(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6258940号
(P6258940)
(45)発行日　平成30年1月10日(2018.1.10)　(24)登録日　平成29年12月15日(2017.12.15)

(51)Int.Cl.　F I
G06F 9/4401 (2018.01)　G06F 9/06 610K

請求項の数 15 (全 25 頁)

(21)出願番号　特願2015-529856(P2015-529856)
(86)(22)出願日　平成25年8月19日(2013.8.19)
(65)公表番号　特表2015-526827(P2015-526827A)
(43)公表日　平成27年9月10日(2015.9.10)
(86)国際出願番号　PCT/US2013/055636
(87)国際公開番号　WO2014/035711
(87)国際公開日　平成26年3月6日(2014.3.6)
審査請求日　平成28年8月12日(2016.8.12)
(31)優先権主張番号　13/598,616
(32)優先日　平成24年8月30日(2012.8.30)
(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　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100101373
弁理士　竹内　茂雄
(74)代理人　100118902
弁理士　山本　修

最終頁に続く

(54)【発明の名称】ＢＰＲＡＭを使用したソフトウェア・アプリケーションのレイアウトおよび実行

(57)【特許請求の範囲】
【請求項１】
アプリケーションを検査してバイト・アドレス可能永続ランダム・アクセス・メモリー（ＢＰＲＡＭ）における事前レイアウトのためにコードおよびデーターを識別するコンピューター実装方法であって、
ターゲット・アプリケーションの高速初期化イメージを作るために高速レイアウトを実行する前記ターゲット・アプリケーションを識別するステップであって、前記ターゲット・アプリケーションが、前記高速初期化イメージとはレイアウトが異なる１つ以上のモジュールを含む、格納済みフォーマットを有する、ステップと、
前記ターゲット・アプリケーションに関連する１つ以上の静的および動的リンク・モジュールを判定し、前記ターゲット・アプリケーションによって使用されるデーターの内、ＢＰＲＡＭにおける格納に適した変化が少ないデーターを識別するために、前記ターゲット・アプリケーションに対して静的分析を実行するステップと、
前記ターゲット・アプリケーションに関連する追加のモジュールおよびデーターを識別するために、前記ターゲット・アプリケーションを実行し前記ターゲット・アプリケーションの初期化を監視することによって動的分析を実行するステップと、
前記高速初期化イメージを格納するのに適したＢＰＲＡＭデバイスを識別するステップと、
前記ターゲット・アプリケーションの格納済みフォーマットとは異なる、静的および動的な分析の間に識別された前記ターゲット・アプリケーションのメモリー内レイアウトに

基づいて、前記ターゲット・アプリケーションによって使用されるデーターの内、まれにしか変化しないデーターで構成される前記高速初期化イメージを作るステップであって、前記高速初期化イメージが、実行する準備ができており、前記ターゲット・アプリケーションを実行する要求を受ける前に作られる、ステップと、

前記高速初期化イメージを前記ＢＰＲＡＭデバイスにコピーするステップと、

を含み、前記ステップが少なくとも１つのプロセッサーによって実行される、方法。

【請求項２】

請求項１記載の方法において、前記ターゲット・アプリケーションを識別するステップが、前記ターゲット・アプリケーションを識別し、前記ＢＰＲＡＭデバイス内における格納のために前記高速初期化イメージを作るプロセスを開始するために、情報技術（ＩＴ）者によって実行することができる管理ツールを設けるステップを含む、方法。

【請求項３】

請求項１記載の方法において、前記ターゲット・アプリケーションを識別するステップが、マニフェストを、前記ターゲット・アプリケーションに関連するインストール・パッケージと共に、提供するステップを含む、方法。

【請求項４】

請求項１記載の方法において、静的分析を実行するステップが、前記ターゲット・アプリケーションの機能を拡張する１つ以上のアドイン・モジュールを調べるステップを含む、方法。

【請求項５】

請求項１記載の方法において、静的および動的分析を実行するステップが、前記ターゲット・アプリケーションに関連するランタイムを調べ、前記ランタイムに関連するコードおよびデーターを前記ＢＰＲＡＭデバイスに入れるステップを含む、方法。

【請求項６】

請求項１記載の方法において、動的分析を実行するステップが、前記ターゲット・アプリケーションがどのように初期化するかについての情報を収集するステップを含み、前記情報が、前記高速初期化イメージを生成するために使用することができ、前記ターゲット・アプリケーションが、当該ターゲット・アプリケーションの典型的な初期化ステップの内少なくとも一部を飛ばして、前記高速初期化イメージから直接実行することができる、方法。

【請求項７】

請求項１記載の方法において、静的および動的分析を実行するステップが、静的および動的な分析の間に識別された前記モジュールの各々を識別するシグネチャー情報を捕らえるステップを含み、前記シグネチャー情報が、前記高速初期化イメージが前記モジュールに関して期限切れであるか否かを示す、方法。

【請求項８】

請求項１記載の方法において、静的および動的分析を実行するステップが、前記ターゲット・アプリケーションが前記高速初期化イメージから初期化されるときに前記初期化を実行することができるように、外部リソースに影響を及ぼす初期化を識別するステップを含む、方法。

【請求項９】

請求項１記載の方法において、前記ＢＰＲＡＭデバイスを識別するステップが、永続的かつバイト・アドレス可能であり、前記高速初期化イメージを格納する十分な空き空間を有するメモリー・デバイスを識別するステップを含む、方法。

【請求項１０】

請求項１記載の方法であって、更に、前記ＢＰＲＡＭデバイスに関する摩耗ステータス情報を判定するステップを含み、前記ＢＰＲＡＭデバイスを識別するステップが、前記判定した摩耗ステータス情報に基づいて、前記ＢＰＲＡＭ内において、前記高速初期化イメージを格納するための位置を選択するステップを含む、方法。

【請求項１１】

請求項１記載の方法において、前記高速初期化イメージを作るステップが、前記ＢＰＲＡＭデバイスに一緒に格納された１つ以上のモジュール・コード・セクションのコンテンツを有する統一レイアウト・イメージを作るステップを含む、方法。

【請求項１２】

請求項１記載の方法において、前記高速初期化イメージをコピーするステップが、前記ターゲット・アプリケーション内において、前記高速初期化イメージから開始するときには不要となる前記ターゲット・アプリケーションの典型的な初期化ステップを超えたポイントを表す実行位置を識別し、前記実行位置に基づいて、前記ターゲット・アプリケーションのエントリー・ポイントを変更するステップを含む、方法。

【請求項１３】

バイト・アドレス可能な永続ランダム・アクセス・メモリー（ＢＰＲＡＭ）を使用するソフトウェア・アプリケーションのレイアウトおよび実行のためのコンピューター・システムであって、

以下のコンポーネント内に具体化されたソフトウェア命令を実行するように構成されたプロセッサーおよびメモリーと、

アプリケーションを初期化するために前記アプリケーションによって使用されるアプリケーション・データーおよびソフトウェア命令を判定し、前記アプリケーションがＢＰＲＡＭから初期化することを可能にする前記アプリケーション・データーおよびソフトウェア命令のレイアウトを決定するアプリケーション・レイアウト・コンポーネントと、

前記ソフトウェアがどのように命令およびデーターにアクセスするか決定し、前記ＢＰＲＡＭにおける格納に適した命令およびデーターを識別するために、前記アプリケーションに関連するソフトウェア・バイナリー、データー・モジュール、または他の格納されたソフトウェア・データーを含むソフトウェアを静的に分析する静的分析コンポーネントと、

実行中に前記アプリケーションを動的に分析し、前記アプリケーションの挙動に関する追加情報を収集する動的分析コンポーネントと、

前記静的および動的な分析および前記アプリケーション・レイアウト・コンポーネントによって決定されたレイアウトに基づいて、前記アプリケーションによって使用されるアプリケーション・データーの内、まれにしか変化しないデーターで構成される、前記アプリケーションを初期化するためのイメージを前記ＢＰＲＡＭ内に作り、前記作ったイメージを前記ＢＰＲＡＭに格納するレイアウト作成コンポーネントと、

前記レイアウト作成コンポーネントによって作られたＢＰＲＡＭにおける前記イメージを使用して、要求時に前記アプリケーションを初期化するアプリケーション初期化コンポーネントと、

を含む、システム。

【請求項１４】

請求項１３記載のシステムにおいて、前記アプリケーション・レイアウト・コンポーネントが前記アプリケーションをハイバネートし、再開イベント時に素早く前記アプリケーションを復元することを可能にする、システム。

【請求項１５】

請求項１３記載のシステムにおいて、前記アプリケーション・レイアウト・コンポーネントが、ＢＰＲＡＭに格納するために、前記アプリケーションの頻繁に変化しないデーターを識別する、システム。

【発明の詳細な説明】

【背景技術】

【０００１】

[0001] コンピューター・システムは、起動されるときに、ブート・フェーズを遂行して、このコンピューターを使用のために準備し、ハードウェアの制御をオペレーティング・システムに引き渡す。一般に、ハードウェアは、基本入力／出力システム（ＢＩＯＳ）および拡張ファームウェア・インターフェース（ＥＦＩ）のような、ハードウェアを初期

化するファームウェアを含む。次いで、このファームウェアは通常ディスク・ドライブに注意を向けて、マスター・ブート・レコード（ＭＢＲ）または他の実行すべき最初の１組の命令を識別する。これらの命令は、ブート・ローダーを呼び出すことができ、そしてブート・ローダーはオペレーティング・システムをロードし初期化する（または、コンピューター・システムをブートするためのオペレーティング・システムのメニューを提供することも可能である）。同様に、一旦オペレーティング・システムがロードされると、アプリケーションも初期化フェーズを遂行し、その間に、このアプリケーションに関連するソフトウェアおよびデーター・モジュールがロードされ、メモリー状態が初期化される（例えば、リソースまたは他のデーターによって）等が行われる。

【０００２】

[0002] これらのブートおよび初期化プロセスは、時間がかかることが多い。精巧なハードウェアが初期化して動作状態に到達するのに時間がかかる可能性があるだけでなく、ソフトウェアをディスクからロードし、種々の初期化タスクを行うために実行するにも時間がかかる。コンピューター・システムのユーザーは、このプロセスが完了するまで、システムもアプリケーションも使用することができず、つまり時間が浪費される。加えて、スリープ状態からコンピューターを再起動(waking)するというような他の期間、低電力状態、即ち、ハイバーネーション(hibernation)の間、システムは同様のブート状プロセスを実行するかもしれず、これがユーザーのコンピューター・システムを遅らせる。移動体デバイス、ゲーム・コンソール、ルーター、およびコンピューター・システムとして内部に実装される他のデバイスでは、消費者がそれらを統合消費者用電子機器と見なしても、ブートするための時間がデバイスの可用性を著しく落とす可能性がある。

【発明の概要】

【発明が解決しようとする課題】

【０００３】

[0003] 標準的なオペレーティング・システムでは、カーネルのような、まれにしか更新されない一定コードが、ユーザー・アプリケーション・コードと同じように、レガシー・ドライブ(legacy drive)から読み出される。これが、最適未満のブート時間となる。何故なら、オペレーティング・システムは、ハード・ドライブまたは他の比較的遅いリソースから複雑なロード・シーケンスを介して、それ自体を動作状態にするからである。最新のフラッシュ・ベース・ディスク・ドライブを使用しても、ソフトウェア・コードをロードし初期構成を設定するための書き込みおよび読み出しの回数は、ユーザーにとって著しい時間量を消費する可能性がある。これは、アプリケーションにも当てはまる。アプリケーションでは、数十ものライブラリーにアクセスしてロードし、続いてライブラリーによって指定されたメモリー・エリアをデーターで初期化することを伴う場合がある。このデーターの大部分は、一定データーであり、オペレーティング・システムまたはアプリケーションがロードする度に同一である。過去の技法には、このデーターを既存の、非バイト・アドレス可能なハード・ドライブに再配列する方法を目的とするものがあるが、それでも著しい遅延を伴う。また、過去の技法には、ユーザーによってアプリケーションが要求される前に通常の初期化シーケンスを早期に実行することを目的とするものもある（例えば、MICROSOFT TM WINDOWS（登録商標） TM Prefetch and SuperFetch featuresを参照のこと）。

【課題を解決するための手段】

【０００４】

[0004] 本明細書では、一定データーおよび実行可能コードを不揮発性高性能メモリー（例えば、ＲｅＲＡＭまたはＰＣＭ）に移動させつつ、変化するデーターをＤＲＡＭのような揮発性ストレージに残すことによって、コンピューター・システムのブート時間および／またはアプリケーションの初期化時間を短縮する(speed up)ソフトウェア・レイアウト・システムについて記載する。このシステムは、オペレーティング・システムまたはアプリケーションのどのコンポーネントおよび形態(aspect)が一定であり、全く変化がないまたは最小限の変化しかないか判定する。この情報から、このシステムは、カーネルを含

む、これら頻繁に使用されるコンポーネントに対してより速いアクセスを与えるために、高性能メモリーＲｅＲＡＭ／ＰＣＭキャッシュを作る。その結果、カーネルまたはアプリケーションのコードおよびデーター構造は、メモリー・フェッチに関して、高速性能アクセスおよび実行を有することになる。このシステムは、アプリケーションに関しても同様のプロセスを実行することができる。ソフトウェア・アプリケーションも共通の起動シーケンスを遂行して、プロセス・データー構造、スレッド、リソース等を初期化する。これらは、本明細書において説明する技法を使用して、予測し高速化することができる。場合によっては、高性能キャッシュが直接バイト・アドレス可能である場合のように、このシステムはキャッシュから直接ブートするまたは実行することさえも可能である。つまり、このソフトウェア・レイアウト・システムは、通常動作のためにオペレーティング・システムおよびアプリケーションを準備するためのより速い方法を提供し、初期化に費やされる時間を短縮する。

【０００５】

[0005] この摘要は、詳細な説明において以下で更に説明する概念から選択したものを、簡略化した形態で紹介するために設けられている。この摘要は、特許請求する主題の主要な特徴や必須の特徴を特定することを意図するのではなく、特許請求する主題の範囲を限定するために使用されることを意図するのでもない。

【図面の簡単な説明】

【０００６】

【図１】図１は、一実施形態におけるソフトウェア・レイアウト・システムのコンポーネントを示すブロック図である。

【図２】図２は、一実施形態において、オペレーティング・システム・カーネルを検査してＢＰＲＡＭにおける事前レイアウトのためにコードおよびデーターを識別するためのソフトウェア・レイアウト・システムの処理を示す流れ図である。

【図３】図３は、一実施形態において、ＢＰＲＡＭからオペレーティング・システム・カーネルを初期化するためのソフトウェア・レイアウト・システムの処理を示す流れ図である。

【図４】図４は、一実施形態において、アプリケーションを検査してＢＰＲＡＭにおける事前レイアウトのためにコードおよびデーターを識別するためのソフトウェア・レイアウト・システムの処理を示す流れ図である。

【図５】図５は、一実施形態において、ＢＰＲＡＭからアプリケーションを初期化するためにソフトウェア・レイアウト・システムの処理を示す流れ図である。

【図６】図６は、一実施形態において、ソフトウェア・レイアウト・システムによって使用される種々のタイプのストレージおよびデーターの関係を示すブロック図である。

【発明を実施するための形態】

【０００７】

[0012] 本明細書では、一定データーおよび実行可能コードを不揮発性高性能メモリー（例えば、ＲｅＲＡＭまたはＰＣＭ）に移動させつつ、変化するデーターをＤＲＡＭのような揮発性ストレージに残すことによって、コンピューター・システムのブート時間および／またはアプリケーションの初期化時間を短縮する(speed up)ソフトウェア・レイアウト・システムについて記載する。このシステムは、オペレーティング・システムまたはアプリケーションのどのコンポーネントおよび形態(aspect)が一定であり、全く変化がないまたは最小限の変化しかないか判定する。この情報から、このシステムは、これら頻繁に使用されるコンポーネントに対してより速いアクセスを与えるために、高性能メモリーＲｅＲＡＭ／ＰＣＭキャッシュを作る。頻繁に使用されるコンポーネントにはカーネルが含まれる。例えば、このシステムはカーネル・データー構造およびコードを不揮発性で高性能のＲｅＲＡＭに保持する一方、動的データー／変化するデーターを揮発性ＤＲＡＭメモリーに残すことができる。その結果、カーネルまたはアプリケーションのコードおよびデーター構造は、メモリー・フェッチに関して、高速性能アクセスおよび実行時間を有する

ことになる。

【０００８】

[0013] このシステムは、アプリケーションに関しても同様のプロセスを実行することができる。ソフトウェア・アプリケーションも共通の起動シーケンスを遂行して、プロセス・データー構造、スレッド、リソース等を初期化する。これらは、本明細書において説明する技法を使用して、予測し高速化することができる。例えば、このシステムは、殆ど変化しないデーターを参照するアプリケーションの起動シーケンスの部分を特定することができ、そのデーターを不揮発性の高性能メモリー・キャッシュにキャッシュすることができ、アプリケーションが起動する毎に、このデーターにアクセスすることができる。実施形態では、このシステムはカーネルまたはプロセスをアクティブにするために、メモリーへの高速転送を可能にする仕方でデーターを配列する(lay out)。例えば、このシステムはメモリーにおけるオペレーティング・システムまたはアプリケーションの最終レイアウトを動作中に決定し、そのレイアウトのイメージを高性能キャッシュに格納することができ、次いでオペレーティング・システムまたはアプリケーションが次に起動することを選択されたときに、このシステムは直接メモリー・アクセス（ＤＭＡ）または他の高性能コピー技法を使用して、事前レイアウト・ブート／初期化イメージを、実行のためにメモリーに移すことができる。場合によっては、高性能キャッシュが直接バイト・アドレス可能である場合のように、このシステムはキャッシュから直接ブートするまたは実行することさえも可能である。つまり、このソフトウェア・レイアウト・システムは、通常動作のためにオペレーティング・システムおよびアプリケーションを準備するためのより速い方法を提供し、初期化に費やされる時間を短縮する。

【０００９】

[0014] 本明細書において説明するように、ソフトウェア・レイアウト・システムは２つの概念的フェーズ、レイアウトおよび起動において動作する。レイアウト・フェーズの間、このシステムは１つ以上のオペレーティング・システムまたはアプリケーションを分析して、オペレーティング・システムまたはアプリケーションがどのようにそれ自体を初期化するか判定する。このシステムは、当技術分野では周知である種々の静的および動的分析技法を使用して、二進実行可能ファイル、メモリー内データー、生の実行(live execution)、または他の情報を分析して、起動中にオペレーティング・システムまたはアプリケーションによって使用される命令およびデーターを識別する。例えば、 MICROSOFT TM WINDOWS（登録商標） TMオペレーティング・システムによって使用される移植可能実行可能フォーマット（ＰＥ）は、読み取り専用または読み取り／書き込みというフラグを立てることができるデーターのセクションを含み、ソフトウェア・レイアウト・システムはこの情報を使用して、読み取り専用セクションをＲｅＲＡＭまたは他のキャッシュに入れることができる。また、このシステムは、過去の変更情報を追跡して、どの命令またはデーターが希にしか変化しないかについてピクチャー(picture)を作ることもできる。次いで、このまれにしか変化しないデーターが、ＲｅＲＡＭのような、もっと速いストレージからのロードのために目標にされる。このシステムは、分析の結果を記述するレイアウト・イメージまたは他のデーター構造を作り、より速いメモリーを使用する格納のために、ソフトウェア命令および／またはデーターを供給する。

【００１０】

[0015] 起動フェーズの間、このシステムは、以前に実行したレイアウト作業を利用して、オペレーティング・システムまたはアプリケーションの起動を一層速くする。これは、ＢＩＯＳまたは他の起動命令に、ＲｅＲＡＭキャッシュから初期化イメージに直接アクセスするように指令することを含むとよく、あるいはＤＭＡまたは他の高速転送方法を使用して、従前の実行のために、実行準備完了イメージをキャッシュから揮発性メモリーにコピーする命令を含むのでもよい。このような転送は、数十個のイメージをひとつずつロードするよりも速いので、著しい初期化プロセスの高速化となり、更にユーザーによる使用のためにアプリケーションの準備ができるまでの時間短縮となる。

【００１１】

[0016] 最近のオペレーティング・システムではいずれも、データーはミュータブルまたはインミュータブルのいずれかである。アプリケーションのヒープ内に含まれるデーターのようなデーターはミュータブルであることが多い。これは、アプリケーションの実行中にデーターが変化するからである。実行可能イメージ（ソフトウェア・コード）およびグローバル定数は、インミュータブル・データーの例である。現行では、ミュータブル・データーおよびインミュータブル・データーは双方共、フラッシュまたはハード・ドライブのような、永続性ストレージに保持され、実行中にアプリケーションにアクセス可能にするために、ＤＲＡＭにキャッシュされる。ハード・ドライブおよびフラッシュ・メモリーは、一般にバイト・アドレス可能であるので、実行は通例このようなデバイスから直接行うことはできない。これが、このようなデーターが最初にＤＲＡＭにコピーされる理由である。ＤＲＡＭはバイト・アドレス可能であり、ソフトウェア・コードを実行することによって、実行しアクセスすることができる。

【００１２】

[0017] 一般に、現行のメモリー階層は、遅く、永続的な一括アクセス技術と、高速、バイト・アドレス可能、非永続的技術との間で差がある。相変化メモリー、ＲｅＲＡＭ、またはメムリスター(memristor)のようなバイト・アクセス可能な永続メモリー技術は、永続的なだけでなくバイト・アドレス可能であるストレージを提供することによって、ＤＲＡＭと永続ストレージとの間のギャップに橋渡しする。本明細書では、これらの技術をＢＰＲＡＭ（バイト・アドレス可能、永続的、ランダム・アクセス・メモリー）と呼ぶ。これらの異なるメモリー技術は全て、様々な性能および永続性（摩耗）保証を有するが、これらは、最近のコンピューターにおいて使用することができる新しいタイプのメモリーを導入する。インミュータブル・オブジェクトを格納するためにＢＰＲＡＭを使用することによって、性能を向上させ、ＤＲＡＭを解放し、電力を節約することができ、ソフトウェア・レイアウト・システムは、アプリケーションまたはオペレーティング・システム内において、これらの利点を得るためにＢＰＲＡＭに移動させることができるデーター構造を識別する。

【００１３】

[0018] このＢＰＲＡＭの使用によって導入される１つの新たな考慮事項は、摩耗(wear)である。摩滅する(wearing out)までのＢＰＲＡＭへの書き込み回数は、ＤＲＡＭまたはハード・ディスクよりも遙かに少ないので、この技術摩滅によってデーターが失われないことを確保するためには、注意が必要となる場合が多い。したがって、最も簡単なＢＰＲＡＭの使用は、最初にＤＲＡＭと並行してメモリー・バスにおいてこれを利用可能にすることである。次に、この利用可能な永続メモリーにアプリケーション実行可能ファイルのみを入れることを考えることができる。より遅い大容量ストレージ(bulk storage)からアプリケーション・バイナリーをフェッチしてバイナリーを揮発性メモリーにキャッシュする代わりに、単に実行可能ファイルをＢＰＲＡＭに格納しＢＰＲＡＭから直接実行することができる。実行可能ファイルはアプリケーション・データーよりも更新される頻度が遙かに少ないので、ＢＰＲＡＭが長い寿命を有することを確信することができる。心配があるとすれば、実行が遅くなることであろう。何故なら、ＢＰＲＡＭはＤＲＡＭよりも遅いことが多いからであるが、ＣＰＵがプリフェッチおよびキャッシュすることによって、この相違の多くを隠すことができ、ＢＰＲＡＭはハード・ドライブよりは遙かに高速化することができる。ＢＰＲＡＭにキャッシュされた実行可能ファイルは、アプリケーション、ライブラリー、動的にリンクされるライブラリー（ＤＬＬ）、またはオペレーティング・システムを構成するコンポーネントを含む、任意のタイプのバイナリーを含むことができる。

【００１４】

[0019] 単に実行可能ファイルをキャッシュすることを超えて、計算環境の他の部分もＢＰＲＡＭにキャッシュすることができる。例えば、コードがコンパイルされるとき、「定数」または「読み取り専用」として定められたデーター（エクゼキュータブル・フォーマットにおいて印される、または他の方法で判定される）は、ＤＲＡＭに保持するのでは

なく、ＢＰＲＡＭに入れることができる。大量の静的データーを計算のために使用するアプリケーションは、ＤＲＡＭを解放し、これらのデーター構造を代わりにＢＰＲＡＭに入れることができる。

【００１５】

[0020] 最後に、ランタイム(run-time)は、そのアクセス・パターンに依存して、データー構造をＢＰＲＡＭ内にまたはＢＰＲＡＭから移動させるように設計することができる。例えば、ハッシュ・テーブルは、実際に状態機械として表されてもよく、この場合頻繁に更新され、次いでリーダーによってのみアクセスされ、次いで再度更新される。これらのタイプのアクセス・パターンは、データー構造をＢＰＲＡＭに入れることができつつも、リーダーによってアクセスされることを意味することもあり得る。しかしながら、アプリケーションがデーター構造に書き込むことによってデーター構造を更新しようとするとき、データー構造は揮発性（ＤＲＡＭ）メモリーに戻される。

【００１６】

[0021] アプリケーションは、いくつかの異なる方法でこの可用性から恩恵を得る。最初に、プロセスが存在するとき、アプリケーションがデーター構造をディスクにシリアル化する必要性を排除する。勿論、これが有用なのは、アプリケーションがデーター構造の一貫性について推論できる場合だけである。アプリケーションがクラッシュしたとき、データー構造はもはや有用でなく、他の位置に保持されているコピー（またはスナップショット）から作り直さなければならないかもしれない。第２に、アプリケーションがそのデーター構造の永続バージョンに直ちにアクセスできれば、アプリケーションの性能を向上させることができる。最後に、ＤＲＡＭよりも低いエネルギ・フットプリントを有するＢＰＲＡＭでは、ＤＲＡＭの有意な部分をＢＰＲＡＭと交換して、ＢＰＲＡＭにそのデーターの一部を保持することによって、計算機(computing machine)の全体的なエネルギ・フットプリントを下げることができる。

【００１７】

[0022] 標準的なオペレーティング・システムでは、カーネルのように、希にしか更新されない一定コードは、ユーザー・アプリケーション・コードと同じ方法でレガシー・ドライブから読み出される。ソフトウェア・レイアウト・システムは、オペレーティング・システムのどのコンポーネントが一定であり、変化を受けないかまたは最少の変化をうけるのか判定し、これらの頻繁に使用される、カーネルを含むコンポーネントに対して速いアクセスを与えるために、高性能ＲｅＲＡＭ／ＰＣＭキャッシュを作る。このように、このシステムはカーネル・データー構造およびコードを不揮発性で高性能のＲｅＲＡＭに保持しつつ、変化するデーターは揮発性ＤＲＡＭまたは他のシステムに残す。

【００１８】

[0023] ユーザー空間アプリケーションは、従前より、オペレーティング・システムが初期化し安定した後に、新しくロードされる。しかしながら、アプリケーションを後にロードするためには、他のオペレーティング・システムがアプリケーションをライブラリーから漸次ロードし、次いでユーザー・データーをメモリーにロードして、完成されたライブ・イメージを形成する必要がある。ソフトウェア・レイアウト・システムは、後に同じまたは他のシステムにおいて高速転送および高速再構成(reconstitution)を可能にするために、迅速に、アプリケーションおよびそのメモリーをＲｅＲＡＭ／ＰＣＭに存続させる。この保留および再開は、オペレーティング・システムのロード・シーケンスを使用してアプリケーション・イメージを形成するコストおよび手間をなくし、ユーザー・データーをライブ状態で永続させる。本システムのこれらおよびその他のエレメントについて、以下で更に詳しく説明する。

【００１９】

[0024] 図１は、一実施形態における、ソフトウェア・レイアウト・システムのコンポーネントを示すブロック図である。システム１００は、カーネル・レイアウト・コンポーネント１１０、アプリケーション・レイアウト・コンポーネント１２０、静的分析コンポーネント１３０、動的分析コンポーネント１４０、レイアウト作成コンポーネント１５０

、摩耗監視コンポーネント１６０、カーネル初期化コンポーネント１７０、およびアプリケーション初期化コンポーネント１８０を含む。これらのコンポーネントの各々について、ここで更に詳細に説明する。

【００２０】

[0025] カーネル・レイアウト・コンポーネント１１０は、ブート・アップ(boot up)するためにオペレーティング・システムによって使用されるデーターおよびソフトウェア命令を決定し、カーネルがＢＰＲＡＭからブートすることを可能にするバイト・アドレス可能永続ランダム・アクセス・メモリー（ＢＰＲＡＭ）のために、カーネル・データーおよびソフトウェア命令のレイアウトを決定する。カーネルは、満足できる性能を可能にするだけ十分速いバイト・アドレス可能メモリーの場合、ＢＰＲＡＭから直接ブートすることができ、またはＢＰＲＡＭに格納されているレイアウトを最初に他のメモリー（例えば、ＤＲＡＭ）にコピーすることによって、間接的にＢＰＲＡＭからブートすることもできる。オペレーティング・システムの中には、コードをどこから実行するかの判断に影響を及ぼす、セキュア・ブート・プロセスのような、追加の要件を有するものもある。カーネル・レイアウト・コンポーネント１１０は、静的分析コンポーネント１３０、動的分析コンポーネント１４０、およびレイアウト作成コンポーネント１５０を呼び出し、カーネルのレイアウト・イメージを決定し、決定したイメージをＢＰＲＡＭにコピーする。また、カーネル・レイアウト・コンポーネント１１０は、コンピューター・システムに、ＢＰＲＡＭにコピーされたイメージからブートさせるために、ブート・データー構造または他の情報を変更させることもできる。また、カーネル・レイアウト・コンポーネント１１０は、再始動、ハイバーネーション、スリープからの再開等のような、カーネルの他のタイプの初期化に対してレイアウトを決定することもできる。

【００２１】

[0026] アプリケーション・レイアウト・コンポーネント１２０は、アプリケーションを初期化するためにこのアプリケーションによって使用されるデーターおよびソフトウェア命令を決定し、このアプリケーションをＢＰＲＡＭから初期化することを可能にするアプリケーション・データーおよびソフトウェア命令のＢＰＲＡＭに対するレイアウトを決定する。カーネルと同様、アプリケーションはＢＰＲＡＭから直接実行することもでき、またはオペレーティング・システムが、ＢＰＲＡＭに格納されているアプリケーション・レイアウトを他のメモリー・デバイスに最初にコピーし、そこからアプリケーションを実行することもできる。また、現在の移動体デバイスにおいて非バイト・アドレス可能なフラッシュ・メモリーを使用して行われることがあるように、アプリケーション状態をハイバネートし急速にアプリケーション状態を復元するために、レイアウトを使用することもできる。アプリケーションは、変化するデーターおよび比較的変化しないデーターの組み合わせで実行するかもしれず、アプリケーション・レイアウト・コンポーネント１２０は、比較的変化しないデーターをＢＰＲＡＭに入れ、一方変化するデーターを、ＤＲＡＭのような、揮発性メモリーに入れることを決定することができる。アプリケーション・バイナリー、定数、および他のタイプのデーターというようなデーター項目は、アプリケーションが実行する度に同じである場合があり、あるいは１年に数回以下だけ更新するかもしれないアプリケーション・バイナリーのように、まれにしか変化しない場合もある。この情報をＢＰＲＡＭに直接実行可能なレイアウトで保持することによって、アプリケーションは、現在ロード・プロセスが通例辿るよりも遙かに速く実行状態に達することが可能になる。このロード・プロセスの間、命令およびデーターがある数十個のバイナリーが別個の１つの記憶位置（例えば、ハード・ドライブ）からロードされ、アプリケーションが実行する準備ができる前に、メモリーにマッピングされればよい。

【００２２】

[0027] アプリケーション・レイアウト・コンポーネント１２０は、静的分析コンポーネント１３０、動的分析コンポーネント１４０、およびレイアウト作成コンポーネント１５０を呼び出し、アプリケーションのレイアウト・イメージを決定し、決定したイメージをＢＰＲＡＭにコピーする。また、アプリケーション・レイアウト・コンポーネント１２

０は、オペレーティング・システムに、アプリケーションを実行する要求を、ＢＰＲＡＭにコピーしたイメージにリディレクトさせるために、オペレーティング・システムのデーター構造または他の情報を変更することもできる。最終的に実行するコンピューター・システム以外のコンピューター・システムにおいてイメージを作ることができ、次いでそのイメージを実行する１つ以上のコンピューター・システムにコピーすることができるように、カーネル・レイアウト・コンポーネント１１０およびアプリケーション・レイアウト・コンポーネント１２０によって作られたイメージは、他のコンピューター・システムおよび／またはオペレーティング・システムに移植可能であるとよい。これによって、仮想環境、アプリケーション・ホスト、移動体デバイス製造業者、または他の者が、高速ロード・アプリケーションおよびオペレーティング・システムのイメージを生成することが可能になり、このイメージは、任意の特定のデバイスのＢＰＲＡＭにコピーし、本明細書において説明する利点を実現することができる。

【００２３】

[0028] 静的分析コンポーネント１３０は、ソフトウェア・バイナリー、データー・モジュール、または他の格納されているソフトウェア・データーを静的に分析して、どのようにソフトウェアが命令およびデーターにアクセスするか判定する。コンポーネント１３０は、バイナリー・コード（例えば、アセンブリー）、中間コード（例えば、 MICROSOFT TM中間言語（ＩＬ）コード）、あるいはアプリケーションまたはオペレーティング・システムの他のコンパイル・バージョンまたは実行可能(runnable)バージョンを分析することができる。静的分析は、過去数年の間に著しく進歩した。システム１００は、静的分析技法を使用して、アプリケーションがコード、データー、および他のリソースを格納するエリアに具体的に的を絞る。静的分析コンポーネント１３０は、アプリケーション・バイナリーをインストルメント化して、情報を受信することまたはアプリケーションの特定のアクションを傍受することができ、傍受したアクションを新たなアクションまたは追加のアクションと交換することができる。例えば、コンポーネント１３０が外部モジュールへのコールを発見した場合、コンポーネント１３０はこのコールを、ＢＰＲＡＭにおける外部モジュールのイメージにアクセスするためのコールと交換することができる。このように、システム１００は、アプリケーションの動作を発見し、アプリケーションおよびオペレーティング・システムの初期化を高速化するためのＢＰＲＡＭの使用を可能にするように、挙動を変更することができる。また、コンポーネント１３０は、特定のコードおよび／またはデーターが、分析対象のソフトウェアによってどのように使用されるかを記述するソフトウェア・データーにおける移植可能な実行可能（ＰＥ）ヘッダまたは他の情報というような、メタデーターにアクセスすることもできる。例えば、ＰＥフォーマットは、特定のセクションに、実行可能コード、読み取り専用データー、読み取り／書き込みデーター、リソース（例えば、ボタン、アイコン、ダイアログ・ボックス等）等の印を付けることを可能にする。

【００２４】

[0029] 動的分析コンポーネント１４０は、実行中のアプリケーションを動的に分析して、静的分析では判定することが難しいアプリケーションの挙動に関する追加の情報を収集する。多くの場合、アプリケーションは静的分析を失敗させるプログラミング・ステップを含む（故意に、または単にコンパイルの後、ステップがそのようになるため）。動的分析は、外部コンポーネントから受信した応答の内容、アプリケーションによって使用されたメモリーの実際の内容、およびユーザー構成情報というような、静的分析の間では推論または近似しか得られない情報を入手可能にする。場合によっては、本システムが、性能オーバーヘッドまたは他の分かっているリソース制約に基づいて、静的分析、動的分析、または両方の内どれを実行すべきか決定する。動的分析は、潜在的に、静的分析では発見されないアプリケーション挙動を発見することができる。また、動的分析コンポーネント１４０は、動的分析を使用して静的分析の結果を確認することもできる。動的分析コンポーネント１４０は、判定した情報をレイアウト作成コンポーネント１５０に提供して、ＢＰＲＡＭにおけるレイアウトに適した特定のアプリケーション・コードおよびデーター

を識別する。例えば、動的レイアウト・コンポーネント１５０は、特定のデーターが読み取り／書き込みの印が付けられていても、実際にはアプリケーションによって変化させられないと判定することもある。逆に、動的分析は、アプリケーションのデーターの内、頻繁に変化させられる部分を識別することもできる。また、アプリケーションが、アプリケーション・コードが読み取り専用でありしたがってＢＰＲＡＭにおける格納に適しているか否かを示す自己変更コードのような挙動を含む場合もある。

【００２５】

[0030] レイアウト作成コンポーネント１５０は、カーネル・レイアウト・コンポーネントまたはアプリケーション・レイアウト・コンポーネントによって決定されたレイアウトに基づいて、オペレーティング・システムまたはアプリケーションを初期化するためのイメージをＢＰＲＡＭ内に作り、この作ったイメージをＢＰＲＡＭに格納する。レイアウト作成コンポーネント１５０は、１つ以上の格納されているモジュール、データー・ストア等からデーター、バイナリー・コード、および他のソフトウェア情報にアクセスして、アプリケーションまたはオペレーティング・システムのメモリー内データーの内、オペレーティング・システムまたはアプリケーションを実行するために使用される少なくとも一部を表すイメージを作ることができる。レイアウト作成コンポーネント１５０は、オペレーティング・システムまたはアプリケーションを実行する前に、ローダー、ヒープ、および現在の他のソフトウェア・コンポーネントによって実行されるステップの多くを実行する。例えば、ローダーは、アプリケーションがロードする毎に、通例ハード・ドライブに格納されているバイナリー・モジュールを引き出し、これらのモジュールの内容をバイト・アドレス可能メモリー・ヒープに入れる役割を果たすが、レイアウト作成コンポーネント１５０は同様のステップを１回実行し、結果的に得られたメモリー・イメージをＢＰＲＡＭに入れる。ＢＰＲＡＭでは、アプリケーションが要求される毎に、もっと遅いストレージからモジュールを都度ロードすることなく、準備しておくことができる。これは、アプリケーション・ロード時間を大幅に短縮する。

【００２６】

[0031] 摩耗監視コンポーネント１６０は、任意に、ＢＰＲＡＭに対する摩耗指定(wear specification)を超える回数のＢＰＲＡＭへの書き込みを避けるために、ＢＰＲＡＭの使用情報を監視する。ＢＰＲＡＭは、多くの場合、容認できる書き込み回数に限度があり、したがって、ＢＰＲＡＭが均一に使用されることを保証するために、摩耗平準化および同様の技法を実行することは役に立つ（例えば、１つのセクションは毎回使用されないが、他のセクションは未使用のままである）。加えて、コンポーネント１６０は静的および／または動的分析によって判定されたデーターの書き込み頻度に基づいて、どのデーターをどのデバイスに格納すべきかについてトレードオフを行うこともできる。特定のＢＰＲＡＭデバイスが特定のデーターの書き込み負荷をかなり長い寿命（例えば、デバイスが特定のコンピューター・システムにおいて使用され得る５～１０年）にわたって処理することができる場合、システム１００はデーターをその位置に格納すればよい。寿命がもっと長いＢＰＲＡＭデバイス（例えば、書き込みサイクルもっと多い）では、システム１００は、頻繁に書き込まれないデーターを、そのデーターが完全に読み取り専用でなくても、ＢＰＲＡＭに含めることを選択してもよい。例えば、アプリケーション構成パラメーターは希にしか変化しないので、ＢＰＲＡＭが適している。ＢＰＲＡＭは時の経過と共に、増々寿命が長くなるように開発されたので、本システムはＢＰＲＡＭをもっと多くのデーターに使用するように構成することができる。

【００２７】

[0032] カーネル初期化コンポーネント１７０は、レイアウト作成コンポーネントによって作られたＢＰＲＡＭにおけるイメージを使用して、オペレーティング・システムを初期化する。回転ハード・ドライブまたはソリッド・ステート・ディスクのように、オペレーティング・システムのバイナリーおよびデーターをセクター・ベースの永続ストレージからロードすることによる従前の仕方でオペレーティング・システムをブートする代わりに、コンポーネント１７０はオペレーティング・システムをＢＰＲＡＭイメージから直接

実行するか、またはＢＰＲＡＭイメージをＤＲＡＭまたは他のメモリーに素早くコピーすることによって実行する。レイアウト作成コンポーネント１５０によって作成されたレイアウトは、コードおよびデーターが本来どこに格納されていたかに基づくのではなく、実行中にコードおよびデーターがどのように使用されるかに基づくので、格納されたイメージからオペレーティング・システムを初期化する方が遙かに速い。数十回になる可能性がある、より遅いストレージへのコールの代わりに、カーネル初期化コンポーネント１７０は、バイト・アクセス可能であることを利用してオペレーティング・システムをＢＰＲＡＭから直接実行することができ、または高速直接メモリー・アクセス（ＤＭＡ）転送を使用して、ＢＰＲＡＭイメージをＤＲＡＭまたは他のメモリーに素早くコピーすることができる。

【００２８】

[0033] アプリケーション初期化コンポーネント１８０は、レイアウト作成コンポーネントによって作られたＢＰＲＡＭにおけるイメージを使用して、アプリケーションを初期化する。アプリケーションのバイナリーおよびデーターを、回転ハード・ドライブまたはソリッド・ステート・ディスクのようなセクター・ベースの永続ストレージからロードすることによる従前の仕方でアプリケーションを初期化する代わりに、コンポーネント１８０はアプリケーションをＢＰＲＡＭイメージから直接ロードするか、またはＢＰＲＡＭイメージをＤＲＡＭまたは他のメモリーに素早くコピーすることによってロードする。レイアウト作成コンポーネント１５０によって作られたレイアウトは、コードおよびデーターが本来どこに格納されていたかに基づくのではなく、実行中にコードおよびデーターがどのように使用されるかに基づくので、格納されたイメージからアプリケーションを初期化する方が遙かに速い。数十回になる可能性がある、より遅いストレージへのコールの代わりに、アプリケーション初期化コンポーネント１８０は、バイト・アクセス可能であることを利用してオペレーティング・システムをＢＰＲＡＭから直接実行することができ、または高速直接メモリー・アクセス（ＤＭＡ）転送を使用して、ＢＰＲＡＭイメージをＤＲＡＭまたは他のメモリーに素早くコピーすることができる。

【００２９】

[0034] ソフトウェア・レイアウト・システムが実装される計算デバイスは、中央処理ユニット、メモリー、入力デバイス（例えば、キーボードおよびポインティング・デバイス）、出力デバイス（例えば、ディスプレイ・デバイス）、および記憶デバイス（例えば、ディスク・ドライブまたは他の不揮発性記憶媒体）を含むことができる。メモリーおよび記憶デバイスは、本システムを実現するまたはイネーブルするコンピューター実行可能命令（例えば、ソフトウェア）をエンコードすることができるコンピューター読み取り可能記憶媒体である。加えて、データー構造およびメッセージ構造もコンピューター読み取り可能記憶媒体に格納することができる。本明細書において特許請求するコンピューター読み取り可能媒体はいずれも、法的に特許可能なカテゴリーに該当する媒体のみを含む。また、本システムは、データーを送信することができる１つ以上の通信リンクも含むことができる。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、二点間ダイアルアップ接続、セル・フォン・ネットワーク等のような、種々の通信リンクを使用することができる。

【００３０】

[0035] 本システムの実施形態は、種々の動作環境において実現することができ、種々の動作環境は、パーソナル・コンピューター、サーバー・コンピューター、ハンドヘルドまたはラップトップ・デバイス、マイクロプロセッサー・システム、マイクロプロセッサー・ベース・システム、プログラマブル消費者用電子機器、ディジタル・カメラ、ネットワークＰＣ、ミニコンピューター、メインフレーム・コンピューター、以上のシステムまたはデバイスの内任意のものを含む分散型計算環境、セット・トップ・ボックス、チップ上システム（ＳＯＣ）等を含む。コンピューター・システムは、セル・フォン、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・コンピューター、プログラマブル消費者用電子機器、ディジタル・カメラ等であってもよい。

【００３１】
[0036] 本システムについて、１つ以上のコンピューターまたは他のデバイスによって実行されるプログラム・モジュールのような、コンピューター実行可能命令という一般的なコンテキストで説明することができる。一般に、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、データー構造等を含み、特定のタスクを実行するか、または特定の抽象データー型を実装する。通例、プログラム・モジュールの機能は、種々の実施形態において所望通りに、組み合わせることまたは分散させることができる。
【００３２】
[0037] 図２は、一実施形態において、オペレーティング・システム・カーネルを検査してＢＰＲＡＭにおける事前レイアウトのためにコードおよびデーターを識別するためのソフトウェア・レイアウト・システムの処理を示す流れ図である。図２および図３は、オペレーティング・システムのカーネルに適用したときの、本システムの動作の２つの異なるフェーズを示す。図２は、高速ブート・レイアウトを決定し、このレイアウトをＢＰＲＡＭにコピーするフェーズを示し、図３は、図２のステップによって既に作られているＢＰＲＡＭイメージからカーネルをブートするフェーズを示す。
【００３３】
[0038] ブロック２１０において開始して、本システムは、カーネルの高速ブート画像を作るために高速レイアウトを実行するターゲット・オペレーティング・システム・カーネルを識別する。ここで、カーネルは、高速ブーティング・イメージとはレイアウトが異なる１つ以上のモジュールを含むフォーマットが格納されている。本システムは、情報技術（ＩＴ）者によって実行することができる管理ツールを提供することができ、またはオペレーティング・システムの一部として自動的に実行してカーネルを識別し、ＢＰＲＡＭにおける格納のために高速ブーティング・イメージを作るプロセスを開始することもできる。カーネルを識別するには、既存のデーター記憶デバイスのブート・レコードまたは他のデーター構造を調べ、コンピューター・システムのブート・アップを監視し、またはターゲット・カーネルを示すモジュール・パスを受けることを含めばよい。
【００３４】
[0039] ブロック２２０に進み、本システムは、識別したターゲット・オペレーティング・システム・カーネルに対して静的分析を実行して、このカーネルに関連する１つ以上の静的および動的にリンクされたモジュールを判定し、このカーネルによって使用されるデーターの内、十分に変化しないものを、バイト・アドレス可能永続ランダム・アクセス・メモリー（ＢＰＲＡＭ）への格納のために識別する。当技術分野には、静的分析を実行するための種々の技法が存在するが、本プロセスが開始すると、一般に、カーネルに関連する主モジュールをロードし、実行可能フォーマットを調べてモジュールのセクション（リソース、定数データー、他のモジュールを参照するリンク・モジュール・セクション、および実行可能コードを含む１つ以上のコード・セクションを含むことができる）を識別し、そして恐らくは、任意の識別された機械コードまたは他の技法を逆アセンブルする(disassembly)。実施形態では、本システムが、動的にリンクされたライブラリー、ハードウェア用ドライバー、またはカーネルによって使用される他のリソースのような、任意の被参照モジュールをロードするとよい場合もある。
【００３５】
[0040] ブロック２３０に進み、本システムは、任意に、オペレーティング・システム・カーネルを実行し、カーネルの初期化を監視して、カーネルに関連する追加のモジュールおよびデーターを識別することによって、動的分析を実行する。動的分析は、静的分析によって発見されたものに加えて、モジュールおよび／またはデーターを発見し、更に静的分析によって識別された項目の確認も行うことができる。動的分析は、静的分析によって判定することが難しい情報を提供することができ、その情報とは、動的に割り当てられカーネルによって使用されるメモリー構造、ある一定の条件の下でのみ動的にロードされるモジュール、特定のハードウェア環境に基づいて異なるカーネルの部分等がある。静的

および動的分析の間、本システムは、カーネルがどのように初期化するかについての情報を収集する。この情報は、事前レイアウト・イメージを生成するために使用することができ、この事前レイアウト・イメージから、カーネルが直接実行することができ、従前の初期化ステップの内少なくとも一部を飛ばすことができる。

【００３６】

[0041] 実施形態の中には、本システムが、発見されたリソースの各々を識別するシグネチャー情報を捕らえるものもある。この識別情報は、ファイル・システムにおけるタイム・スタンプ、チェックサム、または特定のリソースが変化したときを示す他の値を含むことができる。次いで、本システムはこの情報を使用して、オペレーティング・システム・ファイルまたはドライブが、製造業者からの更新によって更新された後というような、レイアウト・分析を再度実行すべきときを判定することができる。本システムは、各ブート・アップの間または特定の回数のブート・アップの後に、レイアウトに寄与した既知の各リソースを実地検証し、いずれかのリソースが変化したか否か判定し、いずれかのリソースが変化していた場合、新たなレイアウト・プロセスを開始してＢＰＲＡＭイメージを更新するプロセスを実行することができる。

【００３７】

[0042] 静的および動的分析の間、本システムはハードウェア・デバイスのような外部リソースに影響を及ぼすあらゆる初期化を観察し、メモを取ることもできる。グラフィクス・カード、入力デバイス、ディスプレイ、および他のデバイスというようなハードウェアは、更なる要求を受けるために、特定の初期化シーケンスが当該ハードウェアを既知の状態に置くことを期待することがある。本ソフトウェア・レイアウト・システムは多数の典型的な初期化ステップを前もって実行することができるが、外部リソースに影響を及ぼすステップは、オペレーティング・システムがブートする度に実行される必要がある場合もある。このため、本システムは、このような初期化ルーチンを追跡し、これらのルーチンに対するポインターまたは他の基準を格納して、オペレーティング・システムが事前レイアウト高速ブート・イメージからロードしているときに、これらを実行できるようにするとよい。

【００３８】

[0043] ブロック２４０に進み、本システムは、オペレーティング・システム・カーネルを高速でブートするための事前レイアウト・イメージを格納するのに適したＢＰＲＡＭデバイスを識別する。ＢＰＲＡＭは、以前から入手可能な典型的なフラッシュ・メモリー、ハード・ドライブ、およびダイナミックＲＡＭ（ＤＲＡＭ）とは、バイト・アドレス可能であるだけでなく永続的であることが異なる。バイト・アドレス可能とは、ハード・ドライブやハード・ドライブと同様に動作するフラッシュでは典型的なように、セクター毎にロードするのではなく、ＢＰＲＡＭの各バイトに直接アドレスできることを意味する。コンピューター・プロセッサーは、通例、バイト・アドレス可能なメモリーに格納されたソフトウェア・コードだけを実行することができる。永続的とは、電力サイクル（即ち、デバイスへの電源を切り、再度入れる）の間、ＢＰＲＡＭがそこに格納されているデーターを保持することを意味する。典型的なＤＲＡＭは、バイト・アドレス可能であるが永続的ではなく、そして典型的なハード・ドライブは、永続的であるがバイト・アドレス可能ではない。つまり、ＢＰＲＡＭは、これらの品質の双方を同時に所有するということにおいて新規である。このため、実行されるべき何かのイメージを、それを実行する要求に先立って形成し、ＢＰＲＡＭに格納し、次いでそこから直接または他のメモリー・デバイスにコピーした後に実行することができる。

【００３９】

[0044] ブロック２５０に進み、本システムは、任意に、ＢＰＲＡＭデバイスに関する摩耗ステータス情報を判定する。これは時の経過と共にデバイスの改良によって解決されるであろうが、現行のＢＰＲＡＭは、限られた回数しか首尾良く書き込むことができないという欠点がある。この回数は、頻繁でないＢＰＲＡＭの更新を可能にするには十分であるが、典型的なＤＲＡＭデバイスまたはハード・ドライブと同じ位の頻度である、ＢＰＲ

ＡＭへの書き込みを可能にするには十分ではない。このため、本システムは、ＢＰＲＡＭデバイスへの書き込みを管理するために摩耗ステータス情報を追跡し、ＢＰＲＡＭデバイスの有効寿命を延ばすために１つ以上の技法を採用することができる。この情報は、デバイスの各領域に何回書き込まれたか、ＢＰＲＡＭのどの部分が現在使用中であるかまたは使用中でないか、各領域に利用可能な予測書き込み回数等を含むとよい。摩耗ステータス情報に基づいて、本システムはレイアウト・イメージを格納するための位置をＢＰＲＡＭ内において選択する。

【００４０】

[0045] ブロック２６０に進み、本システムは、カーネルの格納済みフォーマットとは異なる、カーネルのメモリー内レイアウトに基づいて、高速ブーティング・イメージのためのレイアウトを作る。高速ブーティング・イメージは、実行する準備ができており、カーネルを実行する要求を受ける前に作られる。メモリー内のカーネルの典型的なイメージは、格納されているモジュールをハード・ドライブ・ストレージからＤＲＡＭにロードすることによって作られるが、カーネルを実行する要求のときだけである。本システムは、ここでは、前もってこれらのステップを実行し、カーネルをブートする今後の要求が、これらのステップを毎回繰り返さないことによって、一層速くなるように、その結果をＢＰＲＡＭに入れる。レイアウト・イメージは、１つ以上のモジュール・コード・セクション、１つ以上のデーター・セクション、カーネルによって作られた１つ以上の読み取り専用ヒープ領域等の内容を含むことができる。

【００４１】

[0046] ブロック２７０に進み、本システムは、作ったレイアウト・イメージを、識別したＢＰＲＡＭデバイスにコピーする。ＢＰＲＡＭデバイスは、特定の書き込みプロセスを含むことができ、これによって本システムはイメージをデバイスにコピーする。また、本システムは、カーネル内において、カーネルの典型的な初期化ステップの全てを超えたポイントを表す特定の実行位置を識別することもできる。レイアウト・イメージは、ある量の初期化が実行された後の状態におけるカーネルを含むので、カーネルがＢＰＲＡＭから実行されるとき、通常よりも遅い実行位置において開始することができる。このため、本システムはカーネル・イメージ内においてエントリー・ポイント・アドレスを上書きする、またはカーネルをＢＰＲＡＭから実行するときにしかるべきポイントから開始するようにカーネルに命令する同様の変更を行い、これによって典型的な初期化ステップを飛ばすことができる。ブロック２７０の後、これらのステップは終了する。

【００４２】

[0047] 図３は、一実施形態において、ＢＰＲＡＭからオペレーティング・システム・カーネルを初期化するためのソフトウェア・レイアウト・システムの処理を示す流れ図である。先に注記したように、図３は、図２のステップにおいて既に作られたＢＰＲＡＭイメージからカーネルをブートするフェーズを示す。

【００４３】

[0048] ブロック３１０において開始し、本システムはオペレーティング・システムをブートする要求を受ける。通例オペレーティング・システムの制御下で初期化するアプリケーションとは異なり、オペレーティング・システムは、通例、コンピューター・システムの基本入力／出力システム（ＢＩＯＳ）、ブート・ローダー、ハイパーバイザー（仮想化の場合）、または他の低位ブート・コードの制御下で初期化する。低位ブート・コードは、マスター・ブート・レコード（ＭＢＲ）、グローバル一意識別子（ＧＵＩＤ）パーティション・テーブル（ＧＰＴ）、またはブート可能なオペレーティング・システムで入手できるものを示す他の格納されたデーター構造から、オペレーティング・システムを識別することができる。一旦特定のオペレーティング・システムが識別されたなら、低位ブート・コードは、通例、このオペレーティング・システムの主要モジュールを識別し、このモジュールをロードし、このモジュールのエントリー・ポイントに制御を渡す。ソフトウェア・レイアウト・システムの場合、低位ブート・コードは、ＢＰＲＡＭイメージのエントリー・ポイントをコールすることによって、オペレーティング・システムにＢＰＲＡＭ

からブートするように要求することができる。

【００４４】

[0049] ブロック３２０に進み、本システムは、ＢＰＲＡＭ内に既にレイアウトしてあり、個々のモジュールをロードすることなくオペレーティング・システムを実行することができるイメージを識別する。このレイアウト・イメージは、オペレーティング・システムの典型的な格納済みフォーマットとは対照的に、実行する準備ができている。オペレーティング・システムの典型的な格納済みフォーマットは、個々のモジュールとして格納され、ロードおよび初期化プロセスを完了しないと、実行する準備ができない。ロードおよび初期化プロセスは、モジュールをメモリー内にマッピングし、ヒープまたは他のメモリー・エリアにデーター領域を作る等を行う。対照的に、本ソフトウェア・レイアウト・システムの実行フェーズは、これらのステップが既に早い時点で完了された状態で開始する。このように、ソフトウェア・レイアウト・システムは、ＢＰＲＡＭから直接、またはレイアウト・イメージをＢＰＲＡＭから他のメモリー・デバイスに素早くコピーしそこからオペレーティング・システムを実行することによって、直接オペレーティング・システムの実行に移ることができる。

【００４５】

[0050] 判断ブロック３３０に進み、本システムが以前にレイアウトされたイメージを識別した場合、本システムはブロック３４０に進み、高速ブートを実行する。または、本システムは完了し、オペレーティング・システムに関連する、格納されたモジュールをロードする従前のブート・プロセスに頼る。また、本システムは、識別された以前のレイアウト・イメージが、オペレーティング・システムの格納されているフォーマットに関して最新であるか否か判定することもできる（図示せず）。オペレーティング・システムのモジュールの格納されているバージョンが更新されていた場合、以前のレイアウト・イメージも更新する必要があるかもしれず、本システムは、このような更新をこの時点において実行することができ、または後の時点（例えば、次のブートの前）において更新を実行する必要性を知らせることもできる。

【００４６】

[0051] 判断ブロック３４０に進み、本システムは、識別された以前のレイアウト・イメージを正しい場所で実行することができるか否か判定し、そうである場合、ブロック３６０まで飛ばして、ＢＰＲＡＭイメージを実行する。そうでない場合、本システムはブロック３５０に進む。ＢＰＲＡＭがバイト・アドレス可能であっても、他でイメージを実行する正当な理由がある場合もある。例えば、ＢＰＲＡＭデバイスがＤＲＡＭよりも遅い場合、高速コピーをＤＲＡＭに対して実行し（例えば、直接メモリー・アクセス（ＤＭＡ）または他の方法を使用して）、次いでそこから実行する方が速いかもしれない。あるいはまたは加えて、ＢＰＲＡＭイメージは不必要にＢＰＲＡＭを摩耗させるが、ＤＲＡＭでは摩耗の心配なく正常に実行することができる書き込み可能セクションを含むことができる。これらおよび他の理由のために、ソフトウェア・レイアウト・システムの特定の実現例が、ＢＰＲＡＭから直接実行しないことを選択してもよい。

【００４７】

[0052] ブロック３５０に進み、本システムは、オペレーティング・システムの識別した以前のレイアウト・イメージを、実行のために、他のメモリー・デバイスにコピーする。このコピーは、コピー動作を高速化することができる、ＤＭＡまたは他の高速転送技法を含んでもよい。

【００４８】

[0053] ブロック３６０に進み、本システムは、識別された以前のレイアウト・イメージを使用して、オペレーティング・システムの実行を開始する。この時点におけるイメージは、なおもＢＰＲＡＭ内にあり直接実行されているか、またはＢＰＲＡＭから他のメモリー・デバイスへのコピーである。オペレーティング・システムの実行は、事前レイアウト・プロセスのためにもはや必要でなくなった初期化ステップに基づいて、オペレーティング・システムの代わりのエントリー・ポイントを識別すること、そしてレイアウトを前

もって実行したことによって生ずるオペレーティング・システムの状態と外部リソースを合わせるために実行される、外部リソース用の任意の初期化ルーチンを呼び出すことを含むことができる。

【００４９】

[0054] ブロック３７０に進み、本システムは、セクター・ベースの記憶デバイスに格納されているモジュールからオペレーティング・システムをロードする従前のブートよりも短い時間で、識別された以前のレイアウト・イメージを使用してアプリケーションを実行するために、オペレーティング・システムを準備完了状態に移す。本システムはブート・プロセスの大部分を前もって実行し、これらの部分は、メモリーよりも相対的に遅い記憶デバイスを伴うことにより通例最も遅いので、本明細書において説明するプロセスによって、ブート・プロセスを劇的に高速化することができる。ブロック３７０の後、これらのステップは終了する。

【００５０】

[0055] 図４は、一実施形態において、アプリケーションを検査してＢＰＲＡＭにおける事前レイアウトのためのコードおよびデーターを識別するための、ソフトウェア・レイアウト・システムの処理を示す流れ図である。図４および図５は、アプリケーションに適用したときの本システムの２つの異なる動作フェーズを示す。図４は、アプリケーションの実行準備完了レイアウトを判定し、このレイアウトをＢＰＲＡＭにコピーするフェーズを示し、図５は、図４の状態によって既に作成されているＢＰＲＡＭイメージからアプリケーションを初期化するフェーズを示す。

【００５１】

[0056] ブロック４１０において開始し、本システムは、アプリケーションの高速初期化イメージを作るために高速レイアウトを実行すべきターゲット・アプリケーションを識別する。このアプリケーションは、高速初期化イメージとはレイアウトが異なる１つ以上のモジュールを含む、格納済みフォーマットを有する。本システムは、管理ツールを提供することもできる。この管理ツールは、情報技術（ＩＴ）者によって実行することができ、またはオペレーティング・システムの一部として自動的に実行してアプリケーションを識別し、ＢＰＲＡＭにおける格納のために高速初期化イメージを作るプロセスを開始することもできる。アプリケーションを識別するには、オペレーティング・システムにインストールされたアプリケーションのリストを調べ、アプリケーションを実行する要求を傍受し、既に実行中のアプリケーションを監視し、またはターゲット・アプリケーションを示すモジュール・パスを受けることを含めばよい。

【００５２】

[0057] ブロック４２０に進み、本システムは、識別したターゲット・アプリケーションに対して静的分析を実行して、アプリケーションに関連する１つ以上の静的および動的にリンクされたモジュールを判定し、更にアプリケーションによって使用されるデーターの内、バイト・アドレス可能永続ランダム・アクセス・メモリー（ＢＰＲＡＭ）に格納できる程に変化しないものを識別する。当技術分野では、静的分析を実行するための種々の技法が存在するが、本プロセスが開始すると、一般に、アプリケーションに関連する主モジュールをロードし、実行可能フォーマットを調べてモジュールのセクション（リソース、定数データー、他のモジュールを参照するリンク・モジュール・セクション、および実行可能コードを含む１つ以上のコード・セクションを含むことができる）を識別し、そして恐らくは、任意の識別された機械コードまたは他の技法をディスアセンブルする(disassembly)。実施形態では、本システムが、動的にリンクされたライブラリー、アドイン実行モジュール、またはアプリケーションによって使用される他のリソースのような、任意の被参照モジュールをロードしてもよい場合もある。

【００５３】

[0058] ワード・プロセッサーのようなアプリケーションは、多くの場合、サード・パーティの拡張機能(third-party extensibility)が、１つ以上のアドイン・モジュールを介してアプリケーションに新たな機能性を追加することを許可し、本システムは、インス

トールされたアドインをいずれもレイアウト・イメージに含めることができる。加えて、アプリケーションは、当該アプリケーションが依存する１つ以上のモジュールの形態でランタイムを含むことができる１つ以上のフレームワークを使用して構築されてもよい。例えば、 MICROSOFT TM .NETは、アプリケーションが共通データー構造および機能のために利用することができる共通言語ランタイム（ＣＬＲ）を含み、これらをＢＰＲＡＭ内にアプリケーションと共に含めることができ、または複数のこのようなアプリケーションが利用することができる共通の仕方で含めることができる。

【００５４】

[0059] ブロック４３０に進み、本システムは、任意に、アプリケーションを実行し、アプリケーションの初期化を監視して、アプリケーションに関連する追加のモジュールおよびデーターを識別することによって、動的分析を実行する。動的分析は、静的分析によって発見されたものに加えて、モジュールおよび／またはデーターを発見し、更に静的分析によって識別された項目を確認することもできる。動的分析は、動的に割り当てられアプリケーションによって使用されるメモリー構造、ある種の条件下でのみ動的にロードされるモジュール、特定のハードウェアまたはオペレーティング・システム環境に基づいて異なるアプリケーションの部分等のような、静的分析によってでは判定することが難しい情報を提供することができる。静的および動的分析の間、本システムは、アプリケーションがどのように初期化するかについての情報を収集する。この情報は、事前レイアウト・イメージを生成するために使用することができ、この事前レイアウト・イメージから、アプリケーションは直接実行し、従前からの初期化ステップの少なくとも一部を飛ばすことができる。

【００５５】

[0060] 実施形態の中には、本システムが発見されたリソースの各々を識別するシグネチャー情報を捕らえるものもある。この識別情報は、ファイル・システムにおけるタイム・スタンプ、チェックサム、または特定のリソースが変化したときを示す他の値を含むことができる。次いで、本システムはこの情報を使用して、アプリケーション・ファイルが、製造業者からの更新によって更新された後というような、レイアウト分析を再度実行すべきときを判定することができる。本システムは、アプリケーションの各実行の間、または特定の回数の実行の後に、レイアウトに寄与した既知の各リソースを実地検証し、いずれかのリソースが変化したか否か判定し、いずれかのリソースが変化していた場合、新たなレイアウト・プロセスを開始してＢＰＲＡＭイメージを更新するプロセスを実行することができる。

【００５６】

[0061] 静的および動的分析の間、本システムはシステム構成データーベースのような外部リソースに影響を及ぼすあらゆる初期化を観察し、メモを取ることもできる。本ソフトウェア・レイアウト・システムは多数の典型的な初期化ステップを前もって実行することができるが、外部リソースに影響を及ぼすステップは、アプリケーションが初期化する度に実行される必要がある場合もある。このため、本システムは、このような初期化ルーチンを追跡し、これらのルーチンに対するポインターまたは他の基準を格納して、アプリケーションが事前レイアウト・イメージからロードしているときに、これらを実行できるようにするとよい。

【００５７】

[0062] ブロック４４０に進み、本システムは、アプリケーションを初期化するための事前レイアウト・イメージを格納するのに適したＢＰＲＡＭデバイスを識別する。ＢＰＲＡＭは、以前から入手可能な典型的なフラッシュ・メモリー、ハード・ドライブ、およびダイナミックＲＡＭ（ＤＲＡＭ）とは、バイト・アドレス可能であるだけでなく永続的であることが異なる。バイト・アドレス可能とは、ハード・ドライブやハード・ドライブと同様に動作するフラッシュでは典型的なように、セクター毎にロードするのではなく、ＢＰＲＡＭの各バイトに直接アドレスできることを意味する。コンピューター・プロセッサーは、通例、バイト・アドレス可能なメモリーに格納されたソフトウェア・コードだけを

実行することができる。永続的とは、ＢＰＲＡＭは電力サイクル（即ち、デバイスへの電源を切り、再度入れる）の間、そこに格納されているデーターを保持することを意味する。典型的なＤＲＡＭは、バイト・アドレス可能であるが永続的ではなく、そして典型的なハード・ドライブは、永続的であるがバイト・アドレス可能ではない。つまり、ＢＰＲＡＭは、これらの品質の双方を同時に所有するということにおいて新規である。このため、実行されるべき何かのイメージを、それを実行する要求に先立って形成し、ＢＰＲＡＭに格納し、次いでそこから直接または他のメモリー・デバイスにコピーした後に実行することができる。

【００５８】

[0063] ブロック４５０に進み、本システムは、任意に、ＢＰＲＡＭデバイスに関する摩耗ステータス情報を判定する。これは時の経過と共にデバイスの改良によって解決されるであろうが、現行のＢＰＲＡＭは、限られた回数しか首尾良く書き込むことができないという欠点がある。この回数は、頻繁でないＢＰＲＡＭの更新を可能にするには十分であるが、典型的なＤＲＡＭデバイスまたはハード・ドライブと同じ位の頻度である、ＢＰＲＡＭへの書き込みを可能にするには十分ではない。このため、本システムは、ＢＰＲＡＭデバイスへの書き込みを管理するために摩耗ステータス情報を追跡し、ＢＰＲＡＭデバイスの有効寿命を延ばすために１つ以上の技法を採用することができる。この情報は、デバイスの各領域に何回書き込まれたか、ＢＰＲＡＭのどの部分が現在使用中であるかまたは使用中でないか、各領域に利用可能な予測書き込み回数等を含むとよい。摩耗ステータス情報に基づいて、本システムはレイアウト・イメージを格納するための位置をＢＰＲＡＭ内において選択する。

【００５９】

[0064] ブロック４６０に進み、本システムは、アプリケーションの格納済みフォーマットとは異なる、アプリケーションのメモリー内レイアウトに基づいて、高速初期化イメージのためのレイアウトを作る。高速初期化イメージは、実行する準備ができており、アプリケーションを実行する要求を受ける前に作られる。メモリー内のアプリケーションの典型的なイメージは、格納されているモジュールをハード・ドライブ・ストレージからＤＲＡＭにロードすることによって作られるが、アプリケーションを実行する要求のときだけである。本システムは、ここでは、前もってこれらのステップを実行し、アプリケーションを実行する今後の要求が、これらのステップを毎回繰り返さないことによって、一層速くなるように、その結果をＢＰＲＡＭに入れる。レイアウト・イメージは、１つ以上のモジュール・コード・セクション、１つ以上のデーター・セクション、アプリケーションによって作られた１つ以上の読み取り専用ヒープ領域等の内容を含むことができる。

【００６０】

[0065] ブロック４７０に進み、本システムは、作ったレイアウト・イメージを、識別したＢＰＲＡＭデバイスにコピーする。ＢＰＲＡＭデバイスは、特定の書き込みプロセスを含むことができ、これによって本システムはイメージをデバイスにコピーする。また、本システムは、アプリケーション内において、アプリケーションの典型的な初期化ステップの全てを超えたポイントを表す特定の実行位置を識別することもできる。レイアウト・イメージは、ある量の初期化が実行された後の状態におけるアプリケーションを含むので、アプリケーションがＢＰＲＡＭから実行されるとき、通常よりも遅い実行位置において開始することができる。このため、本システムはアプリケーション・イメージ内においてエントリー・ポイント・アドレスを上書きする、またはアプリケーションをＢＰＲＡＭから実行するときにしかるべきポイントから開始するようにアプリケーションに命令する同様の変更を行い、これによって典型的な初期化ステップを飛ばすことができる。ブロック４７０の後、これらのステップは終了する。

【００６１】

[0066] 図５は、一実施形態において、ＢＰＲＡＭからアプリケーションを初期化するための、ソフトウェア・レイアウト・システムの処理を示す流れ図である。先に注記したように、図５は、図４のステップによって既に作られたＢＰＲＡＭイメージからアプリケ

ーションを初期化するフェーズを示す。

【００６２】

[0067] ブロック５１０において開始し、本システムはアプリケーションを実行する要求を受ける。アプリケーションは、通例、オペレーティング・システムの制御下で初期化する。オペレーティング・システムは、シェルを供給することができ、ユーザーはこのシェルを介して、アプリケーションやアプリケーション・プログラミング・インターフェース（ＡＰＩ）を起動することができ、ＡＰＩを介してアプリケーションは他のアプリケーションを実行することを要求することができる。また、オペレーティング・システムはオペレーティング・システムの初期化のときまたは特定のイベントが発生したときに自動的に起動されるサービスまたはデーモンの概念も含むことができる。アプリケーションがどのように起動されるかには関係なく、本システムは、アプリケーションが要求されたという通知を受ける。これは、ローダー要求、またはアプリケーションを実行することの要求に応答して一般に実行される他のステップをフックする(hook)デバイス・ドライバーをオペレーティング・システム内部で実行することによって行うことができる。実施形態の中には、ソフトウェア・レイアウト・システムが直接オペレーティング・システム内に、ネーティブな機構として組み入れられる場合もある。

【００６３】

[0068] ブロック５２０に進み、本システムは、ＢＰＲＡＭ内に以前にレイアウトされており、個々のモジュールをロードすることなくアプリケーションを実行することができる、イメージを識別する。レイアウト・イメージは、アプリケーションの典型的な格納済みフォーマットとは対照的に、実行する準備ができている。アプリケーションの典型的な格納済みフォーマットは、個々のモジュールとして格納され、ロードおよび初期化プロセスを完了しないと、実行する準備ができない。ロードおよび初期化プロセスは、モジュールをメモリー内にマッピングし、ヒープまたは他のメモリー・エリアにデーター領域を作る等を行う。対照的に、本ソフトウェア・レイアウト・システムの実行フェーズは、これらのステップが既に早い時点で完了された状態で開始する。このように、ソフトウェア・レイアウト・システムは、ＢＰＲＡＭから直接、またはレイアウト・イメージをＢＰＲＡＭから他のメモリー・デバイスに素早くコピーしそこからアプリケーションを実行することによって、直接アプリケーションの実行に移ることができる。

【００６４】

[0069] 判断ブロック５３０に進み、本システムが以前にレイアウトされたイメージを識別した場合、本システムはブロック５４０に進み、アプリケーションの高速ブートを実行する。または、本システムは完了し、アプリケーションに関連する、格納されたモジュールをロードする従前の初期化プロセスに頼る。また、本システムは、識別された以前のレイアウト・イメージが、アプリケーションの格納されているフォーマットに関して最新であるか否か判定することもできる（図示せず）。アプリケーションのモジュールの格納されているバージョンが更新されていた場合、以前のレイアウト・イメージも更新する必要があるかもしれず、本システムは、このような更新をこの時点において実行することができ、または後の時点（例えば、アプリケーションの次の実行の前）において更新を実行する必要性を知らせることができる。

【００６５】

[0070] 判断ブロック５４０に進み、本システムは、識別された以前のレイアウト・イメージを正しい場所で実行することができるか否か判定し、そうである場合、ブロック５６０まで飛ばして、ＢＰＲＡＭイメージを実行する。そうでない場合、本システムはブロック５５０に進む。ＢＰＲＡＭがバイト・アドレス可能であっても、イメージを他で実行する正当な理由があるかもしれない。例えば、ＢＰＲＡＭデバイスがＤＲＡＭよりも遅い場合、高速コピーをＤＲＡＭに対して実行し（例えば、直接メモリー・アクセス（ＤＭＡ）または他の方法を使用して）、次いでそこから実行する方が速いこともあり得る。あるいはまたは加えて、ＢＰＲＡＭイメージは、不必要にＢＰＲＡＭを摩耗させるが、ＤＲＡＭでは摩耗の心配なく正常に実行することができる書き込み可能セクションを含む場合も

ある。これらおよび他の理由のために、ソフトウェア・レイアウト・システムの特定の実現例が、ＢＰＲＡＭから直接実行しないことを選択してもよい。

【００６６】

[0071] ブロック５５０に進み、本システムは、アプリケーションの識別した以前のレイアウト・イメージを、実行のために、他のメモリー・デバイスにコピーする。このコピーは、コピー動作を高速化することができるＤＭＡまたは他の高速転送技法を含めばよい。

【００６７】

[0072] ブロック５６０に進み、本システムは、識別された以前のレイアウト・イメージを使用して、アプリケーションの実行を開始する。この時点におけるイメージは、なおもＢＰＲＡＭ内にあり直接実行されているか、またはＢＰＲＡＭから他のメモリー・デバイスへのコピーである。アプリケーションの実行は、事前レイアウト・プロセスのためにもはや必要でなくなった初期化ステップに基づいて、アプリケーションの代わりのエントリー・ポイントを識別すること、そしてレイアウトを前もって実行したことによって生ずるアプリケーションの状態と外部リソースを合わせるために実行される、外部リソース用の任意の初期化ルーチンを呼び出すことを含むことができる。

【００６８】

[0073] ブロック５７０に進み、本システムは、セクター・ベースの記憶デバイスに格納されているモジュールからアプリケーションをロードする従前のブートよりも短い時間で、識別された以前のレイアウト・イメージを使用してアプリケーションを実行するために、アプリケーションを準備完了状態に移す。本システムはブート・プロセスの大部分を前もって実行し、これらの部分は、メモリーよりも相対的に遅い記憶デバイスを伴うことにより通例最も遅いので、本明細書において説明するプロセスによって、ブート・プロセスを劇的に高速化することができる。ブロック５７０の後、これらのステップは終了する。

【００６９】

[0074] 図６は、一実施形態において、ソフトウェア・レイアウト・システムによって使用される種々のタイプのストレージおよびデーターの関係を示すブロック図である。典型的なコンピューター・システムは、プロセッサー６１０、非永続ダイナミックＲＡＭ６２０、およびセクター・ベースの永続ストレージ６３０を含む。オペレーティング・システム、アプリケーション実行可能モジュール、およびデーター・ファイルを含むファイルは、セクター・ベースの永続ストレージ６３０に格納される。動作の間、コンピューター・システムは、オペレーティング・システム・ファイルおよびアプリケーション・ファイルをセクター・ベースの永続ストレージ６３０からダイナミックＲＡＭ６４０に、プロセッサー６１０において実行される命令によってロードする。これらの命令は、実行可能モジュールを、それらの格納済みフォーマットから、プロセッサーによる実行の準備ができたフォーマットにフォーマットし直す。ソフトウェア・レイアウト・システムは、追加のタイプのメモリー、即ち、ＢＰＲＡＭ６４０を導入する。ＢＰＲＡＭ６４０は、永続的およびバイト・アドレス可能（即ち、セクター・ベースではない）の双方である。このコンピューター・システムは、格納済みフォーマットを実行の準備ができたフォーマットにコピーする初期化プロセスを最初に遂行することなく、オペレーティング・システムおよびアプリケーションを直接ＢＰＲＡＭから実行することができる。と言うよりは、ＢＰＲＡＭ６４０は該当するオペレーティング・システムおよび／またはアプリケーションの実行準備ができたフォーマットを既に含む。

【００７０】

[0075] 実施形態の中には、ソフトウェア・レイアウト・システムが、オペレーティング・システムに関連する変更版インストーラーを含み、この変更版インストーラーが、実行のためにアプリケーションによって使用される情報を記述する場合もある。アプリケーションに関連するインストール・ファイルは、アプリケーションを高速に実行するために適したレイアウト・イメージをオペレーティング・システムおよびインストーラーが作る

ための情報を提供するマニフェストを含むことができる。静的および動的分析とは異なり、インストール情報はアプリケーション製造業者から直接来て、アプリケーションの依存性および他のレイアウト考慮事項をより良く識別するために、製造業者にレイアウト・プロセスを調整することを可能にするが、静的および動的分析を更に実行してもよい。

【００７１】

[0076] 実施形態の中は、ソフトウェア・レイアウト・システムが、セクター・ベースおよびバイト・アドレス可能領域の組み合わせを有するＢＰＲＡＭを使用する場合もある。本明細書ではバイト・アドレス可能部分について先に詳細に説明したが、ＢＰＲＡＭは、セクター・ベースおよびバイト・アドレス可能領域が共存することを可能にするように区分することもできる。セクター・ベース部分は、アプリケーションまたはオペレーティング・システムの部分の格納を見込んでもよい。これらは、前もってレイアウトすることができないが、それでもなおセクター・ベースＢＰＲＡＭからは、それよりも遅いセクター・ベース媒体（例えば、回転ハード・ドライブ）からよりも速くロードすることができる。

【００７２】

[0077] 実施形態の中には、ソフトウェア・レイアウト・システムが複数のアプリケーションからのコールを傍受し、それらが本明細書において説明した全レイアウト・プロセスを使用しなくても、少なくとも一部のアプリケーション参照モジュール(application-referenced module)をＢＰＲＡＭから供給する場合もある。例えば、本システムは、動的リンク・ライブラリ（ＤＬＬ）からロードされた関数のアドレスを戻す共通のWIN32 TM APIコールを傍受し、ＤＬＬをロードしＤＬＬのメモリー・マッピングされた部分からアドレスを戻すのではなく、代わりに、ＤＬＬが既に実行準備完了フォーマットでレイアウトされているＢＰＲＡＭにおける領域へのポインターを戻すこともできる。同様に、本システムは、カーネル・コードを傍受して、デバイス・ドライバーのようなモジュールをロードし、これらのコールをＢＰＲＡＭにリディレクトすることができる。

【００７３】

[0078] 実施形態の中には、ソフトウェア・レイアウト・システムが仮想メモリー・アドレスのＢＰＲＡＭへのマッピングを、実行中のアプリケーションには透過的に行う場合もある。希にしか変化せずＢＰＲＡＭに格納されるアプリケーションおよび／またはオペレーティング・システムのデーターについて、システムはＢＰＲＡＭに格納された領域を標準的な仮想メモリー・アドレス空間にマッピングすることができる。このように、このデーターへのポインターを使用するアプリケーションまたはオペレーティング・システムは、これらのポインターによって参照されるデーターが、ＢＰＲＡＭ、ＤＲＡＭ、または他のどこかに格納されているのか否か知らず、以前のように機能し続けるが、データーは新たな位置にある可能性がある。

【００７４】

[0079] 実施形態では、ソフトウェア・レイアウト・システムが、高速初期化中に何らかのエラーが発生したときは従前の初期化プロセスに頼ることによって、フォールト・トレランスを提供する。モジュールに対する更新、バグ、または他のエラーは、高速初期化プロセスがオペレーティング・システムまたはアプリケーションを実行状態にし損なう原因にある場合がある。このような場合、本システムは従前の初期化プロセスを頼り、セクター・ベース媒体に格納されたモジュールからオペレーティング・システムまたはアプリケーションをロードすることができる。

【００７５】

[0080] 実施形態の中には、ソフトウェア・レイアウト・システムが、１つのアプリケーションの１つよりも多いインスタンス(more instances)を特定のコンピューター・ハードウェアにおいて実行することを可能にする場合もある。アプリケーションの多くはＢＰＲＡＭの共有可能な部分に格納されるので、アプリケーションの新たな各インスタンスによって消費されるリソースの量は大幅に削減される。これは、アプリケーションの複数のインスタンスを実行するのに、大量のリソースを消費する必要がないことを意味する。あ

るアプリケーションでは、多くのインスタンスを実行することは一般的であり、更に多くのインスタンスを実行する能力は非常に役立つ。

【００７６】

[0081] 以上のことから、本明細書では、例示の目的に限ってソフトウェア・レイアウト・システムの具体的な実施形態について説明したが、本発明の主旨および範囲から逸脱することなく種々の変更を行うことができることが認められよう。したがって、本発明は、添付した特許請求の範囲による以外では、限定されないこととする。

【図１】

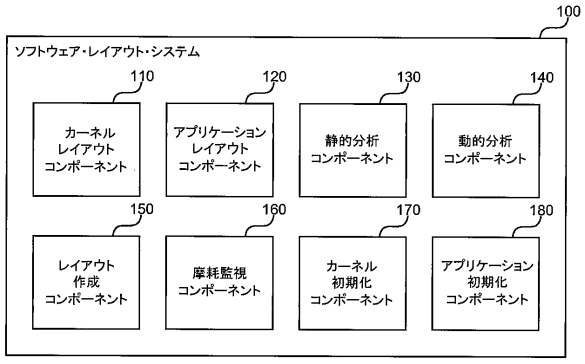

FIG. 1

【図２】

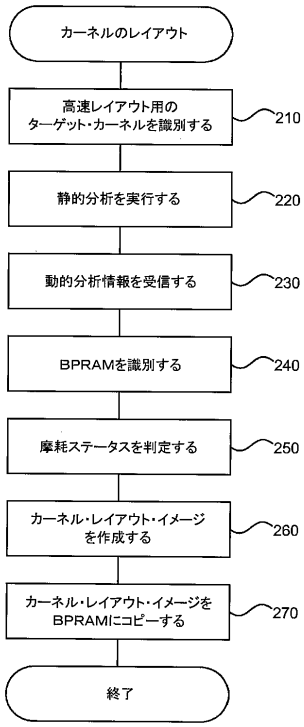

FIG. 2

【図３】

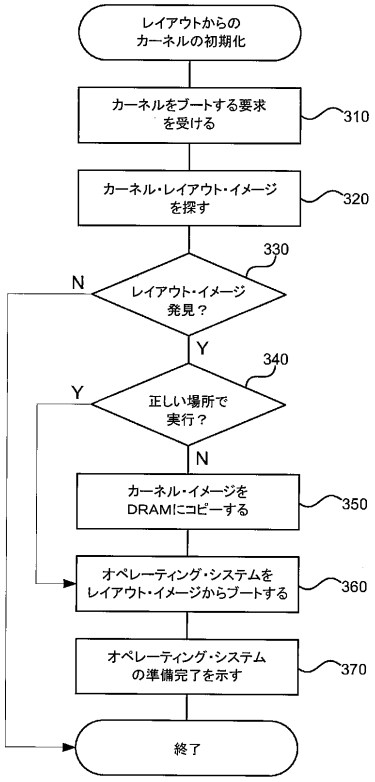

FIG. 3

【図４】

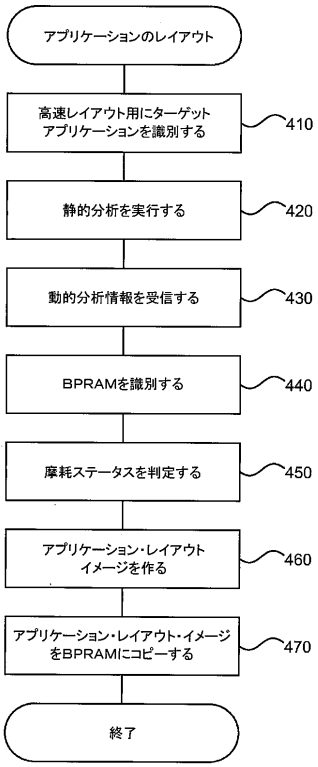

FIG. 4

【図５】

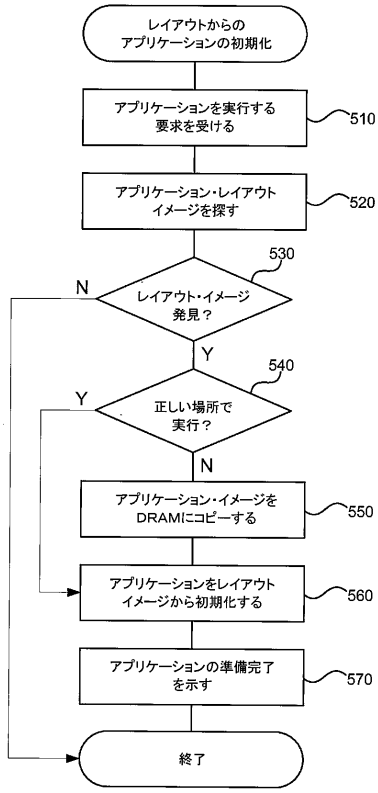

FIG. 5

【図６】

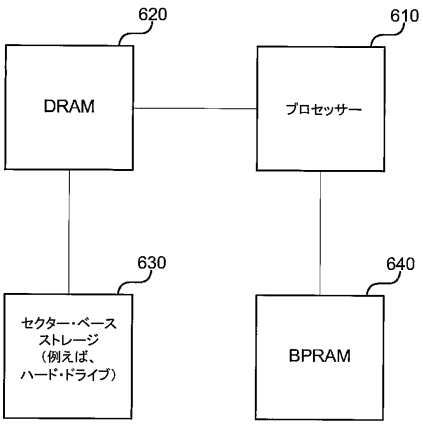

FIG. 6

フロントページの続き

(74)代理人 100120112
弁理士 中西 基晴
(72)発明者 ナイチンゲール，エドモンド
アメリカ合衆国ワシントン州９８０５２－６３９９，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，エルシーエイ－インターナショナル・パテンツ
(72)発明者 スリニヴァサン，キイ
アメリカ合衆国ワシントン州９８０５２－６３９９，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，エルシーエイ－インターナショナル・パテンツ

審査官 多賀 実

(56)参考文献 特開平１０－１９８５７０（ＪＰ，Ａ）
特表２００９－５１２０２０（ＪＰ，Ａ）
特開２００５－０１１１２０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ ９／４４５