアプリケーションの静的及び動的な分析により発見できる情報に限られるかもしれない。しかしながら、そのような分析でさえ、従来のアプリケーションによりも効率的にメモリ割当を管理するのに有用なメタデータを発見できる。一実施形態において、システム100は、アプリケーションが如何にしてメモリを満たしているかを自動的に検出することが可能であってもよく、アプリケーションの明示的な協働なしに、上述したのと同様なメモリコンテンツのオンデマンド充足を実行することが可能であってもよい。

【００４３】

静的分析要素130は、アプリケーションのバイナリコード又はその他のアプリケーションコードを静的に分析し、アプリケーションがメモリをどのように使用しているかを判断する。要素130は、バイナリコード(二進コード)、中間コード(例えば、マイクロソフトTM中間言語コード[MICROSOFT TM intermediate language(IL) code])、又はアプリケーションに関する他のコンパイルされた又は実行可能なバージョンを分析してもよい。静的な分析はここ数年かなり進歩し、アプリケーションバイナリが何であるか及びそれがどのようであるかを判定する多くの技術が当該技術分野に存在している。メモリ管理システム100はこれらの技術を利用してアプリケーションがメモリを割り当てて使用している場所を具体的に突き止める。要素130は、アプリケーションバイナリを利用して情報を受信し或いはアプリケーションの特定の処理を捕らえ(intercept)、捕らえた処理を新たな又は追加的な処理で置換してもよい。例えば、要素130がメモリ割当機能 (メモリ割当関数)の呼び出し(call)を発見した場合、要素130は静的な分析により利用可能な任意のメタデータを収集し、メタデータをパラメータとして受信することが可能なバージョンの割当機能(割当関数)を呼び出してもよい。このように、ホストは、そのような情報を提供するようにアプリケーションが開発者によりあたかも修正されているかのように、アプリケーションからメタデータ情報を受信する。同様に、システムがどのようにしてアプリケーションが特定のメモリ割当を充足する(割当に応じてもよい)又はアクセスするかを判断し、メモリリファレンスを示す充足仕様及び情報をホストに提供できるようにする。

【００４４】

動的分析要素135は、動作しているアプリケーションを動的に分析し、アプリケーションによるメモリの利用に関する追加的な情報であって静的な分析によっては判断することが困難な情報を収集する。しばしば、アプリケーションは静的な分析を困難にするプログラミングステップを含んでいる(意図的である場合もあるし、或いは処理のステップがそのような傾向を帯びることが事後的に判明する場合もある)。動的な分析は、外部要素から受信した応答の内容やアプリケーションにより使用されるメモリの実際の内容等のような有用な情報をもたらし、これらは静的な分析の場合には推測や近似でしかなかったものである。従って、動的な分析は、メモリの利用の仕方及びその他のアプリケーションの活動状況(これらは静的な分析によっては見出せない)を見出すことができる。要素135は動的な分析を活用して静的な分析の結果を確認又は評価してもよい。動的分析要素135は判断した情報をホストに提供し、従来のアプリケーションが、システム100により提供された少なくともいくつかの機能を利用できるようにする。

【００４５】

ホスト要素140は或る環境を含み、その環境の中でアプリケーションは動作しかつシステム100により提供されたメモリ管理にアクセスできる。プラットフォームに依存してホストは様々なソフトウェアエンティティを含むことが可能である。従来のデスクトップコ

ンピュータシステムの場合、ホストはオペレーティングシステムであり、カーネル及びその他のカーネルモードソフトウェアコード(例えば、ドライバ)を含んでもよい。管理されるソフトウェア環境に関し、ホストは、MICROSOFT TM.NET TMランタイム又はMICROSOFT TM SILVERLIGHT TMランタイム等のようなランタイム(runtime)を含んでもよい。ウェブアプリケーションは、アプリケーションホストを含んでもよく、他のサーバ又はデスクトップコンピュータ環境の中で動作するサンドボックス環境(sandboxed environment)で動作してもよい。サーバコンピュータ環境は、初期化ハードウェア又はソフトウェアを含み、1つ以上のオペレーティングシステムカーネルよりも高いレベルで動作するハイパーバイザを含んでもよい。メモリ管理システム100には、様々なレベルのソフトウェアコードが実装され、何らかの特定のタイプのホストには限定されない。ホスト要素140は、システム100が動作している特定のプラットフォームのリソースを管理しかつアプリケーションとやり取りを行う責務を担う要素を表現する。

【００４６】

リクエスト受信要素145、リクエスト保存要素150、割当要素155及びメモリ処理要素160は、共に、上記のシステム100を実現するように修正されるメモリマネジャの要素を形成する。

【００４７】

リクエスト受信要素145は、メモリを割り当てることを求めるリクエスト又は要求をアプリケーションから受信する。リクエストの各々は、割り当てられたメモリをアプリケーションがどのように使用するかを示す情報をホストに提供するメタデータを含む。リクエスト受信要素145は特定のホスト上で動作している複数の又は全てのアプリケーションからリクエストを受信し、メモリが如何にして使用されているかを示す情報を広範囲にわたって取得できるようにする。その後ホストは、(例えば、メモリの逼迫(pressure)を緩和するためにメモリをページアウトする(paging out)又は解放する等のような)特定の処理を行うことに何れの割り当てが影響するかについて優先順位を付けることが可能である。

【００４８】

リクエスト保存要素150は、受信したリクエスト及び関連するメタデータ情報を、以後のメモリ管理処理における使用に備えて保存する。ホストはリクエスト(要求)が受信された際に割当要求を満たしていてもいなくてもよい。しかしながら、ホストが速やかに又は後にリクエストに応じるか否かによらず、ホストは割当に関してアプリケーションから提供された情報を保存し、メモリ処理が適切である後の時点でホストが利用可能な情報を有するようにする。一実施形態において、ホストは、アプリケーションがメモリを割り当てた後に、別個の通知によりメタデータ及びその他の情報をアプリケーションから受信する。その場合、ホストは割当情報を複数回保存してもよい或いは有用な情報が利用可能である。

【００４９】

割当要素155は1つ以上の受信したアプリケーション割当リクエストを満足させるようにメモリを割り振る実際の処理を実行する。割当要素155は、アプリケーション群又はアプリケーションヒープ(application heap)によりメモリを割り当ててもよいし、或いはホストによりアクセスされ管理されるコンピュータシステムにインストールされた物理メモリから直接的にメモリを割り当ててもよい。割当の後、メモリは、特定のアプリケーションに拘束され、(通常の共有メモリ方式により他のアプリケーションにより利用することが望まれない限り)他のアプリケーションによっては利用できなくなる。割当要素155は、ページテーブル及び仮想メモリを利用して割当に関するウィンドウを物理メモリに提供し、ディスクにより又はスワップファイルを通じて他のストレージにより、提供されるメモリが返されてもよい。

【００５０】

メモリ処理要素160は、装置のメモリを管理する処理を実行し、以前に受信され保存されたリクエスト情報にアクセスし、実行される処理の影響を受ける1つ以上の適切なメモリ割当を決定する。この処理は、メモリコンテンツをスワップファイルと置換すること、

以前に割り当てたメモリを解放すること、アプリケーションがメモリのフットプリントを減らすのを要求すること、及びその他の処理(1つ以上のアプリケーションにより使用されるメモリに影響する処理)を含んでもよい。典型的には、メモリ管理処理を実行するホストの目的は、メモリ利用度を下げること(例えば、メモリが逼迫している場合)、アプリケーションの要請を満足する程度に十分なメモリを収集すること等のような現在の又は差し迫った状況に対処することである。

【００５１】

データ保存要素165はメモリを管理するために1つ以上のアプリケーションの代わりにホストにより使用される情報を保存する。データ保存要素165は、システム100により使用するのとは別に設定されたメモリを含み、フラッシュ又はディスクストレージ或いはデータを保存する他の手段を区別する。データ保存要素165は、保存されるデータをアプリケーション各々のメモリ空間に含め、アプリケーションの代わりにメモリを管理するために、ホストにより使用されるアプリケーション及びその他の情報に関連するアプリケーションを追跡する。

【００５２】

メモリ管理システムが実装されるコンピュータ装置は、中央処理装置、メモリ、入力装置(例えば、キーボード及びポインティングデバイス)、出力装置(例えば、ディスプレイ装置)及びストレージ装置(例えば、ディスクドライブ又はその他の不揮発性記憶媒体)を含んでもよい。メモリ及びストレージ装置(記憶装置)は、コンピュータにより読み取られることが可能な記憶媒体(コンピュータ可読媒体)であり、システムを実現する又は実行可能にするコンピュータ実行可能命令(例えば、ソフトウェア)(コンピュータにより実行されることが可能な命令)と共にエンコードされていてもよい。更に、データ構造及びメッセージ構造は、通信リンクにおける信号のようなデータ伝送媒体を通じて保存又は伝送されてもよい。インターネット、ローカルエリアネットワーク、ワイドエリアネットワーク、1対1ダイヤルアップ接続、セルラ電話ネットワーク等のような様々な通信リンクが使用されてもよい。

【００５３】

システムの実施形態は様々なオペレーティング環境で実現されてもよく、オペレーティング環境は、パーソナルコンピュータ、サーバコンピュータ、携帯装置、ラップトップ装置、マルチプロセッサシステム、マイクロプロセッサ方式のシステム、プログラマブル消費者電子装置、ディジタルカメラ、ネットワークPC、ミニコンピュータ、メインフレームコンピュータ、上記のシステムや装置の内の任意のものを含む分散されたコンピュータ環境、セットトップボックス、システムオンチップ(SOC)等を含む。コンピュータシステムは、セルラ電話機、パーソナルディジタルアシスタント、スマートフォン、パーソナルコンピュータ、プログラマブル消費者電子装置、ディジタルカメラ等であってもよい。

【００５４】

本システムは、1つ以上のコンピュータ又はその他の装置により実行されるコンピュータ実行可能命令(例えば、プログラムモジュール)の観点から一般的に説明されてもよい。概して、プログラムモジュールは、ルーチン、プログラム、オブジェクト、コンポーネント、要素、データ構造等(特定のタスクを実行する或いは特定のアブストラクトデータタイプを実現するもの)を含んでもよい。典型的には、プログラムモジュールの機能は様々な実施形態で望まれるように組み合わせられたり分散されたりしてよい。

【００５５】

図2は一実施形態によるメモリ管理システムの動作環境を示すブロック図である。この環境は、カーネル/ホスト210、1つ以上のアプリケーション220、メモリ処理フレームワーク(メモリ処理手段)230、保存された割当メタデータ240、及び1つ以上のツール250を含む。カーネル/ホスト210は1つ上のアプリケーション220により共有されるメモリを管理する。メモリ処理フレームワーク230は1つ以上のAPIを提供し、アプリケーション220は、その1つ以上のAPIを呼び出してメモリを割り振ってメモリ割当メタデータを提供する。メモリ処理フレームワーク230は受信したメタデータ240を保存する。ツール250は、割当情報を

提供するように元々は設計されていないアプリケーション上で動作し、静的及び動的な分析を通じてそのような情報を抽出する。以後、抽出された如何なる情報も割当メタデータ240として保存される。

＜＜システム動作＞＞

図3は、メモリの割当及び利用を要求するためにソフトウェアアプリケーションにおいて行われるメモリ管理システムの処理を示す一実施形態によるフローチャートを示す。

【００５６】

始めに、ブロック310において、アプリケーションがメモリ割当をどのように使用することになるかを示す割当メタデータを、アプリケーションが設定する。アプリケーションは、割当関数(allocation function)を渡すパラメータ構造を構築し、割当メタデータを提供するためのAPIを呼び出す、或いは上記のメモリ管理システムによりメモリを割り振るために導出されたクラスのパラメータを設定する。割当メタデータは、アプリケーションに対するメモリの優先度、割り当てられたメモリをアプリケーションがどの程度頻繁に利用することを予定しているか、アプリケーションの内容(コンテンツ)を置換することがアプリケーションにとってどの程度困難であるか、アプリケーションによるメモリの利用に関連するその他の情報、及び、複数のアプリケーションにより共有される物理メモリリソースを効率的に使用するためにホストがメモリをどのように処理できるか等の情報を含んでもよい。

【００５７】

次に、ブロック320において、アプリケーションは、メモリ割当の内容(コンテンツ)を満たすメモリ充足関数又はメモリ充足機能(memory fill function)を設定する。メモリ充足機能は、ファイルからの情報にアクセスし、1つ以上の演算を実行し、メモリ割当に保存される結果を生成する、或いはメモリ割当を満たす他の処理を実行してもよい。メモリ充足機能を割当の直後に起動するのではなく、メモリ充足機能をホストに提供することで、アプリケーションは、メモリを解放及び再配置するのに必要な情報をホストに提供し、或いはメモリの割当及び初期配置をホストにとっていっそう相応しい時点まで遅らせる。

【００５８】

次に、ブロック330において、アプリケーションはホストにより提供される割当インタフェースを起動し、割当インタフェースは、複数のアプリケーションの間で共有される物理メモリを管理する。割当インタフェースは、オペレーティングシステムAPI、ランタイム機能、又はメモリ管理を担うホストとアプリケーションとの間で情報をやり取りする他の機能を含んでよい。割当インタフェースは、現在のメモリ割当で通常提供されているよりも多くの情報を提供し、コンピュータ装置でメモリをいっそう効率的に管理するのに有用なホスト情報をもたらす。

【００５９】

次に、ブロック340において、アプリケーションは割当インタフェースを起動したことに応じてリファレンスを受信し、リファレンスはアプリケーションにより割り当てられる以後のメモリについての間接的な識別子として機能する。ホストは要求を受信した場合にメモリを速やかに割り当てても割り当てなくてもよい。更に、アプリケーションがメモリを利用する前にホストが再び制御を受ける必要があるような方法で、ホストは(例えば、メモリを解放する、ディスクにページングする等により)刻々とメモリを修正してもよい。従って、ホストはメモリを指すリファレンスをアプリケーションに提供し、アプリケーションがメモリを利用することを希望した場合に、アプリケーションはそのリファレンスを用いてそのメモリに対する直接的なアクセスを行う(すなわち、ポインタとして利用できる)。

【００６０】

次に、判定ブロック350において、割り当てられたメモリとの関係をアプリケーションが絶ったこと(メモリとの関係が終わったこと)をアプリケーションが検出した場合、アプリケーションは終了し、そうでなかった場合アプリケーションはブロック360に続く。終

了する前に、アプリケーションは、以前に割り当てたメモリの割当を解除(解放)するために割当インタフェースを起動する。アプリケーションがメモリを利用していなかった場合、及びホストがメモリを実際に割り当てていなかった場合、この処理は、割当に関してホストが保存したエントリを単に除去し、アプリケーションを制御する処理に戻る。

【００６１】

次に、判定ブロック360において、アプリケーションが、アプリケーションにより割り当てたメモリの利用を検出した場合、アプリケーションはブロック370に続き、そうでなかった場合、アプリケーションはブロック350に戻り、メモリアクセス又はメモリ割当の目的が達成されることを待機する。アプリケーションは、メモリがアプリケーションソフトウェアコードにおいてアクセスされた場所が何処であっても特定の機能を呼び出す、或いはメモリ割当を或るクラスにカプセル化し(encapsulate)、ホストに対して適切な呼び出し(call)と共にメモリのアクセスをラッピングし(wrap)、メモリを利用可能にする。

【００６２】

次に、ブロック370において、アプリケーションは、ホストからのメモリ割当に対する直接的なアクセスを要求する。ホストがメモリを既に割り当てていた場合、ホストは、メモリがアクセスできるポインタをアプリケーションに返す。ホストがメモリを割り当てていなかった場合又はメモリの割当を解除して再利用する場合、ホストはアプリケーションリクエストに応じてメモリを割り振り、受け取った充足機能を起動してメモリ内容を満たし、メモリにアクセスするポインタ又はその他の手段をアプリケーションに返す。

【００６３】

次に、ブロック380において、アプリケーションは、メモリに関してホストから受信したアドレスを通じてメモリ割当にアクセスする。アプリケーションは、メモリを修正する、メモリから情報を読み取る、或いはその他の一般的なメモリの動作を実行してもよい。アプリケーションがメモリにアクセスすると、アプリケーションは、メモリを利用できないようにしてもよい管理動作をホストが再び実行できることをホストに通知してもよい。アプリケーションは、ホストが再びメモリを再構築できるように充足機能(関数)により使用されるデータを更新してもよく、必要であれば、メモリは最新の変更内容を含む。ブロック380の後、これらのステップは終了する。

【００６４】

図4は、メモリの割当及び利用を要求するためのアプリケーション要求を受信するホストにおいて行われるメモリ管理システムの処理を示す一実施形態によるフローチャートを示す。始めに、ブロック410において、ホストはメモリ割当リクエストをアプリケーションから受信する。ホストは複数のアプリケーションからのリクエストを処理し、ホストはメモリマネジャを含み、メモリマネジャは、ホストのプラットフォーム上で動作している複数のアプリケーションの中で、限られた物理メモリをスライスする又は共有する。メモリ割当リクエストは、要求されているメモリサイズ及びその他のメモリ要求に関連するパラメータ等のような情報を含んでもよい。

【００６５】

次に、ブロック420において、ホストはメモリ割当メタデータを受信し、メモリ割当メタデータは、要求されたメモリ割当を利用するためにアプリケーションが計画又は予定している方法を指定する。メタデータは、アプリケーションに対するメモリ割当の重要度又は優先度、割当がディスクに対するページングに相応しいか否か、ホストが割当を解放しなければならない場合にアプリケーションがメモリの内容を復元できるか否か等を示す情報を含んでもよい。

【００６６】

次に、ブロック430において、要求されたメモリ割当(コンテンツ)に応じるためにホストにより起動できるメモリ充足機能をアプリケーションから受け入れる。メモリ割当を充足する手段(割当に応じて割当を行う手段)を有することで、他のアプリケーションのリクエストを満足するためにホストがメモリ割当を解放する又は他のメモリの目的を緩和/排除する必要がある場合に、ホストがメモリの割当を遅延させるだけでなくメモリを復元す

ることさえ可能になる。

【００６７】

次に、ブロック440において、ホストがリクエストに応じて直接的にメモリを割り当てることが可能であるとホストが判断した場合、ホストはブロック450に進み、そうでなかった場合ホストはブロック460に進む。通知されたアプリケーションにとってメモリがアプリケーションにどの程度重要であるか、及びアプリケーションがメモリにどの程度頻繁にアクセスすることを予定しているかに基づいて、ホストはメモリを直接的に割り当てるか否かを判断してもよい。ホストは、ホストが動作しているコンピュータ装置がリクエストの時点でどの程度忙しいか等のような他の要因を考慮してもよい。

【００６８】

次に、ブロック450において、ホストは、ホストにとって利用可能な物理メモリのうち要求されたメモリを割り当てる。ホストは、アプリケーションが物理メモリにアクセスする際のページテーブル又はその他の仮想メモリを設定してもよい。従来のホストは、アプリケーションのリクエストに直接的に応答してメモリを割り当てるが、ホストがリクエストに応じることができなかった場合、そのリクエストは失敗に終わる。しかしながら、本メモリ管理システムは、メモリ及びその他のホストリソースをいっそう効率的に利用するために、リクエスト(に応じること)を遅らせる或いは他のメモリ管理処理を実行してもよい。

【００６９】

次に、ブロック460において、ホストは、受信したメタデータ及び充足機能と共に、後の割り当てに備えて受信したリクエストを保存する。ホストは、メモリ管理処理(より多くのメモリを利用可能にすること等)が必要とされる場合に選択する様々なアプリケーションにより要求されるメモリのデータ保存を保持してもよい。アプリケーションにより提供されるメタデータは、アプリケーションの中で何れのアプリケーション及び割り当てが、ホストのメモリ管理処理により最小の影響しかもたらさないかを、ホストが判断することを促す。ブロック460の後、これらのステップは終了する。

【００７０】

場合によっては、アプリケーションの開発者は、新たなオペレーティングシステムの特徴(機能)をサポートするように(使用できるように)変更を加えることを意図していない又は不可能である。そのような場合であっても、図5及び図6を参照しながら詳細に説明されるように、メモリ管理システムは、アプリケーションバイナリ情報を分析してメモリ割当に関する情報を発見することで、何らかの改善されたメモリ管理を行うことが可能である。

【００７１】

図5は、メモリ割当情報を提供するように明確には設計されていないアプリケーションを分析するメモリ管理システムの処理を示す一実施形態によるフローチャートである。

【００７２】

始めに、ブロック510において、アプリケーションにより実行されるメモリ割当を記述する(示す)情報が利用可能でないアプリケーションを検出する。アプリケーションは、メモリ管理システムにより提供されているメモリモデルにアプリケーションが加わっているか否か(含まれているか否か)を指定するフラグ又は他の情報をバイナリモジュールに含んでいてもよいし、或いはその他の指示情報を提供していてもよい。本システムは、以前に決定したアプリケーション割当情報をキャッシュし(一時記憶し)、アプリケーション各々に対する分析が一度しか実行されないようにする。従ってこのステップにおいてシステムは分析が以前には実行されていないことも確認又は判断する。

【００７３】

次に、ブロック520において、システムは、アプリケーションについて分析を行い、アプリケーションによりなされるメモリ割当を確認する。この分析はアプリケーションの静的及び/又は動的な分析による様々な形態を含んでよい。静的な分析は、アプリケーションを動作させることなく行われ、アプリケーションのバイナリコードを検査し、アプリケ

ーションの動作を確認する。動的な分析は、アプリケーションが動作している間に行われ、アプリケーションのメモリ内コード(in-memory code)、データ構造及びその他の情報を検査し、アプリケーションの動作を確認する。システムは、メモリを割り当てること、メモリにアクセスすること及びメモリを解放することのための特定のホストAPIを呼び出すコール(call)を探し、ロケーション、関連する補足情報及び処理ステップ等のメモリの割り当てに使用される事項を記録する。

【００７４】

ブロック530において、システムはアプリケーションを使用して(hook)、アプリケーションの中でアプリケーションがメモリを割り振るロケーションにある割当情報を提供する。例えばマイクロソフトTMデートゥア(MICROSOFT TM Detours)等のような様々な形態のアプリケーションフッキング技術(application hooking technology)が利用可能であり、これは、アプリケーションのバイナリコードのリダイレクションが、プログラムの任意のポイントでアプリケーションの通常の動作を捕らえる(intercept)又は補強する(augment)ことを可能にする。例えば、システムは、アプリケーションにより元々呼び出されたメタデータなしに、標準的な割当機能ではなく、割り当てメタデータを提供する割当機能を起動するアプリケーションフックを提供してもよい。

【００７５】

次に、ブロック540において、システムは、メモリを割り当てるためのアプリケーションからのリクエスト、及びフックされたアプリケーションコードにより提供される関連割当情報を受信する。リクエストはホストにより受信され、ホストは、割当情報を提供するように明示的には設計されていないアプリケーションから割当情報が到来していることを判断できてもよい。ホストは、そのようなアプリケーションについて、システムと共に動作するように設計されているアプリケーションとは異なるように(別様に)動作してもよい。例えば、ホストは、システムと共に動作するように設計されていないアプリケーションにより提供される情報を、潜在的にほとんど把握しなくてもよい。一実施形態において、新たなメモリモデルの採用を促すため、システムは、メモリの逼迫が生じてきた場合に、それらのデータをディスクに交換することを優先することで、或いは他のメモリ管理処理を実行することで、新たなメモリモデルに対応していないアプリケーションにペナルティを課してもよい。ブロック540の後、これらのステップは終了する。

【００７６】

図6は、アプリケーションを静的に分析し及び改善されたメモリ情報のマニフェストを提供するメモリ管理システムの処理を示す一実施形態によるフローチャートである。

【００７７】

始めに、ブロック610において、システムは、メモリ割当機能を呼び出すコール(call)に関してメモリ割当情報を提供しないコンパイルされたアプリケーションコードを受信する。システムは、コンピュータ装置のハードドライブ又はその他のストレージのスキャンによるバルク処理の最中に、或いは特定のユーザ又はアドミニストレータのリクエストに応答して、アプリケーションを動作させる第1のリクエストによりアプリケーションコードを受信する。コンパイルされたアプリケーションコードは、特定のプロセッサのバイナリコード(例えば、x86又はx64バイナリコード)、特定のランタイムの中間言語コード、或いは非ソースアプリケーションコードによる他の形態を含んでもよい。ソースが利用可能である場合、アプリケーションは、本願の他の箇所で説明されているようにシステムを利用する直接的な修正を受けてもよい。

【００７８】

次に、ブロック620において、システムは受信したアプリケーションコードについて静的な分析を実行し、コードの中でアプリケーションがメモリを割り当てるロケーションを確認する。静的な分析は、インポートされたモジュール又はその他の手段を通じて、メモリ割当機能を呼び出すコールを探す。静的な分析は、機能を呼び出すコールを探すだけでなく、場合によっては通知されるパラメータを探し出すことにも優れている。一実施形態において、システムは、動的な分析と共に静的な分析を改善し、具体的には、アプリケー

ションを実行し、静的には利用可能でない或いは静的には容易に検出できない情報を確認する。

【００７９】

次に、ブロック630において、システムは、分析したアプリケーションコードにおいて1つ以上のメモリ関連コード処理を確認する。コード処理(code action)は、メモリを割り当てること、メモリにアクセスすること、メモリを解放すること、及びその他のメモリに関する処理を含んでもよい。システムは、処理が生じるアドレスによりコード処理を特定してもよいし、或いはホストに後に提供できる他の識別情報又はデフォルトアプリケーション処理を変更する際に修正される識別情報により、コード処理を特定してもよい。

【００８０】

次に，ブロック640において、システムは確認されたコード処理各々に関連する関連情報(surrounding information)を特定し、関連情報は、割り当てられたメモリがアプリケーションによりどのように使用されるかを示す追加的な情報を提供する。関連情報は、アプリケーションがどの程度長い間保存されているか(例えば、同じ機能の中で一度使用されている、或いは反復的な以後の利用の際にグローバル変数に保存されている)、メモリ割当に応じることがアプリケーションにとってどの程度容易であるか等をシステムに伝える。関連情報は静的及び/又は動的な分析により確認できる。

【００８１】

次に、ブロック650において、システムは、アプリケーションを実行させているホストが後に読み出すことに備えて、確認したメモリ関連コード処理と何らかの確認された関連情報とをデータ保存部に保存する。一実施形態において、システムは、確認した情報を、アプリケーションモジュールと共に保存されているマニフェストに保存し、アプリケーションをロードしているホストが、確認した場所(ロケーション)について何らかのインターセプト(interception)又はフッキング(hooking)を実行し、アプリケーションにより行われるメモリ割当を示す豊富な情報をホストに提供する。ホストはこの情報を利用して例えば先手を打ってメモリを割り当てる一方、システムは分析により判明したようにアプリケーションが将来のある時点で必要とされるになるまでアイドルである又は低い稼働状態である。これは、本メモリ管理システムにより提供されるメモリモデルに、従来のアプリケーションが参加できるようにする。ブロック650の後、これらのステップは終了する。

【００８２】

図7は、検出されたメモリの逼迫に応じてメモリに関して処理を行うメモリ管理システムの処理を示す一実施形態によるフローチャートを示す。限られたメモリしか備わっていない装置が、利用可能なメモリの限度に近づきつつある場合、メモリの逼迫が生じる。例えば、システムのオペレーティングシステムは、物理RAMの90％以上が使用されている場合に、メモリの逼迫が生じていると定めてもよい。

【００８３】

始めに、ブロック710において、複数のメモリ割当リクエストを示す情報と、メモリ割当が1つ以上のアプリケーションによりどのように使用されるかを示す情報とを、システムが受信する。場合によっては、システムは、アプリケーションから受信される割り当てのリスト又はその他のデータ構造を保存することに加えて、アプリケーションの各々がどのように使用されるかを示すメタデータも保存する。割当利用情報は、何れの割当を解放できるか、何れの割当についてページングできるか、或いは利用可能なメモリを効率的に管理するために何れの割当について他の処理が可能であるか等を、システムが決定することを促す優先度又はその他の情報を含む。

【００８４】

次に、ブロック720において、システムはメモリの逼迫を検出し、これは、アプリケーションを効率的又は効果的に実行させ続けるための処理が必要であることを示す。検出されたメモリの逼迫は、カーネル又はその他のホストが検出できる様々な状態又は動作を含んでもよい。例えば、カーネルはメモリのバンクを節電状態に切り替えることを希望してもよいし、バンクに保存された割当を検出し、それらの割当を解放する又はディスクと交

換し、バンクを切り替えることができるようにしてもよい。別の例として、システムはメモリバンク(複数)を最適化し、スイッチオフにできる空のバンクを形成してもよい。ホストがメモリの利用を完全には制御しない場合、アプリケーションとの協働なしに、メモリを移動させる又は解放する処理を実行することはしばしば困難である。しかしながら、本願で説明されているメモリ管理システムは、ホストに豊富な情報をもたらし、メモリがどのように管理されるかという観点からの制御を上回る優れた制御を行うことができる。

【００８５】

次に、ブロック730において、システムは処理の対象となる割当を決定するために受信した割当リクエストを列挙する。システムは、割当に関するリスト又はその他のデータ構造を一通り検査し、解放される必要がある場合に容易に戻せる割当、又はアプリケーションが再度使用する見込みの薄い割当を判別する。

【００８６】

次に、ブロック740において、システムは列挙された割当のうち処理対象となる1つ以上を選択する。アプリケーションにより要求された割当がそのアプリケーションによりどのように使用されているかを示す受信した情報に基づいて、システムは割当を選択してもよい。場合によっては、システムは、メモリマネジャによる特定の目的又は基準に従って複数の割当を選択してもよい(特定の目的又は基準は、例えば、特定の全体サイズのメモリについて処理を行うことや、あるサイズに達している又は超えている割当を発見すること等である)。

【００８７】

次に、ブロック750において、システムは選択した割当について処理を実行し、メモリの逼迫を緩和する。処理は、以前に割り当てたメモリを解放すること、メモリ内容をディスク又は他のストレージに移すこと、メモリを以前のロケーションから新たなロケーションに移動させること等である。場合によっては、システムはアプリケーションに処理を通知し、アプリケーションが、割当に依存する処理を修正できるようにする。例えば、アプリケーションは、アプリケーションにより次回使用されるメモリを割り当て直すことで対応してもよい。ブロック750の後、これらの処理は終了する。

【００８８】

図8は、メモリがホストによって事前に修正されているアプリケーションをアクティブ化するメモリ管理システムの処理を示す一実施形態によるフローチャートを示す。多くのマルチタスクシステムにおいて、アプリケーションはバックグランドに追いやられ又はプッシュされ、後にユーザにより又はオペレーティングシステムにより再度アクティブ化される。モバイル装置の場合、ユーザは一度に1つのアプリケーションとやり取りを行い、オペレーティングシステムは、特定のアプリケーションを最前面(fore)にする一方、他のアプリケーションを可能性のある候補のままにする。オペレーティングシステムは、他のアプリケーションのメモリ又はストリームを、ディスク又は他のストレージに移す又は解放してもよい。(他のアプリケーションを)アクティブ化又は活性化する際、オペレーティングシステムはそのアプリケーションを再び動作させるように準備する処理を行う。

【００８９】

始めに、ブロック810において、システムは、アクティブ化するためのアプリケーションリクエストを受信する。リクエストは、アプリケーション自身から到来するかもしれないし、ユーザから或いはオペレーティングシステムから来るかもしれない。アクティブ化するためのリクエストは、ユーザがアプリケーションと再びやり取りを行おうとしていることを単に示し、アプリケーションが以前に割り当てたメモリを要求するようにしてもよい。

【００９０】

次に、ブロック820において、システムは1つ以上の過去に受信したアプリケーションメモリ割当を確認する。システムは、1つ以上の割当により要求される割当各々のリスト又は他のデータ構造を維持し、システムがリストを一通り検査する(走査する)ことで、以前に受信したメモリ割当を確認できるようにしてもよい。システムは割当各々の状態を検査

し、例えば、割当に関連するメモリが依然として利用可能であるか否か、割当に関連するメモリがアプリケーションにより最後に設定された内容を依然として有しているか否か等を検査する。

【００９１】

次に、判定ブロック830において、アプリケーションの割当が全て準備できているとシステムが判断した場合、システムはブロック860にジャンプし、そうでなかった場合システムはブロック840に進む。システムは、アプリケーション割当について、解放、移動又はそれ以外の処理を実行し、例えば他のアプリケーションを実行させることのような他のタスクのために更なるメモリを用意又は提供してもよい。そして、アプリケーションを再び実行させる時に至った場合、システムは、以前になされていた割当を再現する或いはアプリケーションにその割当を通知し、アプリケーションが適切な割当を実行できるようにしてもよい。

【００９２】

次に、ブロック840において、システムは、準備が未完了の割当に割当を行い、アプリケーションにより予想されている状態に割当を設定する。システムは、割当に応じるように利用可能な物理メモリを割り当てることで、物理メモリをその割当に指定してもよい。場合によっては、システムは、例えば、物理メモリ及びスワップファイルにより戻される(backed)もののような仮想メモリを提供してもよい。

【００９３】

次に、ブロック850において、システムは、アプリケーションにより提供された充足機能(fill function)を利用して、割り当てられたメモリ内容に応じる。アプリケーションは、システムがアプリケーションのメモリ割当を解放及び再構築できる程度に十分な情報を、システムに提供する。これは、アプリケーション同士の間でリソースを共有しつつ、アプリケーションに関する不都合な影響を小さく維持できるように、システムが効果的な判断を行えるようにする。理想的には、システムによるメモリの稼働率が低い場合又は他の要請がある場合に、システムが、いずれにせよ直ぐには使用されそうにないメモリを解放又は移動させ、再び必要とされる前に、メモリを復帰させる時間を確保する。従来のシステムの場合、ホストが実行できる最良の処理は推測であるにすぎないが、本願によるメモリ管理システムを利用すると、ホストは処理対象とする1つ以上のメモリ割当を非常に効果的に選択できる。

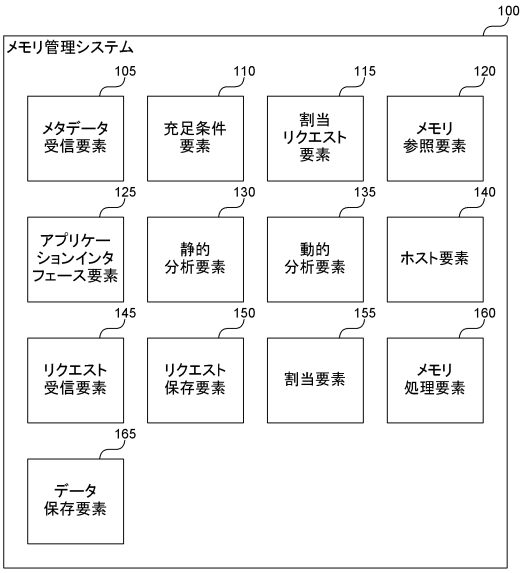
【００９４】

次に、ブロック860において、システムは，要求されたアプリケーションをアクティブ化し、アプリケーションが期待するメモリ割当をアプリケーションに提供する。システムは、アプリケーションを中断する一方、アプリケーションのためにメモリを準備し、その後、全てのアプリケーションのメモリ割当の準備が整った場合にアプリケーションを再開してもよい。

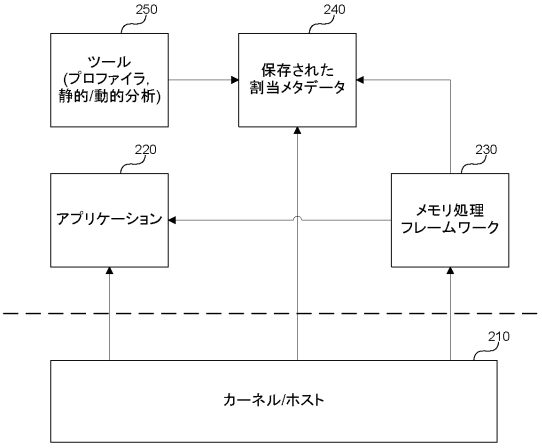
【００９５】

以上、メモリ管理システムに関する具体的な実施形態が例示を意図して説明されてきたが、本発明の精神及び範囲から逸脱することなく、様々な変形がなされてよいことが、認められるであろう。すなわち、本発明は添付の特許請求の範囲以外によっては限定されない。

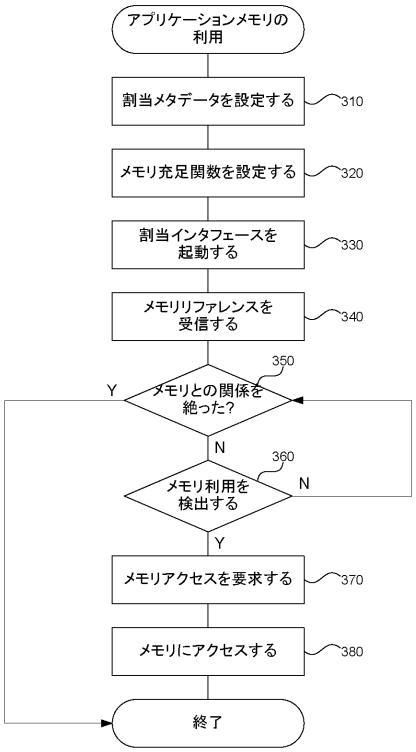
【図１】


100
メモリ管理システム
105
メタデータ受信要素
110
充足条件要素
115
割当リクエスト要素
120
メモリ参照要素
125
アプリケーションインタフェース要素
130
静的分析要素
135
動的分析要素
140
ホスト要素
145
リクエスト受信要素
150
リクエスト保存要素
155
割当要素
160
メモリ処理要素
165
データ保存要素

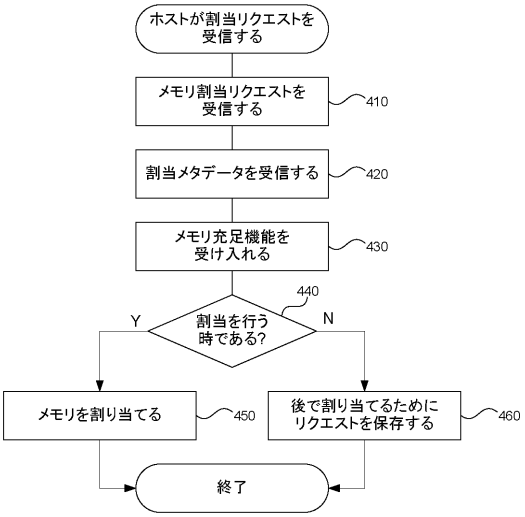
【図２】


250
ツール(プロファイラ,静的/動的分析)
240
保存された割当メタデータ
220
アプリケーション
230
メモリ処理フレームワーク
210
カーネル/ホスト

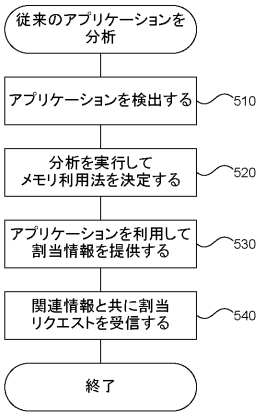
【図３】


アプリケーションメモリの利用
割当メタデータを設定する
310
メモリ充足関数を設定する
320
割当インタフェースを起動する
330
メモリリファレンスを受信する
340
350
メモリとの関係を絶った?
Y
N
360
メモリ利用を検出する
N
Y
メモリアクセスを要求する
370
メモリにアクセスする
380
終了

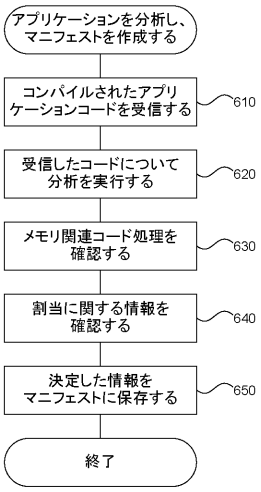
【図４】


ホストが割当リクエストを受信する
メモリ割当リクエストを受信する
410
割当メタデータを受信する
420
メモリ充足機能を受け入れる
430
440
割当を行う時である?
Y
N
メモリを割り当てる
450
後で割り当てるためにリクエストを保存する
460
終了

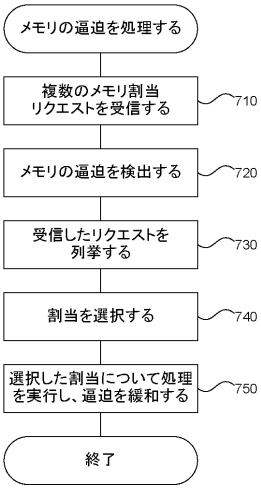
【図５】


従来のアプリケーションを分析
アプリケーションを検出する
510
分析を実行してメモリ利用法を決定する
520
アプリケーションを利用して割当情報を提供する
530
関連情報と共に割当リクエストを受信する
540
終了

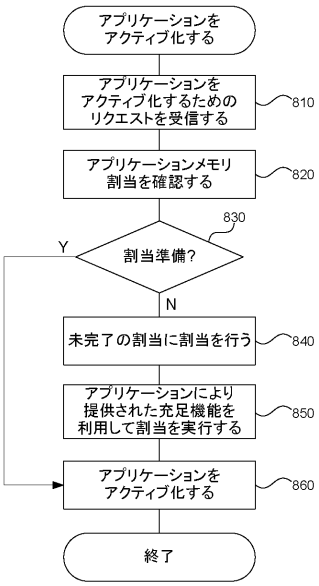
【図６】


アプリケーションを分析し、マニフェストを作成する
コンパイルされたアプリケーションコードを受信する
610
受信したコードについて分析を実行する
620
メモリ関連コード処理を確認する
630
割当に関する情報を確認する
640
決定した情報をマニフェストに保存する
650
終了

【図７】


メモリの逼迫を処理する
複数のメモリ割当リクエストを受信する
710
メモリの逼迫を検出する
720
受信したリクエストを列挙する
730
割当を選択する
740
選択した割当について処理を実行し、逼迫を緩和する
750
終了

【図８】


アプリケーションをアクティブ化する
アプリケーションをアクティブ化するためのリクエストを受信する
810
アプリケーションメモリ割当を確認する
820
830
割当準備?
Y
N
未完了の割当に割当を行う
840
アプリケーションにより提供された充足機能を利用して割当を実行する
850
アプリケーションをアクティブ化する
860
終了

フロントページの続き

(72)発明者 スプラドリン, ジェレマイア シー.
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 ハント, ギャレン
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 トマス, アニル, エフ.
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内
(72)発明者 メイレット, スティーヴン
アメリカ合衆国 ９８０５２－６３９９ ワシントン州 レッドモンド ワン マイクロソフト ウェイ マイクロソフト コーポレーション エルシーエー－インターナショナル パテンツ 内

審査官 多賀 実

(56)参考文献 特開２００３－２４１９６７（ＪＰ，Ａ）
特表２００９－５４０４６３（ＪＰ，Ａ）
特開平１０－２６９１０５（ＪＰ，Ａ）
特開平０８－２９７６０４（ＪＰ，Ａ）
特表２００９－５４０４６４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ１１／２８－１１／３６
Ｇ０６Ｆ１２／００－１２／０２
Ｇ０６Ｆ ９／４４
Ｇ０６Ｆ ９／５０