(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第6329329号
(P6329329)
(45)発行日　平成30年5月23日(2018.5.23)　(24)登録日　平成30年4月27日(2018.4.27)

(51)Int.Cl.　　F I
G06F 8/41 (2018.01)　　G06F 9/44 322J
G06F 9/445 (2018.01)　　G06F 9/44 322L
　　G06F 9/06 640D

請求項の数 21　(全 33 頁)

(21)出願番号　特願2017-551130(P2017-551130)
(86)(22)出願日　平成28年3月29日(2016.3.29)
(65)公表番号　特表2018-510428(P2018-510428A)
(43)公表日　平成30年4月12日(2018.4.12)
(86)国際出願番号　PCT/US2016/024791
(87)国際公開番号　WO2016/195790
(87)国際公開日　平成28年12月8日(2016.12.8)
審査請求日　平成29年9月29日(2017.9.29)
(31)優先権主張番号　14/726,376
(32)優先日　平成27年5月29日(2015.5.29)
(33)優先権主張国　米国(US)

早期審査対象出願

(73)特許権者　502208397
グーグル　エルエルシー
アメリカ合衆国　カリフォルニア州　９４０４３　マウンテン　ビュー　アンフィシアター　パークウェイ　１６００
(74)代理人　100142907
弁理士　本田　淳
(72)発明者　グオ、ヤン
アメリカ合衆国　９４０４３　カリフォルニア州　マウンテン　ビュー　アンフィシアター　パークウェイ　１６００　グーグル　インコーポレイテッド内

最終頁に続く

(54)【発明の名称】コード・キャッシング・システム

(57)【特許請求の範囲】
【請求項１】
１以上のプロセッサと、
命令が記憶されている機械可読媒体と、を備え、前記命令は前記１以上のプロセッサによって実行されるときに前記１以上のプロセッサに、
実行を待機している一次ソース・コードの第１表示を受信する工程と、
前記第１表示を受信することに応じて前記一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認する工程と、
前記リソース・キャッシュ内のキャッシュ・ミスに際し、
前記一次ソース・コードからコンパイルされた第１実行可能コードを取得する工程と、
前記一次ソース・コード内で参照された二次ソース・コードを、前記二次ソース・コードのサイズ及び前記二次ソース・コードのコンパイル時間のうちの１以上に基づいて選択する工程と、
選択された前記二次ソース・コードからコンパイルされた第２実行可能コードを取得する工程と、
前記第１実行可能コードと前記第２実行可能コードとを、複数の実行コンテキスト内で再構築されるように構成されるシリアライズ・コードにシリアライズする工程と、
前記シリアライズ・コードと、選択された前記二次ソース・コード内で参照されたオブジェクトと、選択された前記二次ソース・コード内で参照された前記オブジェクトにア

クセスするためのアクセス方法と、前記オブジェクトのタイプとをキャッシュ・データとして前記リソース・キャッシュ内に記憶し、前記命令は、キャッシュされた前記オブジェクトについてキャッシュ・ヒットが生じたときであって、前記オブジェクトの前記タイプが、前記キャッシュ・ヒットに関するデータアクセスのオブジェクトのタイプに合致したとき、キャッシュされた前記オブジェクトにアクセスするために前記アクセス方法を前記１以上のプロセッサに用いさせる工程と、

選択された前記二次ソース・コードの実行履歴を取得する工程と、

選択された前記二次ソース・コードの前記実行履歴に基づいて、選択された前記二次ソース・コードを、選択された前記二次ソース・コードよりも頻繁に実行されている他の二次ソース・コードに置き換える工程と、

を含む動作を実行させる、コード・キャッシング・システム。

【請求項２】

前記命令はさらに、前記１以上のプロセッサに、

第１実行コンテキスト内の前記第１実行可能コードの実行から実行結果を取得する工程であって、選択された前記二次ソース・コードは前記実行結果に基づいて選択される工程を含む動作を実行させる、

請求項１に記載のシステム。

【請求項３】

選択された前記二次ソース・コードは、さらに、前記第２実行可能コードのサイズ、選択された前記二次ソース・コードが前記一次ソース・コード内で参照される予測、選択された前記二次ソース・コードが前記一次ソース・コード内で参照される回数、または選択された前記二次ソース・コードが前記一次ソース・コード内で参照される頻度のうち１以上に基づいて選択される、

請求項１に記載のシステム。

【請求項４】

前記命令はさらに、前記１以上のプロセッサに、

第２実行コンテキスト内で実行を待機している前記一次ソース・コードの第２表示を受信する工程であって、前記第２表示は前記第１表示に続いて受信される工程と、

前記第２表示を受信することに応じて前記一次ソース・コードと選択された前記二次ソース・コードとに対応するキャッシュ・データがあるかリソース・キャッシュを確認する工程と、

前記リソース・キャッシュ内のキャッシュ・ヒットに際し、

前記シリアライズ・コードを含む前記キャッシュ・データを前記リソース・キャッシュから取り出す工程と、

取り出された前記シリアライズ・コードを第３実行可能コードにデシリアライズする工程と、

前記第２実行コンテキスト内における実行用の前記第３実行可能コードを提供する工程と、

を含む動作を実行させる、

請求項２に記載のシステム。

【請求項５】

前記第１実行コンテキストと前記第２実行コンテキストとはコード実行用の２つの異なる仮想マシン環境である、

請求項４に記載のシステム。

【請求項６】

前記一次ソース・コードはウェブ・ページ内のスクリプトであり、前記第１実行コンテキストと前記第２実行コンテキストとは前記スクリプトが実行されるウェブ・ブラウザ・セッションである、

請求項５に記載のシステム。

【請求項７】

前記一次ソース・コードはウェブ・ページのトップ・レベル・スクリプトである、
請求項１に記載のシステム。
【請求項８】
前記第１実行可能コードは、前記第１実行可能コードに埋め込まれたメモリ・アドレスを含む１組の参照を含み、
前記第１実行可能コードをシリアライズすることは、埋め込まれた前記メモリ・アドレスを抽象アドレスと置き換える工程を含む、
請求項１に記載のシステム。
【請求項９】
１以上のプロセッサと、
命令が記憶されている機械可読媒体と、を備え、前記命令は前記１以上のプロセッサによって実行されるときに前記１以上のプロセッサに、
実行を待機している一次ソース・コードの第１表示を受信する工程と、
前記第１表示を受信することに応じて前記一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認する工程と、
前記リソース・キャッシュ内のキャッシュ・ミスに際し、
前記一次ソース・コードからコンパイルされた第１実行可能コードを取得する工程と、
前記一次ソース・コード内で参照された二次ソース・コードを、前記二次ソース・コードのサイズ及び前記二次ソース・コードのコンパイル時間のうちの１以上に基づいて選択する工程と、
選択された前記二次ソース・コードからコンパイルされた第２実行可能コードを取得する工程と、
前記一次ソース・コードのサイズ、所定期間内で前記一次ソース・コードが実行される回数もしくは頻度、または前記一次ソース・コードのコンパイル時間のうち１つ以上に基づいて、前記第１実行可能コードと前記第２実行可能コードとを、複数の実行コンテキスト内で再構築されるように構成されるシリアライズ・コードにシリアライズする工程と、
前記シリアライズ・コードと、選択された前記二次ソース・コード内で参照されたオブジェクトと、選択された前記二次ソース・コード内で参照された前記オブジェクトにアクセスするためのアクセス方法と、前記オブジェクトのタイプとをキャッシュ・データとして前記リソース・キャッシュ内に記憶し、前記命令は、キャッシュされた前記オブジェクトについてキャッシュ・ヒットが生じたときであって、前記オブジェクトの前記タイプが、前記キャッシュ・ヒットに関するデータアクセスのオブジェクトのタイプに合致したとき、キャッシュされた前記オブジェクトにアクセスするために前記アクセス方法を前記１以上のプロセッサに用いさせる工程と、
選択された前記二次ソース・コードの実行履歴を取得する工程と、
選択された前記二次ソース・コードの前記実行履歴に基づいて、選択された前記二次ソース・コードを、選択された前記二次ソース・コードよりも頻繁に実行されている他の二次ソース・コードに置き換える工程と、
を含む動作を実行させる、コード・キャッシング・システム。
【請求項１０】
選択された前記二次ソース・コードは、さらに、前記第２実行可能コードのサイズ、選択された前記二次ソース・コードが前記一次ソース・コード内で参照される予測、選択された前記二次ソース・コードが前記一次ソース・コード内で参照される回数、または選択された前記二次ソース・コードが前記一次ソース・コード内で参照される頻度のうち１以上に基づいて選択される、
請求項９に記載のシステム。
【請求項１１】
前記命令はさらに、前記１以上のプロセッサに、

第１実行コンテキスト内の前記第１実行可能コードの実行から実行結果を取得する工程であって、選択された前記二次ソース・コードは前記実行結果に基づいて選択される工程を含む動作を実行させる、
請求項９に記載のシステム。
【請求項１２】
前記命令はさらに、前記１以上のプロセッサに、
第２実行コンテキスト内で実行を待機している前記一次ソース・コードの第２表示を受信する工程であって、前記第２表示は前記第１表示に続いて受信される工程と、
前記第２表示を受信することに応じて前記一次ソース・コードと選択された前記二次ソース・コードとに対応するキャッシュ・データがあるかリソース・キャッシュを確認する工程と、
前記リソース・キャッシュ内のキャッシュ・ヒットに際し、
前記シリアライズ・コードを含む前記キャッシュ・データを前記リソース・キャッシュから取り出す工程と、
取り出された前記シリアライズ・コードを第３実行可能コードにデシリアライズする工程と、
前記第２実行コンテキスト内における実行用の前記第３実行可能コードを提供する工程と、
を含む動作を実行させる、
請求項１１に記載のシステム。
【請求項１３】
前記第１実行コンテキストと前記第２実行コンテキストとはコード実行用の２つの異なる仮想マシン環境である、
請求項１２に記載のシステム。
【請求項１４】
前記第１実行可能コードは、前記第１実行可能コードに埋め込まれたメモリ・アドレスを含む１組の参照を含み、
前記第１実行可能コードをシリアライズすることは、埋め込まれた前記メモリ・アドレスを抽象アドレスと置き換える工程を含む、
請求項９に記載のシステム。
【請求項１５】
命令を含むコンピュータ可読媒体であって、前記命令はコンピュータによって実行されるときに前記コンピュータに、
実行を待機している一次ソース・コードの第１表示を受信させ、
前記第１表示を受信することに応じて前記一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認させ、
前記リソース・キャッシュ内のキャッシュ・ミスに際し、
前記一次ソース・コードからコンパイルされた第１実行可能コードを取得させ、
第１実行コンテキスト内の前記第１実行可能コードの実行から実行結果を取得させ、
前記一次ソース・コード内で参照された二次ソース・コードを前記実行結果に基づいて、さらに、前記二次ソース・コードのサイズ及び前記二次ソース・コードのコンパイル時間のうちの１以上に基づいて選択させ、
選択された前記二次ソース・コードからコンパイルされた第２実行可能コードを取得させ、
前記第１実行可能コードと前記第２実行可能コードとを、複数の実行コンテキスト内で再構築されるように構成されるシリアライズ・コードにシリアライズさせ、
前記シリアライズ・コードと、選択された前記二次ソース・コード内で参照されたオブジェクトと、選択された前記二次ソース・コード内で参照された前記オブジェクトにアクセスするためのアクセス方法と、前記オブジェクトのタイプとをキャッシュ・データとして前記リソース・キャッシュ内に記憶させ前記命令は、キャッシュされた前記オブジェ

クトについてキャッシュ・ヒットが生じたときであって、前記オブジェクトの前記タイプが、前記キャッシュ・ヒットに関するデータアクセスのオブジェクトのタイプに合致したとき、キャッシュされた前記オブジェクトにアクセスするために前記アクセス方法を前記コンピュータに用いさせ、
選択された前記二次ソース・コードの実行履歴を取得させ、
選択された前記二次ソース・コードの前記実行履歴に基づいて、選択された前記二次ソース・コードを、選択された前記二次ソース・コードよりも頻繁に実行されている他の二次ソース・コードに置き換えさせる
コンピュータ可読媒体。
【請求項１６】
前記第１実行可能コードは、前記第１実行可能コードに埋め込まれたメモリ・アドレスを含む１組の参照を含み、
前記第１実行可能コードをシリアライズすることは、埋め込まれた前記メモリ・アドレスを抽象アドレスと置き換えることを含む、
請求項１５に記載のコンピュータ可読媒体。
【請求項１７】
前記実行結果は、前記一次ソース・コード内で参照された選択された前記二次ソース・コードが前記一次ソース・コードの実行時に実行される回数を示す、
請求項１５に記載のコンピュータ可読媒体。
【請求項１８】
前記命令はさらに、前記コンピュータに、
第２実行コンテキスト内で実行を待機している前記一次ソース・コードの第２表示を受信させ、前記第２表示は前記第１表示に続いて受信され、
前記第２表示を受信することに応じて前記一次ソース・コードと選択された前記二次ソース・コードとに対応するキャッシュ・データがあるかリソース・キャッシュを確認させ、
前記リソース・キャッシュ内のキャッシュ・ヒットに際し、
前記シリアライズ・コードを含む前記キャッシュ・データを前記リソース・キャッシュから取り出させ、
取り出された前記シリアライズ・コードを第３実行可能コードにデシリアライズさせ、
前記第２実行コンテキスト内における実行用の前記第３実行可能コードを提供させる、
請求項１５に記載のコンピュータ可読媒体。
【請求項１９】
前記第１実行コンテキストと前記第２実行コンテキストとはコード実行用の２つの異なる仮想マシン環境である、
請求項１８に記載のコンピュータ可読媒体。
【請求項２０】
前記命令はさらに、前記コンピュータに、
前記第１実行可能コードに関連付けられている第１実行プロファイルと前記第２実行可能コードに関連付けられている第２実行プロファイルとを取得させ、
前記第１実行プロファイルと第２実行プロファイルとをシリアライズさせ、
シリアライズされた前記第１実行プロファイルと前記第２実行プロファイルとを前記リソース・キャッシュ内に記憶させ、取り出された前記シリアライズ・コードをデシリアライズすることは前記第１実行プロファイルと前記第２実行プロファイルとをデシリアライズさせることを含む、
請求項１８に記載のコンピュータ可読媒体。
【請求項２１】
命令を含むメモリと、

１以上のプロセッサと、を備え、該１以上のプロセッサは、
リソース・キャッシュ内のキャッシュ・ミスに際し、
実行を待機している一次ソース・コードからコンパイルされた第１実行可能コードを取得することと、
前記一次ソース・コード内で参照された二次ソース・コードからコンパイルされた第２実行可能コードを取得し、前記二次ソース・コードは、前記二次ソース・コードのサイズ及び前記二次ソース・コードのコンパイル時間のうちの１以上に基づいて選択されることと、
前記第１実行可能コードと前記第２実行可能コードとを、複数の実行コンテキスト内で再構築されるように構成されるシリアライズ・コードにシリアライズすることと、
前記シリアライズ・コードと、選択された前記二次ソース・コード内で参照されたオブジェクトと、選択された前記二次ソース・コード内で参照された前記オブジェクトにアクセスするためのアクセス方法と、前記オブジェクトのタイプとをキャッシュ・データとして前記リソース・キャッシュ内に記憶し、前記命令は、キャッシュされた前記オブジェクトについてキャッシュ・ヒットが生じたときであって、前記オブジェクトの前記タイプが、前記キャッシュ・ヒットに関するデータアクセスのオブジェクトのタイプに合致したとき、キャッシュされた前記オブジェクトにアクセスするために前記アクセス方法を前記１以上のプロセッサに用いさせることと、
選択された前記二次ソース・コードの実行履歴を取得することと、
選択された前記二次ソース・コードの前記実行履歴に基づいて、選択された前記二次ソース・コードを、選択された前記二次ソース・コードよりも頻繁に実行されている他の二次ソース・コードに置き換えることと、
を行うように前記命令を実行するように構成されている、コード・キャッシング・システム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、実行可能コードをキャッシュすることに関し、特に、コンパイルするためのソース・コードを選択し、実行可能コードをシリアライズおよびキャッシュし、キャッシュされた実行可能コードを実行前にデシリアライズすることに関する。
【背景技術】
【０００２】
仮想マシン（ＶＭ）は、オンデマンドでのコンパイルに際してソフトウェア・コードを実行するために用いられる。オンデマンドでのコンパイルは、実行時コンパイルとも称され、コードのコンパイルがコードの実行直前に行われるか、コードの一部がオンデマンドに（レイジーに）コンパイルされる。コードがリソース（たとえば、ウェブ・ページ内のリソース）に埋め込まれると、クライアント・デバイス上で動作しているアプリケーション（たとえば、ブラウザ）内でリソースがレンダリングされる速度は、埋め込まれたコードの実行速度によって決まる。クライアント・デバイスのユーザは、同一のワーク・セッション（たとえば、ウェブ・ブラウザ・セッション）内で、または、複数の独立ワーク・セッションにわたって、同一のリソースに繰り返しアクセスし得る。しかしながら、ユーザがリソースにアクセスするたびにコードがコンパイルされる必要がある場合、著しく長い時間が各コンパイルに必要となり得る。
【図面の簡単な説明】
【０００３】
【図１】コード・キャッシングが処理され得る例示的ネットワーク環境を示す図。
【図２】コード・キャッシングが提供され得るクライアント・デバイスを示す図。
【図３】スクリプトのキャッシュ・コードへのシリアライズとキャッシュ・コードの実行可能コードへのデシリアライズとが提供され得るコード・キャッシング・プラットフォームのフロー図。

【図４Ａ】スクリプトのキャッシュ・コードへのシリアライズが提供され得るプロセスの例を示す図。
【図４Ｂ】スクリプトのキャッシュ・コードへのシリアライズが提供され得るプロセスの例を示す図。
【図５】本願の技術のいくつかの実施形態が実装される例示的電子システムを概念的に示す図。
【発明を実施するための形態】
【０００４】
様々な態様において、開示される主題はシステム内において具現化される。前記システムは、１以上のプロセッサを備える。前記システムはさらに、メモリを備える。前記メモリは命令を備え、前記命令は実行されるときに前記１以上のプロセッサに、方法を実装させる。前記命令は、実行を待機している一次ソース・コードの第１表示を受信するためのコードを含む。前記命令は、前記第１表示を受信することに応じて前記一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認するためのコードを含む。前記命令は、前記リソース・キャッシュ内のキャッシュ・ミスに際し、前記一次ソース・コードからコンパイルされた第１実行可能コードを取得するためのコードを含む。前記命令は、前記一次ソース・コード内で参照された二次ソース・コードを選択するためのコードを含む。前記命令は、選択された前記二次ソース・コードからコンパイルされた第２実行可能コードを取得するためのコードを含む。前記命令は、前記第１実行可能コードと前記第２実行可能コードとをシリアライズ・コードにシリアライズするためのコードを含む。前記命令は、前記シリアライズ・コードをキャッシュ・データとして前記リソース・キャッシュ内に記憶するためのコードを含む。
【０００５】
これらの、および、他の実施形態は、以下の特徴のうち１以上を含み得る。前記命令は、第１実行コンテキスト内の前記第１実行可能コードの実行から実行結果を取得するためのコードであって、前記二次ソース・コードは前記実行結果に基づいて選択されるコードを含んでもよい。前記二次ソース・コードは、前記二次ソース・コードのサイズ、前記第２実行可能コードのサイズ、前記一次ソース・コード内で前記二次ソース・コードが参照される予測、前記一次ソース・コード内で前記二次ソース・コードが参照される回数、前記一次ソース・コード内で前記二次ソース・コードが参照される頻度、または前記二次ソース・コードのコンパイル時間のうち１以上に基づいて選択されてもよい。前記命令は、第２実行コンテキスト内で実行を待機している前記一次ソース・コードの第２表示を受信するためのコードを含んでもよくであって、前記第２表示は前記第１表示に続いて受信される。前記命令は、前記第２表示を受信することに応じて前記一次ソース・コードと選択された前記二次ソース・コードとに対応するキャッシュ・データがあるかリソース・キャッシュを確認するためのコードを含んでもよい。前記命令は、前記リソース・キャッシュ内のキャッシュ・ヒットに際し、前記シリアライズ・コードを含む前記キャッシュ・データを前記リソース・キャッシュから取り出すためのコードを含んでもよい。前記命令は、取り出された前記シリアライズ・コードを第３実行可能コードにデシリアライズするためのコードを含んでもよい。前記命令は、前記第２実行コンテキスト内における実行用の前記第３実行可能コードを提供する工程ためのコードを含んでもよい。
【０００６】
これらの、および、他の実施形態は、以下の特徴のうち１以上を含み得る。前記第１実行コンテキストと前記第２実行コンテキストとはコード実行用の２つの異なる仮想マシン環境であってもよい。前記一次ソース・コードはウェブ・ページ内のスクリプトであってもよく、前記第１実行コンテキストと前記第２実行コンテキストとは前記スクリプトが実行されるウェブ・ブラウザ・セッションであってもよい。前記一次ソース・コードはウェブ・ページのトップ・レベル・スクリプトであってもよい。前記第１実行可能コードは、前記第１実行可能コードに埋め込まれたメモリ・アドレスを含む１組の参照を含んでもよい。前記命令は、埋め込まれた前記メモリ・アドレスを抽象アドレスと置き換えるための

コードを含んでもよい。

【０００７】

１つの革新的な態様において、開示される主題はシステム内において具現化される。前記システムは、１以上のプロセッサを含む。前記システムはさらに、メモリを含む。前記メモリは、命令を含み、実行されるときに前記１以上のプロセッサに方法を実装させる。前記命令は実行を待機している一次ソース・コードの第１表示を受信するためのコードを含む。前記命令は、前記第１表示を受信することに応じて前記一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認するためのコードを含む。前記命令は、前記リソース・キャッシュ内のキャッシュ・ミスに際し、前記一次ソース・コードからコンパイルされた第１実行可能コードを取得するためのコードを含む。前記命令は、前記一次ソース・コードのサイズ、所定期間内で前記一次ソース・コードが実行される回数もしくは頻度、または前記一次ソース・コードのコンパイル時間のうち１つ以上に基づいて、前記第１実行可能コードをシリアライズ・コードにシリアライズするためのコードを含む。前記命令は、前記シリアライズ・コードをキャッシュ・データとして前記リソース・キャッシュ内に記憶するためのコードを含む。

【０００８】

これらの、および、他の実施形態は、以下の特徴のうち１以上を含み得る。前記命令は、前記一次ソース・コード内で参照される二次ソース・コードを、少なくとも前記二次ソース・コードのサイズ、第２実行可能コードのサイズ、前記一次ソース・コード内で前記二次ソース・コードが参照される回数、前記一次ソース・コード内で前記二次ソース・コードが参照される頻度、もしくは前記二次ソース・コードのコンパイル時間、またはそれらの組み合わせに基づいて、選択するためのコードを含んでもよい。前記命令は、前記二次ソース・コードからコンパイルされた第２実行可能コードを取得するためのコードを含んでもよい。前記命令は、前記第２実行可能コードを前記シリアライズ・コードにシリアライズするためのコードを含んでもよい。前記命令は、第１実行コンテキスト内の前記第１実行可能コードの実行から実行結果を取得するためのコードを含んでもよい。前記二次ソース・コードは前記実行結果に基づいて選択される。前記命令はさらに、第２実行コンテキスト内で実行を待機している前記一次ソース・コードの第２表示を受信するためのコードを含んでもよい。前記第２表示は前記第１表示に続いて受信される。前記第２表示を受信することに応じて前記一次ソース・コードと選択された前記二次ソース・コードとに対応するキャッシュ・データがあるかリソース・キャッシュを確認するためのコードを含んでもよい。前記命令は前記リソース・キャッシュ内のキャッシュ・ヒットに際し、前記シリアライズ・コードを含む前記キャッシュ・データを前記リソース・キャッシュから取り出すためのコードを含んでもよい。前記命令は取り出された前記シリアライズ・コードを第３実行可能コードにデシリアライズするためのコードを含んでもよい。前記命令は前記第２実行コンテキスト内における実行用の前記第３実行可能コードを提供するためのコードを含んでもよい。

【０００９】

これらの、および、他の実施形態は、以下の特徴のうち１以上を含み得る。前記第１実行コンテキストと前記第２実行コンテキストとはコード実行用の２つの異なる仮想マシン環境であってもよい。前記第１実行可能コードは、前記第１実行可能コードに埋め込まれたメモリ・アドレスを含む１組の参照を含んでもよく、前記第１実行可能コードをシリアライズするだめのコードを含む命令は、埋め込まれた前記メモリ・アドレスを抽象アドレスと置き換えためのコードを含んでもよい。

【００１０】

１つの革新的な態様において、開示される主題はコンピュータ可読媒体内において具現化される。前記コンピュータ可読媒体は、コンピュータによって実行される命令を含む。前記命令はコンピュータに、実行を待機している一次ソース・コードの第１表示を受信させるためのコードを含む。前記命令はコンピュータに、前記第１表示を受信することに応じて前記一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシ

ュを確認させるためのコードを含む。前記命令はコンピュータに、前記リソース・キャッシュ内のキャッシュ・ミスに際し、前記一次ソース・コードからコンパイルされた第１実行可能コードを取得させるためのコードを含む。前記命令はコンピュータに、第１実行コンテキスト内の前記第１実行可能コードの実行から実行結果を取得させるためのコードを含む。前記命令はコンピュータに、前記一次ソース・コード内で参照された二次ソース・コードを前記実行に基づいて選択させるためのコードを含む。前記命令はコンピュータに、選択された前記二次ソース・コードからコンパイルされた第２実行可能コードを取得させるためのコードを含む。前記命令はコンピュータに、前記第１実行可能コードと前記第２実行可能コードとをシリアライズ・コードにシリアライズさせるためのコードを含む。前記命令はコンピュータに、前記シリアライズ・コードをキャッシュ・データとして前記リソース・キャッシュ内に記憶させるためのコードを含む。

【００１１】

これらの、および、他の実施形態は、以下の特徴のうち１以上を含み得る。前記第１実行可能コードは、前記第１実行可能コードに埋め込まれたメモリ・アドレスを含む１組の参照を含んでもよく、前記第１実行可能コードをシリアライズすることは、埋め込まれた前記メモリ・アドレスを抽象アドレスと置き換えることを含んでもよい。前記実行結果は、前記一次ソース・コード内で参照された前記二次ソース・コードが前記一次ソース・コードの実行時に実行される回数を示してもよい。前記命令は前記コンピュータに、第２実行コンテキスト内で実行を待機している前記一次ソース・コードの第２表示を受信させるためのコードを含んでもよい。前記第２表示は前記第１表示に続いて受信される。前記第２表示を受信することに応じて前記一次ソース・コードと選択された前記二次ソース・コードとに対応するキャッシュ・データがあるかリソース・キャッシュを確認させるためのコードを含んでもよい。前記命令は前記コンピュータに、前記リソース・キャッシュ内のキャッシュ・ヒットに際し、前記シリアライズ・コードを含む前記キャッシュ・データを前記リソース・キャッシュから取り出させるためのコードを含んでもよい。前記命令は前記コンピュータに、取り出された前記シリアライズ・コードを第３実行可能コードにデシリアライズさせるためのコードを含んでもよい。前記命令は前記コンピュータに、前記第２実行コンテキスト内における実行用の前記第３実行可能コードを提供させるためのコードを含んでもよい。前記第１実行コンテキストと前記第２実行コンテキストとはコード実行用の２つの異なる仮想マシン環境であってもよい。前記命令は前記コンピュータに、前記第１実行可能コードに関連付けられている第１実行プロファイルと前記第２実行可能コードに関連付けられている第２実行プロファイルとを取得させるためのコードを含んでもよい。前記命令は前記コンピュータに、前記第１実行プロファイルと第２実行プロファイルとをシリアライズさせるためのコードを含んでもよい。前記命令は前記コンピュータに、シリアライズされた前記第１実行プロファイルと前記第２実行プロファイルとを前記リソース・キャッシュ内に記憶させるためのコードを含んでもよい。取り出された前記シリアライズ・コードをデシリアライズすることは前記第１実行プロファイルと前記第２実行プロファイルとをデシリアライズさせることを含んでもよい。

【００１２】

本願の技術の様々な構成が例示のために図示および説明される以下の詳細な説明から、当業者によって本願の技術の他の構成が容易に識別されると理解される。以下から分かるように、本願の技術の範囲を逸脱することなく、本願の技術は他の異なる構成が可能であり、そのいくつかの詳細は他の様々な点において変形が可能である。したがって、図面および詳細な説明は、本質的に例示として考えられるべきものであって、限定として考えられるものではない。

【００１３】

本願の技術の特徴は、添付される請求項に示される。しかしながら、説明の便宜上、本願の技術のいくつかの実施形態は、図１～図５に示されている。

用語解説

抽象化する（抽象化）：抽象化とは、オブジェクト（たとえば、コード）および実行コ

ンテキストに特有である他のオブジェクト内にあるメモリ・アドレスへの参照を、オブジェクトが新しい実行コンテキスト内で再生成されるように、参照（例えば、リロケーション・データ）を再生成するための命令に置き換えるプロセスである。

【００１４】

シリアライズする（シリアライズ）：シリアライズとは、移送可能な順次データであって、オブジェクトに関する情報（たとえば、オブジェクトのタイプまたはオブジェクト内に記憶されたデータのタイプ）だけでなくオブジェクトのデータを含む順次データとして、オブジェクト（たとえば、コンパイルされた実行可能コード）の表現を生成するプロセスである。

【００１５】

デシリアライズする（デシリアライズ）：デシリアライズとは、シリアライズされたオブジェクトの順次データからオブジェクト（たとえば、コンパイルされた実行可能コード）を抽出および生成するプロセスである。

【００１６】

ソース・コード：ソース・コードとは、コンピュータによって行われ、人間可読言語を用いて書かれるアクションを指定するコンピュータ命令の集合である。

実行可能コード：実行可能コードとは、コンピュータによって実行可能であって、ソース・コードをパースおよびコンパイルすることによって生成される、１組のマシン言語命令である。

【００１７】

実行コンテキスト（実行環境）：実行コンテキストとは、実行可能コードが実行されるコンピュータ・システムまたはコンピュータ・システム（たとえば、仮想マシン）のソフトウェア・エミュレーションを規定する１組の操作パラメータである。

【００１８】

リロケーション・データ：リロケーション・データとは、現行の実行コンテキスト内のメモリ位置またはオブジェクトに対する抽象化されたコード参照を再記憶するために用いられる命令または情報である。

【００１９】

以下に示す詳細な説明は、本願の技術の様々な構成の説明を意図し、本願の技術が実施される唯一の構成だけを表すことを意図しない。添付される図面は、本明細書に組み込まれ、詳細な説明の一部を構成する。詳細な説明は、本願の技術の十分な理解を提供するための具体的詳細を含む。しかしながら、本願の技術は本明細書に示される具体的詳細には限定されず、これら具体的詳細なしに実施されることは明白である。いくつかの事例においては、本願の技術の概念を曖昧にすることを避けるために、構造および要素はブロック図で示される。

【００２０】

リソースのロード時間は、コンピューティング・デバイスを通じてリソースにアクセスしようとする際のユーザ・エクスピリエンスの重要な要素である。リソースは、コンピューティング・デバイス内のローカル・メモリからロードされるアプリケーション、ファイル、またはドキュメントであってもよいし、ネットワーク・リソース（たとえば、ネットワークからロードされるウェブ・ページ内のファイルまたはドキュメント）であってもよい。しかしながら、動的スクリプト（たとえば、リソース内に埋め込まれているＪａｖａＳｃｒｉｐｔコード）は、実行される前にコンパイルされる必要がある。これによって、リソースをロードするプロセスが低速化する可能性がある。したがって、コンパイル・コードをリソース・キャッシュにキャッシュすること、および、後続のセッション時のリソースへの将来的なアクセス用にキャッシュ・コードを再利用することによって、コンピューティング・デバイス上でリソースをレンダリングする効率性を改善することが望まれている。

【００２１】

ソース・コードは、コンパイラ・プログラムによって、コンピュータによって実行可能

なローレベル・マシン・コードに変換され得る。マシン・コードは、その後、実行用に記憶され得る。それに代えて、直接的にオン・ザ・フライでソース・コード・プログラムの結果を分析および実行するために、インタプリタが用いられ得る。アプリケーションまたはドキュメント（たとえば、実行可能コードを生成するためのウェブ・ページ）に埋め込まれたソース・コード（たとえば、ＪａｖａＳｃｒｉｐｔ（登録商標）コード）をパースおよびコンパイルすることは、リソースをロードするためのクリティカル・パス上である。コンピュータ・プロセッサによって実行されるとき、実行可能コードによって、プロセッサは命令に従ってタスクを行う。マシン言語は、コンピュータがハードウェアにおいて実行する１組のネイティブ命令である。ローディング・プロセスを高速化するために、コンパイル・コードがシリアライズおよびキャッシュされる（たとえば、バイナリ・フォーマットにおいて）。実行可能コードがキャッシュされると、スクリプトの後続の実行において、ソース・コードはリパースおよびリコンパイルされる必要がなくなる。そして、実行可能コードはキャッシュからデシリアライズされて実行される。

【００２２】

コンパイル・コードのシリアライズ中、コードの状態は、１つの実行環境から他の実行環境へと移送される形式へと変換される。シリアライズ・プロセスは、データ構造またはオブジェクト状態を、シリアライズ・オブジェクトに埋め込まれた情報に基づいて、同一または別のコンテキスト（たとえば、実行環境）内で再構築されるフォーマットへと変換する。シリアライズは、コンパイル・コードから、たとえば、定数、文字列リテラル、オブジェクト・リテラルを表すデータ構造、オンデマンドでのコンパイルに必要なリロケーション・データを含む機能オブジェクト、共有される部分コード、実行環境（たとえば、ＶＭ）への参照など、そのコードの様々なコンポーネントと、そのコードが参照するオブジェクトとについての実行可能コードを提供し得る。

【００２３】

図１は、コード・キャッシングが処理され得る例示的ネットワーク環境１００を示す。ネットワーク環境１００は、クライアント・デバイス１０１ａおよび１０１ｎと、サーバ１０９ａおよび１０９ｍとを含む。２つのクライアント・デバイス１０１ａおよび１０１ｎと、２つのサーバ１０９ａおよび１０９ｍとが図１に示されるが、本願の技術はそれらに限定されず、３つ以上のクライアント・デバイスおよびサーバ、または、たった１つのクライアント・デバイスおよび／またはサーバに適用され得る。クライアント・デバイス１０１ａおよび１０１ｎと、サーバ１０９ａおよび１０９ｍとは、ネットワーク１０５を通じて互いに通信可能であり得る。サーバ１０９ａまたは１０９ｍは、図示しないクライアント・デバイスおよびコンピュータ可読記憶媒体と類似するコンピューティング・デバイスを含み得る。

【００２４】

クライアント・デバイス１０１ａおよび１０１ｎの各々は、様々な形態の処理デバイスを表し得る。例示的処理デバイスは、デスクトップ・コンピュータ、ラップトップ・コンピュータ、ハンドヘルド・コンピュータ、パーソナル・デジタル・アシスタント（ＰＤＡ）、セルラー式電話機、ネットワーク・アプライアンス、カメラ、スマートフォン、拡張汎用パケット無線システム（ＥＧＰＲＳ）モバイル・フォン、メディア・プレイヤ、ナビゲーション・デバイス、電子メール・デバイス、ゲーム機、または、これらのデータ処理デバイスもしくはその他のデータ処理デバイスの任意の組み合わせを含み得る。クライアント・デバイス１０１ａおよび１０１ｎとサーバ１０９ａおよび１０９ｍとは、それ以外のクライアント・デバイス１０１ａおよび１０１ｎとサーバ１０９ａおよび１０９ｍとのうちいずれかにおいて実行もしくは記憶されたアプリケーション・ソフトウェアへのアクセスを提供されるか、該アプリケーション・ソフトウェアを受信し得る。

【００２５】

サーバ１０９ａおよび１０９ｍは、クライアント・デバイス１０１ａおよび１０１ｎにコンテンツを提供するためのプロセッサ、メモリ、および通信能力を有する任意のシステムまたはデバイスであり得る。いくつかの例示的態様において、サーバ１０９ａまたは１

０９ｍは、１つのコンピューティング・デバイス（たとえば、コンピュータ・サーバ）であり得る。他の実施形態において、サーバ１０９ａまたは１０９ｍは、協働してサーバ・コンピュータのアクション（たとえば、クラウド・コンピューティング）を実行する２以上のコンピューティング・デバイスを表し得る。さらに、サーバ１０９ａまたは１０９ｍは、様々な形態のサーバを表してもよく、ウェブサーバ、アプリケーション・サーバ、プロキシ・サーバ、ネットワーク・サーバ、またはサーバ・ファームを含むが、これらに限定されない。

【００２６】

いくつかの態様では、クライアント・デバイス１０１ａおよび１０１ｎは、図示しない通信インターフェースを通じて無線通信してもよい。通信インターフェースは、必要に応じてデジタル信号処理回路を含み得る。通信インターフェースは、様々なモードまたはプロトコル（たとえば、グローバル・システム・フォー・モバイル・コミュニケーション（ＧＳＭ（登録商標））ボイス・コール、ショート・メッセージ・サービス（ＳＭＳ）、拡張メッセージング・サービス（ＥＭＳ）、あるいは、マルチメディア・メッセージング・サービス（ＭＭＳ）・メッセージング、符号分割多元接続（ＣＤＭＡ）、時分割多元接続（ＴＤＭＡ）、パーソナル・デジタル・セルラー（ＰＤＣ）、ワイドバンド符号分割多元接続（ＷＣＤＭＡ（登録商標））、ＣＤＭＡ２０００、または、汎用パケット無線システム（ＧＰＲＳ）、他の形式またはプロトコル）で通信を提供し得る。たとえば通信は、図示しない無線周波数トランシーバを通じて生じ得る。さらに、狭域通信は、たとえばＢＬＵＥＴＯＯＴＨ（登録商標）、Ｗｉ－Ｆｉ（登録商標）、または他のそうしたトランシーバが用いられて生じ得る。

【００２７】

いくつかの態様において、ネットワーク環境１００は、複数のネットワーク（たとえば、ネットワーク１０５）に広がる分散型クライアント／サーバ・システムであり得る。ネットワーク１０５は、大型のコンピュータ・ネットワーク（たとえば、ローカル・エリア・ネットワーク（ＬＡＮ）、ワイド・エリア・ネットワーク（ＷＡＮ）、インターネット、セルラー・ネットワーク、または、任意の数のモバイル・クライアント、固定クライアント、およびサーバを接続するそれらの組み合わせ）であり得る。さらに、ネットワーク１０５は、バス・ネットワーク、スター・ネットワーク、リング・ネットワーク、メッシュ・ネットワーク、スター－バス・ネットワーク、ツリーまたは階層型ネットワークなどを含むネットワーク・トポロジーのうちの任意の１以上を含むが、これらに限定されない。いくつかの態様において、各クライアント（たとえば、クライアント・デバイス１０１ａおよび１０１ｎ）とサーバ１０９ａまたは１０９ｍとの間の通信は、バーチャル・プライベート・ネットワーク（ＶＰＮ）、セキュア・シェル（ＳＳＨ）・トンネル、または他のセキュア・ネットワーク接続を介して生じ得る。いくつかの態様において、ネットワーク１０５はさらに、コーポレート・ネットワーク（たとえばイントラネット）と、１以上の無線アクセスポイントとを含み得る。

【００２８】

例示的態様において、クライアント・デバイス１０１ａまたは１０１ｎはどちらも互いに通信可能であって、ネットワーク・プロトコル（たとえば、ハイパーテキスト・トランスファー・プロトコル（ＨＴＴＰ）プロトコル）を用いてサーバ１０９ａまたは１０９とはネットワーク１０５を介して通信可能である。クライアント・デバイス１０１ａまたは１０１ｎは、クライアント・デバイス１０１ａまたは１０１ｎのユーザ・インタフェース内においてアプリケーション１０７ａまたは１０７ｎ（たとえば、ウェブ・ブラウザ）用のユーザ入力を受信する。ユーザ入力は、どのコンテンツをロードするのかを指定することができる。たとえば、ユーザ入力は、どのコンテンツがサーバ１０９ａまたは１０９ｍからロードされるかを指定するウェブサイトに関連付けられているユニフォーム・リソース・ロケータ（ＵＲＬ）アドレスであり得る。さらにユーザは、入力を行うことで、ユーザがクライアント・デバイス１０１ａまたは１０１ｎ（図１に示されている）のユーザ・インタフェース（ＵＩ）上で、ウェブ・ページのどの部分をユーザが見たいか指定し得る

。そのような例示的態様において、クライアント・デバイス１０１ａまたは１０１ｎの各々は、ユーザからの要求に基づくキャッシュされた実行可能コード用のキャッシュを確認することができ、クライアント・デバイス１０１ａまたは１０１ｎ上での実行用にキャッシュされた実行可能コードを提供することができるコード・キャッシング・プラットフォーム１０３または１０３ｎを含み得る。

【００２９】

図２は、コード・キャッシングが提供され得るクライアント・デバイスを示す。クライアント・デバイス２００は、図１のクライアント・デバイス１０１ａおよび１０１ｎに類似している。図示する通り、クライアント・デバイス２００は、プロセッサ２０１、ネットワーク・インターフェース２０３、およびメモリ２０５を含む。プロセッサ２０１は、コンピュータ可読媒体（たとえば、メモリ２０５）内に記憶されたコンピュータ命令を実行するよう構成されている。プロセッサ２０１は、中央処理装置（ＣＰＵ）であり得る。ネットワーク・インターフェース２０３は、クライアント・デバイス２０１がネットワーク１０５内（たとえば、インターネット、イントラネット、ローカル・エリア・ネットワーク、またはセルラー・ネットワーク）内のデータを送受信可能になるよう構成されている。ネットワーク・インターフェース２０３は、ネットワーク・インターフェース・カード（ＮＩＣ）を含み得る。

【００３０】

メモリ０５は、データおよび命令を記憶する。図示する通り、メモリ２０５は、図１のアプリケーション１０７ａおよび１０７ｎに類似するアプリケーション２０７と、図１のコード・キャッシング・プラットフォーム１０３ａおよび１０３ｎに類似するコード・キャッシング・プラットフォーム２０９を有する。アプリケーション２０７とコード・キャッシング・プラットフォーム２０９とは、実行環境（たとえば、仮想マシン２１１）内で機能し得る。

【００３１】

アプリケーション２０７は、プロセッサ２０１によってクライアント・デバイス００上で実行されるソフトウェア・アプリケーションであり得る。たとえば、アプリケーション２０７は、ネットワーク１０５を通じてサーバ１０９ａまたは１０９ｍによって提供されるサービスへのアクセスをクライアント・デバイス２００のユーザに対して提供するウェブ・ブラウザであり得る。サーバ１０９ａおよび１０９ｍは、ウェブサイトの形式においてクライアント・デバイス２００に対してサービスを提供する可能性があり、サーバ１０９ａまたは１０９ｍによって提供されるウェブ・ページは、クライアント・デバイス２００においてロードされる際にコンパイルおよび実行される複数のソース・コードまたはスクリプト（たとえば、ＪａｖａＳｃｒｉｐｔ）を含み得る。

【００３２】

コード・キャッシング・プラットフォーム２０９は、アプリケーション２０７に対するコード・キャッシング・サービスを提供するよう構成され得る。アプリケーション２０７において、アプリケーション２０７は、サーバ１０９ａおよび１０９ｍによってクライアント・デバイス２００に対して提供されるサービスをレンダリングする。いくつかの態様において、コード・キャッシング・プラットフォーム２０９は、クライアント・デバイス２００上での実行を待機するアプリケーション２０７に埋め込まれたソース・コードの表示を受信する。ソース・コードの表示は、メモリ２０５中のアプリケーション２０７内に埋め込まれたソース・コードの位置を指すリンクであり得る。該表示は、ソース・コードに関連付けられている識別子、または、ソース・コードのフル・コンテンツでもあり得る。該表示を受信することに応じて、コード・キャッシング・プラットフォーム２０９は、ソース・コードに対応するキャッシュされた実行可能コード用のメモリ２０５内でリソース・キャッシュ２１３を確認することが可能である。リソース・キャッシュ２１３は、永続性／不揮発性メモリ内に記憶され得る。

【００３３】

たとえば、ソース・コードまたはソース・コードの一部（たとえば、サブコード、機能

、など）は、以前にリソース・キャッシュ２１３内にコンパイルおよびキャッシュされていた可能性がある。二次ソース・コードとも称されるサブ・コードは、ソース・コード内で参照される（たとえば、呼び出される）機能であり得る。二次ソース・コードは、様々な条件（たとえば、ユーザ入力、他の二次ソース・コードからの出力など）に基づいて、ソース・コード内で参照され得る。さらに、ソース・コードに関連するデータ（たとえば、タイプ情報）は、キャッシュされていた可能性がある。実行可能コードをより効率的に生成するために、キャッシュ・データはコンパイラによって用いられる。たとえば、コード・キャッシング・プラットフォーム２０９は、ソース・コードを実行可能コードへとコンパイルするためクライアント・デバイス２００内のコンパイラ（図２に示されていない）に送ることによって、ソース・コードからコンパイルされた実行可能コードを取得し、クライアント・デバイス２００上の実行用に実行可能コードを提供し得る。ソース・コードに対応するキャッシュされた実行可能コードまたはソース・コード内で参照された二次ソース・コードがリソース・キャッシュ２１３内で発見される場合、コード・キャッシング・プラットフォーム２０９は、コンパイルされるソース・コード内で参照される二次ソース・コードを選択し得る。コード・キャッシング・プラットフォーム２０９は、ソース・コードに対応する実行可能コードおよび／または二次ソース・コードに対応する実行可能コードをキャッシュし得る。しかしながら、ソース・コードに対応するキャッシュされた実行可能コードがリソース・キャッシュ２１３内で発見されると、キャッシュされた実行可能コードは、ソース・キャッシュから取り出され、ソース・コードのコンパイルの必要がなく実行される。

【００３４】

実行可能コードは、１組の参照を含み得る。実行可能コードに含まれる参照は、たとえば、実行可能コードが実行されている実行環境（たとえば、仮想マシン２１１）に対応する実行可能コードに埋め込まれているメモリ・アドレスであり得る。埋込は、１つの実行環境から別の実行環境へはポーティング不可能である場合がある。しかしながら、埋込アドレスを抽象化することによって生成されたリロケーション・データは、複数の実行コンテキストにわたってである。リロケーション・データは、たとえば、ポーティング後にノン・ポータブル・アドレスを再記憶することによって、実行可能コード内の埋込アドレスの抽象化を実装するために用いられる。ソース・コード用の実行可能コードおよび選択された二次ソース・コードを取得すると、コード・キャッシング・プラットフォーム２０９は、実行可能コードをシリアライズする。シリアライズすることは、実行コンテキスト（たとえば、仮想マシン２１１）から実行可能コード内の１組の参照を抽象化することを含む。実行コンテキストは、コンピュータ・システム内の操作環境に対して、実行可能コードを実行するための電力およびメモリへのアクセスを提供する。コード・キャッシング・プラットフォーム２０９は、実行可能コードに関連付けられているリロケーション・データを生成し、抽象化された参照のさらなる変換が生成されたリロケーション・データに基づいて行われるように、抽象化を反映する。

【００３５】

コード・キャッシング・プラットフォーム２０９は、ソース・コード用のシリアライズ・コードと、ソース・コード内で参照された選択された二次ソース・コードとを、メモリ２０５におけるリソース・キャッシュ２１３内のキャッシュ・データとして記憶する。従来のキャッシュ・マネジメント・ストラテジーは、リソース・キャッシュを管理するために用いられ得る。さらに、キャッシュ・タグは、リソース・キャッシュ内のキャッシュ・データ位置を識別するために用いられ得る。さらに、コード・キャッシング・プラットフォーム２０９は、仮想マシン２１１における実行用の実行可能コードを提供できる。いくつかの態様において、仮想マシン２１１におけるシリアライズ・コードの記憶の後に、コード・キャッシング・プラットフォーム２０９は、異なる仮想マシン環境において、実行を待機するソース・コードの二次表示を受信し得る。たとえば、上述のような実行可能コードの実行後、クライアント・デバイス２００は、再始動または再起動されていた可能性がある。上述の通り、表示に応じて、コード・キャッシング・プラットフォーム２０９は

、ソース・コードに対応するキャッシュされた実行可能コードまたはソース・コード内で参照された二次ソース・コード用のメモリ２０５内でリソース・キャッシュ２１３を確認できる。二次ソース・コードが以前にリソース・キャッシュ２１３内でコンパイルおよびキャッシュされていたので、二次ソース・コードに対応するキャッシュされた実行可能コードは、リソース・キャッシュ２１３内で発見される。リソース・キャッシュ２１３内におけるキャッシュ・ヒットの際、コード・キャッシング・プラットフォーム２０９は、キャッシュ・データをメモリ２０５内のリソース・キャッシュ２１３から取り出す。

【００３６】

キャッシュ・コードは、以前のコンパイル時に二次ソース・コードから抽象化された参照に対応するシリアライズ・コードを含む。コード・キャッシング・プラットフォーム２０９は、取り出されたシリアライズ・コードから、新しい実行可能な二次ソース・コードへとデシリアライズすることができる。デシリアライズ・プロセスは、イニシャル・コードと、イニシャル・コードによって参照されるオブジェクトのタイプと、オブジェクト内に記憶されたデータのタイプとに関するシリアライズ・データを伴って情報（たとえば、リロケーション・データ）を用いて、一連の記憶されたデータからコードを抽出する。デシリアライズを通じて生成される実行可能コードは、イニシャル実行可能コードと類似の構造、特徴、およびビヘイビアを有する。したがって、コンピュータによる実行可能コードと該実行可能コードによって参照されたオブジェクトとの実行によって、イニシャル・コードの実行によって行われるのと同一のタスクが行われる。コードによって参照されたコードとオブジェクトとは両方とも、シリアライズおよびデシリアライズの対象となる。デシリアライズは、取り出されたシリアライズ・コードを用いて、新しい実行可能二次ソース・コード内の各二次ソース・コードからの１組の参照を、該新しい実行可能二次ソース・コードが再始動の際にクライアント・デバイス２００上に構築された新しい仮想マシン環境内で実行可能となるように、再記憶することを含み得る。コード・キャッシング・プラットフォーム２０９は、その後、クライアント・デバイス２００上における新しい仮想マシン環境内での実行用の新しい実行可能二次ソース・コードを提供可能である。新しい実行可能二次ソース・コードは、キャッシュされたことがなく、この実行用にコンパイルされているソース・コード内で参照される他の二次ソース・コードとともに実行され得る。コード・キャッシング・プラットフォーム２０９は、ソース・コードの様々な実行におけるキャッシング用の新しい二次ソース・コードを選択し得る。さらに、コード・キャッシング・プラットフォーム２０９は、以前選択およびキャッシュされた二次ソース・コードを、ソース・コードの二次ソース・コードの実行履歴に基づいて、他の、たとえば、より頻繁に実行される二次ソース・コードに置き換え得る。

【００３７】

前述の通り、コード・キャッシング・プラットフォーム２０９によるコード・キャッシングは、ネットワーク１０５を介してのサーバ１０９ａまたは１０９ｂとの通信をせずに、クライアント・デバイス１０１ａまたは１０１ｂ内で行われる。たとえば、例示的態様において、コード・キャッシング・プラットフォーム２０９によってシリアライズおよびデシリアライズされているソース・コードは、クライアント・デバイス１０１ａまたは１０１ｎのメモリ上で局所的に記憶されたコードであり得る。

【００３８】

図３は、スクリプトのキャッシュ・コードへのシリアライズとキャッシュ・コードの実行可能コードへのデシリアライズとが提供され得るコード・キャッシング・プラットフォームのフロー図である。図３に示す通り、コード・キャッシング・プラットフォーム２０９は、フロント・エンド３０３と、シリアライジング・モジュール３０９と、デシリアライジング・モジュール３１９とを備える。フロント・エンド３０３は、確認モジュール３０５と、圧縮モジュール３０７とを備える。フロント・エンド３０３は、ソース・コード３０１を受信する。ソース・コードを受信すると、確認モジュール３０５は、ソース・コードに対応するキャッシュされたシリアライズ・コード用のデータ・ストア３１３を確認する（矢印３１１で図示）。シリアライズ・コードがデータ・ストア３１３内で発見され

ない場合（キャッシュ・ミス）、フロント・エンド３０３はコンパイラ３２７にソース・コードを送る（矢印３２５で図示）。コンパイラ３２７は、ソース・コードをコンパイルし、実行可能コードおよび参照を記憶する。コンパイラ３２７は、実行環境３３１（図３に示さず）に関連付けられているメモリ位置内の実行可能コードおよび参照を記憶し得る。実行可能コードは、実行環境３３１内での実行のために実行環境３３１に送られる（矢印３２９で図示）。実行環境３３１は、たとえば、ブラウザ・セッション内における図２の仮想マシン２１１に類似する仮想マシンである。

【００３９】

コンパイラ３２７は、実行可能コードおよび参照をシリアライズ用にシリアライジング・モジュール３０９に送る（３３５に図示）。フロント・エンド３０３は、たとえば、過去の実行における部分の実行の頻度に基づいて、または、ソース・コード内の該部分を参照する機能呼出の頻度に基づいて、シリアライズ用のソース・コード内で参照された部分（たとえば、二次ソース・コード）を選択し得る。コード・キャッシング・プラットフォーム２０９は、様々な基準（たとえば、ヒューリスティクス）に基づいて、シリアライズおよび／またはキャッシング用のソース・コード内で参照されたソース・コードまたは二次ソース・コードを選択できる。いくつかの例示的基準は、ソース・コードまたは二次ソース・コードのサイズ、ソース・コードまたは二次ソース・コードの寿命（たとえば、ソース・コードまたは二次ソース・コードの最終改定日または満了日に基づく）、ソース・コードまたは二次ソース・コードの使用頻度（たとえば、ｎ番目のエンカウンタ後にキャッシュする）、リソース・キャッシュ２１３内でキャッシュ・コードが置き換えられる頻度、二次ソース・コードが一次ソース・コード内で参照される回数、または、二次ソース・コードのコンパイル時間であり得る。コード・キャッシング・プラットフォーム２０９は、二次ソース・コードがソース・コード内で参照されるという予測に基づいて、シリアライズおよびキャッシング用のソース・コード内で参照された二次ソース・コードを選択し得る。たとえば、コード・キャッシング・プラットフォーム２０９は、ソース・コードがコンパイルされているとき、二次ソース・コードがソース・コード内で参照されることを予測し得る。ソース・コードからのコンパイル結果は、二次ソース・コードが実行中にソース・コードによって参照されるであろうことを示し得る。

【００４０】

コード・キャッシング・プラットフォーム２０９は、ソース・コードが閾値Ｓよりも小さなサイズである場合、ソース・コードをシリアライズし得る。その論理的根拠は、コードの大部分をシリアライズすることは、キャッシュ・データを記憶するためには、長時間を要する可能性があり（たとえば、クライアント・デバイス１０１ａ～１０１ｎ上の利用可能ディスク・スペースに基づいて判定される）、メモリの高容量を必要とする可能性がある、ということである。

【００４１】

他の例示的基準は、キャッシュ・アクセス・レイテンシ（たとえば、キャッシュ・エントリ・サイズ）、ソース・コードまたはソース・コードの一部のコンパイル時間（たとえば、ソース・コードのイニシャル・コンパイルにどれほど時間がかかるのかを記録し、その時間を判定の考慮に入れる）、リソース・キャッシュ２１３をホストするクライアント・デバイス１０１ａ～１０１ｎのメモリ容量、ソース・コード更新の頻度（たとえば、ソース・コードが頻繁に更新されている場合、ソース・コードをキャッシュすることは有益でない可能性がある）などであり得る。コード・キャッシング・プラットフォーム２０９は、シリアライズするコードの選択の前に、上記要因間におけるトレード・オフを算出し得る。たとえば、重量要因は、上述の様々な基準の重要性のレベルを判定するために、ソース・コードまたはソース・コードの一部用に規定される。トレード・オフの算出は、たとえば、規定された重量要因に基づいて行われる。たとえば、クライアント・デバイス１０１ａ～１０１ｎのコード・サイズとメモリ容量との間におけるトレード・オフにより、コードが頻繁に呼び出されていても、クライアント・デバイス１０１ａ～１０１ｎ上の大幅なメモリ・スペースを占め得る大きいサイズであるコードの上記キャッシングをもたら

し得る。

【００４２】

様々な例において、フロント・エンドは、実行可能コードおよび参照をシリアライズするかどうかに関する判定を下すことができる。たとえば、いくつかの基準は、シリアライズされるソース・コードまたはソース・コードの一部のタイプと、シリアライズされずにコンパイルおよび実行されるタイプに関して、サーバ１０９ａまたは１０９ｍによって規定され得る。さらに、ソース・コードの一定のタイプは、セキュリティの目的におけるソース・コードの実行毎の前に照合される必要があり得る。そのような場合、フロント・エンド３０３は、ソース・コードまたはソース・コードの一部に、シリアライズされるべきでないソース・コードとして、警告を与え得る。

【００４３】

他のコード・キャッシング基準は、所定の回数より多くコンパイルされるコードへのコード・キャッシングの適用も含み得る。たとえば、キャッシュ・ミスが初めて検知されると、ソース・コードはコンパイルされるが、シリアライズまたはキャッシュはされない。ソース・コードは、実行されているコードのシリアライズなしで、「一度コンパイルしたもの」としてマークされる。同様に、ソース・コードは、所定の閾値Ｃに達するまでにソース・コードがコンパイルされた回数を示すカウンタを伴ってマークされる可能性があり、Ｃ番目のキャッシュ・ミスに際し、コード・キャッシング・プラットフォーム２０９はソース・コードをシリアライズおよびキャッシュすることができる。それに代えて、ジェネリック・マーカでなく、タイムスタンプが１番目のキャッシュ・ミス上に設定されてもよく、１番目のキャッシュ・ミスとＣ番目のキャッシュ・ミスとの間における時間差を、シリアライズの判定を行う際に用いられてもよい。基準は、たとえば、「コンパイル・コードが少なくとも週にＣ回実行される場合、コンパイル・コードはキャッシュされる」に変換される。ソース・コードをキャッシュするための判定についての他の基準は、たとえば、コンパイル時間、生成されたコードのサイズ、ＨＴＴＰキャッシュ・ヘッダ・データなどが考えられる。

【００４４】

コード・キャッシング・プラットフォーム２０９は、後続の実行でさらにキャッシュされて用いられる二次ソース・コードを選択するために、ソース・コードの実行データを用い得る。実行データは、たとえば、実行されるコードが実行中に参照するコードの他の部分、ソース・コードの実行プロファイル、キャッシングに価値をもたせるべくソース・コードが頻繁に実行されるか（たとえば、所定の回数を超える回数）、実行中に観察されるタイプ（たとえば、ユーザ規定のテータ構造）情報などを含む。さらに、実行データは、該データが実行コンテキスト内でデシリアライズされるようにシリアライズされ得る。デシリアライズ後、実行データがデシリアライズ・コードとともに用いられ得る。たとえば、実行プロファイルは、実行可能コードを最適化する時間を費やすか否かについて判定するために、実行コンテキスト内で用いられ得る。

【００４５】

動的型付け言語（たとえば、ＪａｖａＳｃｒｉｐｔ）は、ソース・コード内に規定されるオブジェクトに関する多数のタイプ情報を提供する。動的型によって、ソース・コードのオブジェクトは、ソース・コードの様々な実行中に様々なタイプを有することができる。しかしながら、ソース・コードの様々な実行を観察することによって、オブジェクトのタイプがどれほど頻繁に変化するかに関する洞察が得られる可能性がある。

【００４６】

特定のオブジェクトＯのタイプＴは、他のコンテンツに基づいて判定される。しかしながら、ソース・コードがオブジェクトＯと対話すると、ソース・コードはタイプ情報へのアクセスを有さない。したがって、オブジェクト・タイプＴはソース・コードにとって未知である。結果として、ソース・コードは、オブジェクトＯにアクセスする前に、タイプＴを判定する必要がある。いくつかの態様において、コード・キャッシング・プラットフォーム２０９は、オブジェクト・タイプＴを判定およびキャッシュする。たとえば、コー

ド・キャッシング・プラットフォーム２０９は、オブジェクトＯが実行環境内（たとえば、ウェブサイト内）でアクセスされるとき、オブジェクトの特定のタイプをソース・コード内で観察し得る。コード・キャッシング・プラットフォーム２０９は、アクセス・ウェブサイトを、ウェブサイトによってアクセスされたオブジェクトのタイプＴに関連付けることができる。タイプＴとともに、コード・キャッシング・プラットフォーム２０９はさらに、アクセス時間において判定された「どのようにアクセスするか」の情報をキャッシュすることができる。その後、同一の実行環境においてソース・コードによってオブジェクトＯがアクセスされるとき、予期されるタイプＴとオブジェクトＯにアクセスするための方法とはキャッシュ・データから既知である。また、コード・キャッシング・プラットフォーム２０９は、予期されるタイプＴが現在のアクセスにおけるオブジェクトのタイプと同一であるかを確認することができる。タイプが合致する場合、キャッシュされたアクセス方法は、必要とされるアクセス方法のさらなる判定なしにオブジェクトＯにアクセスするために用いられる。

【００４７】

さらに、オブジェクト・タイプＴは、実行環境内のオブジェクトＯにアクセスするためのアクセス方法に関する情報を提供することができる。オブジェクトＯにアクセスするためのアクセス方法が判定されると、シリアライズ中に、アクセス方法に関する情報はリソース・キャッシュ内にタイプ情報として記憶され、アクセス方法に関連付けられる。記憶されたタイプ情報を用いることで、デシリアライズの際、オブジェクトＯへの後続のアクセスを、アクセス方法を判定する必要なしに高速化することができる。

【００４８】

シリアライジング・モジュール３０９によるシリアライズは、実行可能コード内のアドレスを抽象アドレスと置き換えることによって、実行可能コード内の参照を抽象化することができる。シリアライジング・モジュール３０９は、実行可能コードに対応するシリアライズ・コードと、フロント・エンド３０３への実行可能コード内の参照の抽象化に対応するリロケーション・データとを提供する（３２３に図示）。シリアライズ・コードを提供する際、シリアライジング・モジュール３０９は、実行可能コードに対応する関連部分および参照と、実行環境３３１からの参照とを、発見することができる。シリアライジング・モジュール３０９は、その部分と参照とを、シリアライズ・コードに含まれるよう、実行環境３３１から独立して隣接バイナリ表現に変換することができる。フロント・エンド３０３は、シリアライジング・モジュール３０９によって生成されるシリアライズ・コードをデータ・ストア３１３内に、キャッシュされたシリアライズ・コードとして記憶することができる。シリアライジング・モジュール３０９は、抽象化に対応するシリアライズ・コードを、そのコードをシリアライズ・コード内の他のデータ・タイプから差別化するタグを用いて、符号化し得る。

【００４９】

他の例において、シリアライズ・コードは、フロント・エンド３０３によってデータ・ストア３１３内およびメタデータ３１５中のリロケーション・データ内に記憶された後、フロント・エンド３０３は、異なる実行環境３３３（たとえば、新しいブラウザ・セッション内の新しいＶＭ）内のソース・コードの実行用に新しい要求を受信し得る。この例において、確認モジュール３０５は、データ・ストア３１３内のキャッシュされたシリアライズ・コードを発見することに成功し得る（キャッシュ・ヒット）。キャッシュ・ヒットに際して、フロント・エンド３０３は、キャッシュされたシリアライズ・コードをデータ・ストア３１３からロードすることができ（３１７で図示）、デシリアライジング・モジュール３１９にロードされたキャッシュされたシリアライズ・コードを送ることができる。キャッシュされたシリアライズ・コードをロードする前に、フロント・エンド３０３は、キャッシュされたシリアライズ・コードがソース・コードの同一バージョンに対応するかを判定できる。バージョンが合致しない場合、フロント・エンド３０３は、キャッシュされたシリアライズ・コードをデシリアライズしないことを判定し、コンパイルされるソース・コード用のコンパイラ３２７に要求を送ることができる。フロント・エンド３０３

はさらに、メモリ・スペースを確保するために、旧式のキャッシュ・コードを退避させ得る。フロント・エンド３０３は、ネットワーク１０５（たとえば、ハイパーテキスト・トランスファー・プロトコル（ＨＴＴＰ））内の通信プロトコルによって提供される標準バージョン確認プロセスに基づいて、キャッシュされたシリアライズ・コードとソース・コードとのバージョンを判定し得る。

【００５０】

キャッシュされたシリアライズ・コードをデータ・ストア３１３から取得すると、デシリアライジング・モジュール３１９は、キャッシュされたシリアライズ・コードの妥当性を照合して、そのキャッシュされたシリアライズ・コードを、シリアライズ・コード内のリロケーション・データを用いて、実行可能コードおよび参照にデシリアライズする。デシリアライジング・モジュール３１９は、実行可能コードおよび参照を実行のために実行環境３３１に送る（３２１で図示）。シリアライズ・コードがソース・コードの一部に対応する場合、キャッシュされていない該ソース・コードの一部はコンパイラ３２７によって実行可能コードへとコンパイラされ、実行環境３３１内で実行される前に、実行可能コードはデシリアライズされた実行可能コードと組み合わされる。

【００５１】

なお、シリアライジング・モジュール３０９とデシリアライジング・モジュール３１９とは、異なる実行環境３３１および３３３内でアクティベートされ得る。シリアライズ・コードは、（実行可能コードおよび参照が生成および実行される）実行環境３３１内で生成される。一方、デシリアライジング・モジュール３１９によって提供される実行可能コードおよび参照は、後続の実行環境３３３内で生成される。

【００５２】

データ・ストア３１３内のキャッシュ・データのサイズを削減するために、圧縮モジュール３０７は、たとえば、データ・ストア３１３内のシリアライズ・コードを記憶する前に、シリアライズ・コード上の様々な圧縮アルゴリズムを適用することによって、データを圧縮し得る。圧縮モジュール３０７はさらに、キャッシュされたシリアライズ・コードをデシリアライジング・モジュール３１９に送る前に、キャッシュされたシリアライズ・コードを解凍し得る。

【００５３】

キャッシュされたシリアライズ・コードは、異なる実行環境において、実行可能コードおよび参照のコンテキスト独立全表現として用いられる。たとえば、実行環境３３３用の実行可能コードおよび参照を生成しているとき、デシリアライジング・モジュール３１９は、実行環境３３１用の実行可能コードおよび参照を生成する際のコンパイラ３２７のビヘイビアを模倣することができる。したがって、キャッシュされたシリアライズ・コードの実行可能コードおよび参照へのデシリアライズ後、実行環境３３３内の実行可能コードおよび参照の実行によって、実行環境３３１内のコンパイラ３２７によって提供される実行可能コードおよび参照の実行と同等の結果を得ることができる。

【００５４】

デシリアライジング・モジュール３１９によるデシリアライズのプロセスは、メタデータ３１５内のリロケーション・データを用いて参照を再記憶することによって、該参照が実行環境３３１と実行環境３３３との間における差異を反映することができる。たとえば、上述の通り、シリアライズとデシリアライズとが異なる実行環境（たとえば、ＶＭの例）内で動作し得るため、実行環境３３１に対応する実行可能コードおよび参照のメモリ・アドレスは、実行環境３３３内では有効でない可能性がある。しかしながら、シリアライズとデシリアライズとによって、デシリアライズされた実行可能コードおよび参照内のメモリ・アドレスが実行環境３３３内で有効であることが確実になる。たとえば、実行環境内の機能アドレスへの参照は、デシリアライジング・モジュール３１９によって文字通りに符号化されない。これは、それらのアドレスが異なる実行環境３３１および３３３内では異なるためである。

【００５５】

シリアライジング・モジュール３０９は、ソース・コード内と実行可能コードおよび参照内との各オブジェクトを精査し、各オブジェクトがどのようにシリアライズされるべきかを判定し得る。たとえば、一定の固有オブジェクト（たとえば、正規不定値）は、抽象インデックス（たとえば、ルート・アレイ）内の実行可能コードおよび参照に含まれ得る。そのような固有オブジェクトがシリアライジング・モジュール３０９によってシリアライズされると、シリアライジング・モジュール３０９は、固有抽象インデックスによって該オブジェクトをシリアライズする。デシリアライジング・モジュール３１９によって固有オブジェクトがデシリアライズされると、該インデックスはオブジェクトを識別するために用いられる。

【００５６】

さらに、ソース・コードは、エレメンタリ・タスクを行う実行可能コードの生成された部分であるビルトイン・コードを含み得る。ビルトイン・コードは、ソース・コード内の複数のオブジェクトによって共有され得るので、実行環境３３１（または３３３）内で常に利用可能であるべきである。シリアライジング・モジュール３０９は、各ビルトイン・コードをビルトイン識別子へとマッピングする可能性があり、該ビルトイン・コードをシリアライズ・コード内のビルトイン・コードの表現としてシリアライズする。デシリアライズの際、フロント・エンド３０３は、実行環境３３３内において、同一の識別子とともにビルトイン・コードを発見することができる。

【００５７】

同様に、ソース・コードは、コード・スタブを含み得る。コード・スタブは、オンデマンドで生成され実行環境３３３上で必ずしも利用可能でないことを除けば、ビルトイン・コードと類似する。シリアライジング・モジュール３０９は、シリアライズ・コードを生成する際、コード・スタブをコード・スタブ・キーへとマッピング可能である。デシリアライズの際、フロント・エンド３０３は、実行環境３３３がコード・スタブを含むか確認し得る。実行環境３３３がコード・スタブを含まない場合、コード・スタブ・キーに対応するコード・スタブは、デシリアライジング・モジュール３１９によって生成される。

【００５８】

さらに、ソース・コードは、文字列リテラルを含み得る。固有文字列リテラルは、シリアライジング・モジュール３０９によって、文字通りにシリアライズされる。デシリアライズの際、フロント・エンド３０３は、同一の文字列リテラルが実行環境３３３内に既に存在するかを確認する。固有文字列リテラルが実行環境３３３内に存在する場合、デシリアライジング・モジュール３１９は文字列リテラルを既に存在している固有文字列リテラルへと正規化する。正規化プロセスは、２つ以上の表現（たとえば、１つの表現はキャッシュされたシリアライズ・コード内、２番目の表現は実行環境３３３内）とともに、文字列リテラルを標準または正規の形式へと変換する。

【００５９】

いくつかの例において、シリアライズされたオブジェクトは、ソース・コードのソース文字列への参照を有する。そのような例において、シリアライジング・モジュール３０９は、オブジェクトを特別な表現に置き換えることができる。デシリアライズの際、デシリアライジング・モジュール３１９は、特別なコードを、実行環境３３３内でのデシリアライズ時において個別に提供されるソース文字列への参照に置き換えることができる。

【００６０】

上述のオブジェクト・アドレスのシリアライズに加えて、実行可能コードに埋め込まれたアドレスも実行環境内の一定値のアドレスまたはＶＭの機能のアドレスであり得る。実行環境内の一定値のアドレスまたはＶＭの機能のアドレスは、コンパイラとＶＭとにとって既知である。これらのアドレスは、たとえば、参照識別子として表現され、該参照識別子はデシリアライズ時にアドレスを再記憶するために用いられる。さらに、ソース・コードは、非固有オブジェクトを含み得る。非固有オブジェクトは、コピーとしてシリアライズされる。たとえば、非固有オブジェクトは、データ・フィールドと他のオブジェクトへの参照とで構成され得る。シリアライズ時、データ・フィールドはコピーされ、他のオブ

ジェクトへの参照は、固有である場合、固有オブジェクトに関して前述されたようにシリアライズされる。

【００６１】

以下の例は、コンパイルからサンプル・コードのシリアライズおよびデシリアライズへのコード・キャッシング・プロセスを説明している。サンプルのＪａｖａＳｃｒｉｐｔコード（Ａ）が本例において検討される。

【００６２】

【表１】

```
function foo ( ) { return 1; }
foo ( );
```
(A)

【００６３】

サンプル・コード（Ａ）用の実行可能コードは、たとえば、機能ヘッダ、スタック・オーバーフローに対しての確認命令（呼出スタック・オーバーフロー・ビルトイン）、変数宣言「ｆｏｏ」、内部関数（）｛ｒｅｔｕｒｎ　１；｝用のクロージャのインスタンス化、クロージャを「ｆｏｏ」という名称のグローバル・プロパティへ割り当てること、「ｆｏｏ」という名称のグローバル・プロパティをロードすること、ロードされたプロパティを呼び出すこと、不定値に戻ることなどの命令を含む。

【００６４】

さらに、サンプル・コード（Ａ）用の実行可能コードは、リロケーション情報への参照を含み得る。リロケーション情報は、スタック・オーバーフロー・ビルトインはビルトインであること、文字列リテラル「ｆｏｏ」はオンヒープ・オブジェクトであること、（）｛ｒｅｔｕｒｎ　１；｝用の機能記述はオンヒープ・オブジェクトであること、機能クロージャ初期化は実行時呼出として実装されて、それ自体が外部参照であるということ、プロパティをロードすることはスタブに対する呼出であること、プロパティを呼び出すことはスタブに対する呼出であること、不定値はオンヒープ・オブジェクト（たとえば、図３に関して上述の固有オブジェクト）であり得ること、を示し得る。

【００６５】

また、サンプル・コード（Ａ）用の実行可能コードは、文字列リテラル「ｆｏｏ」に関連付けられているメタデータ（たとえば、「ｆｏｏ」がマップ・オブジェクトによって示される「内在文字列」というタイプのオブジェクトであること、文字列コンテンツが「ｆｏｏ」であること、文字列の長さが３であること）を含み得る。さらに、サンプル・コード（Ａ）用の実行可能コードは、トップ・レベル機能および内部関数（）｛ｒｅｔｕｒｎ　１；｝用の機能記述を含む。機能記述は、マップ・オブジェクトによって示される「共有機能情報」というタイプのオブジェクトであり得る。機能記述は、ソース・コード内の始点と終点とに関する文字位置を含み得る。さらに、機能記述は、サンプル・コード（Ａ）が既にコンパイルされていた場合、コンパイル・コードを指し得る。

【００６６】

当初は、内部関数「ｆｏｏ」はレイジーにコンパイルされる。レイジー・コンパイルとは、内部関数「ｆｏｏ」はトップ・レベル・コードがコンパイルされる際にはコンパイルされず、トップ・レベル・コードのコンパイル中においてのみ内部関数「ｆｏｏ」が生成される。内部関数「ｆｏｏ」が最終的に呼び出されるとき（たとえば、トップ・レベル・コードの実行中）、内部関数「ｆｏｏ」はオンデマンドにコンパイルされる。しかしながら、内部関数「ｆｏｏ」がコンパイルされない限り、内部関数「ｆｏｏ」用の実行可能コードは存在せず、内部関数用の実行可能コードはシリアライズされない。

【００６７】

コード・キャッシング・プラットフォーム２０９がキャッシュ・データ内の内部関数「ｆｏｏ」用のコンパイル・コードを含むようにする様々な方法がある。コード・キャッシング・プラットフォーム２０９は、トップ・レベル・コード内で内部関数「ｆｏｏ」が呼

び出される（「ｆｏｏ」がサンプル・コード（Ａ）内で規定され、呼び出される）ことを認識し得る。この認識を以て、コード・キャッシング・プラットフォーム２０９は、トップ・レベル・コードのコンパイルにおいてトップ・レベル・コードをコンパイルすることの一部として内部関数「ｆｏｏ」をしきりにコンパイルする可能性があり、トップ・レベル・コードをコンパイルした後、シリアライズ用のコンパイル・コードを用いた。

【００６８】

シリアライズ・コード内の内部関数「ｆｏｏ」用のコンパイル・コードを含む他の方法は、コード・キャッシング・プラットフォーム２０９にトップ・レベル・コードを少なくともシリアライズ前に実行させることである。内部関数「ｆｏｏ」はトップ・レベル・コードの実行中に呼び出されるので、「ｆｏｏ」は上述の通りレイジーにコンパイルされ、コンパイル・コードはメモリ内に記憶される。この場合、後になってトップ・レベル・コードをシリアライズする際、「ｆｏｏ」用の内部関数記述（トップ・レベル・コードの初期コンパイルの間に生成される）は、内部関数「ｆｏｏ」のレイジーにコンパイルされたコードを指す。これにより、コード・キャッシング・プラットフォーム２０９は、トップ・レベル・コードをシリアライズする際、内部関数「ｆｏｏ」をシリアライズできる。

【００６９】

シリアライジング・モジュール３０９は、オブジェクト・グラフを訪れること、参照をトラバースすること、トップ・レベル・コード用の機能記述において始動することによって、サンプル・コード（Ａ）をシリアライズすることができる。コード・キャッシング・プラットフォーム２０９は、オブジェクトが２度以上シリアライズされるように、データ・ストア３１３内で既に訪れられたオブジェクトの履歴を保持する可能性があるか、または、無限ループに陥る可能性もある。既に訪れられてシリアライズされたオブジェクトは、バック参照として表現（符号化）される。オブジェクトは、コンテンツまたは他のオブジェクトへの参照のどちらかを含み得る。オブジェクト・コンテンツは、文字通りにシリアライズされる。しかしながら、他のオブジェクトへの参照は、既知オブジェクトへの参照またはシリアライズされる必要もあるオブジェクトへの参照であってもよい。たとえば、関数記述は、「共有関数情報」マップと機能用のコンパイル・コードとに対する参照を有し得る。さらに、機能関数は、文字通りにシリアライズされる文字位置上の情報を含み得る。シリアライジング・モジュール３０９によってエンカウンタされるマップ・オブジェクトは、正規オブジェクトおよびルート・リストの一部であり得る。シリアライジング・モジュール３０９は、ルート・リスト内でマップ・オブジェクトを発見してもよく、オブジェクトを表現するためにルート・リスト・インデックスを用いてもよい。

【００７０】

トップ・レベル関数用のコード・オブジェクトは、そのマップ（コード・オブジェクト用）への参照とリロケーション情報への参照とを除き、文字通りにシリアライズされる。命令ストリームは、リロケーション情報によって記述された埋め込みポインタを含み得る。さらに、これらのポインタはシリアライズ中に訪れられる。ポインタは、デシリアライズに際して更新される必要があるため、文字通りにシリアライズされる前に零（０）の値に置き換えられる。

【００７１】

トップ・レベル関数用のコード・オブジェクトは、内部関数「ｆｏｏ」の関数記述を指す。内部関数「ｆｏｏ」がこの時点で既にコンパイルされていた場合、その関数記述は、シリアライジング・モジュール３０９がトップ・レベル機能から内部関数「ｆｏｏ」にトラバースできるようになり、トップ・レベルと内部関数との両方のコード・オブジェクトをシリアライズできるようになるように、コンパイル・コードを指す。さらに、コード・スタブはコード・スタブ・キーによって表現され、ビルトインはビルトインＩＤによって表現される。文字列コンテンツは文字通りにシリアライズされ（文字列マップは除く）、外部参照は外部参照ＩＤにマッピングされる。

【００７２】

デシリアライジング・モジュール３１９によるデシリアライズのプロセスは、オブジェ

クト・グラフを訪れることによってシリアライズが行われた順番をリトレースすることを含み得る。このトレーシングは、シリアライズ・コード表現を、デシリアライズ・コードが実行された実行環境３３３内のコード表現に逆変換するために用いられる。たとえば、シリアライズ中に不定値がルート・リスト・インデックスとして符号化された場合、デシリアライジング・モジュール３１９は、実行環境３３３内の不定値を発見するために、ルート・リスト・インデックスを用いることができる。さらに、逆参照は、既にデシリアライズされたオブジェクトへの参照に逆変換される。デシリアライズの結果は、トップ・レベル機能用の機能記述であり得る。

【００７３】

図４Ａおよび図４Ｂは、スクリプトのキャッシュ・コードへのシリアライズが提供され得るプロセスの例を示す。図４Ａおよび図４Ｂは図１、図２および図３を参照して説明されるが、主題の技術はそれらに限られず、他のクライアント・デバイスとシステムとにも適用されてもよい。図４Ａは、コードをキャッシュ・コードへとシリアライズするためのプロセスの例である。ブロック４０１において、フロント・エンド３０３は、実行を待機する一次ソース・コードの表示（たとえば、リンク、アドレス、ＵＲＬなど）を受信する。たとえば、一次ソース・コードは、現在のブラウザ・セッション内における実行用のウェブ・ページ内のスクリプトであり得る。図１のコード・キャッシング・プラットフォーム１０３ａまたは１０３ｎ（または図２のコード・キャッシング・プラットフォーム２０９）をホストするクライアント・デバイス１０１ａまたは１０１ｎのユーザは、現在のブラウザ・セッション内にＵＲＬアドレスを入力することによって、一次ソース・コードの実行を要求し得る。ＵＲＬアドレスは、埋め込まれた一次ソース・コードを含むウェブ・ページをロードするためのリンクを提供し得る。

【００７４】

ウェブ・ページをロードすると、ブラウザは一次ソース・コードの表示をフロント・エンド３０３に送り得る。第１表示を受信することに応じて、ブロック４０３において、確認モジュール３０５は、一次ソース・コードに対応するキャッシュ・データがあるかデータ・ストア３１３におけるリソース・キャッシュを確認する。たとえば、ウェブ・ページは現在のブラウザ・セッションの前の他のブラウザ・セッションにおいてロードされていた可能性があり、コード・キャッシング・プラットフォーム２０９は一次ソース・コードに対応する実行可能コードをデータ・ストア３１３内に記憶した可能性がある。データ・ストア３１３が一次ソース・コードに対応する実行可能コードを含む場合、ブロック４０３における確認はキャッシュ・ヒットに戻り得る。しかしながら、ブロック４０３における確認がヒットを発見しない場合、戻られた結果はキャッシュ・ミスになり得る（ブロック４０５に図示）。

【００７５】

ブロック４０７において、リソース・キャッシュ内のキャッシュ・ミスに際し、フロント・エンド３０３は一次ソース・コードからコンパイルされた実行可能コードを取得し得る。たとえば、フロント・エンド３０３は、コンパイル用に一次ソース・コードをコンパイラ３２７に送ることが可能である。

【００７６】

ブロック４０９において、フロント・エンド３０３は、一次ソース・コード内で参照されたソース・コードまたは二次ソース・コードの部分を選択することが可能である。たとえば、二次ソース・コードは、一次ソース・コード内で参照される機能であり得る。ソース・コードの部分の選択は、図２および図３に関して説明された様々な要因と基準（ヒューリスティクス）とに基づき得る。ブロック４１１において、フロント・エンド３０３は、二次ソース・コードからコンパイルされた実行可能コードを取得し得る。たとえば、コンパイル用に二次ソース・コードをコンパイラ３２７に送ることが可能である。いくつかの例において、フロント・エンド３０３は、一次ソース・コード内で参照される一次ソース・コードと１組の二次ソース・コードとを含むソース・コード全体を、その一部を選択せずに、コンパイラに送り得る。たとえば、キャッシュ・ミスを受信に際し、フロント・

エンド３０３は、ソース・コードおよび／または該ソース・コード内で参照される選択された二次ソース・コードをコンパイラ３２７に送り得る。

【００７７】

コンパイラ３２７は、受信された一次ソース・コードまたは二次ソース・コードをコンパイルし、実行可能コードをシリアライジング・モジュール３０９に送る。コンパイラ３２７は、クライアント・デバイス１０１ａまたは１０１ｎのメモリ内に実行可能コードを記憶し得る。実行可能コードは、１組の参照（たとえば、ソース・コードまたは選択された一部内のオブジェクトに対応するアドレスおよび識別子）を含む。実行可能コードに対応する１組の参照は、実行環境３３１に対応する実行可能コード内に埋め込まれたメモリ・アドレスを含み得る。実行環境３３１（または３３３）は、ソース・コードが実行されるウェブ・ブラウザ・セッション内のＶＭであり得る。参照は、実行環境３３１（たとえば、実行可能コード用の実行エンジン（図示せず）によって生成されるＶＭ環境）内の実行可能コードを実行するとき、実行可能コード３２９用の実行データをプロセッサ２０１に提供する。

【００７８】

コンパイラ３２７によって実行可能コードを受信すると、ブロック４１３において、シリアライジング・モジュール３０９は実行可能コードをシリアライズする。シリアライズすることは、図２および図３に関して上述されたように、実行環境３３１から実行可能コード内の１組の参照を抽象化することと、抽象化を反映するようにシリアライズ・コードに関連付けられているリロケーション・データを規定することと、を含む。シリアライジング・モジュール３０９によって実行環境３３１から１組の参照を抽象化することは、埋め込まれたメモリ・アドレスを抽象アドレスに置き換えることを含み得る。たとえば、埋込アドレスは、ＶＭの機能に対し、所定リストの機能内における特定のインデックスに変換される。いくつかの例において、実行可能コードが同一コードを指すアドレスを含む場合、メモリ・アドレスを実行可能コードに割り当てられたメモリのブロックの開始アドレスからのオフセット・アドレスに置き換えることによって、抽象アドレスが生成される。そして、埋込アドレスは、コードの開始からの関連オフセットに変換され得る。

【００７９】

ブロック４１５において、フロント・エンド３０３は、シリアライズ・コードを、実行環境３３１内の実行用に提供されるキャッシュ・データとしてデータ・ストア３１３内におけるリソース・キャッシュ内に記憶する。いくつかの例において、シリアライジング・モジュール３０９によって実行可能コードをシリアライズすることの前に、フロント・エンド３０３はシリアライズ実行に関するプロセッサ２０１の利用可能性を判定し得る。そのような例において、プロセッサが利用可能でない場合、フロント・エンド３０３は、プロセッサが利用可能になるまでシリアライズすることを遅延させ得る。

【００８０】

ソース・コードは、複数の二次ソース・コード（たとえば、機能）への参照を含み得る。そのような場合、実行可能コードは各実行可能二次ソース・コードがソース・コード内で参照される二次ソース・コードに対応するように、複数の二次ソース・コードを含み得る。フロント・エンド３０３は、シリアライズ用の実行可能二次ソース・コードを選択することができる。さらにフロント・エンド３０３は、シリアライズ・コード内でシリアライズされないままである実行可能二次ソース・コードを判定し得る。フロント・エンド３０３は、実行可能コードに関連付けられている参照、リロケーション・データ、二次ソース・コードのタイプ、二次ソース・コードのセキュリティ識別子などに基づいて、二次ソース・コードのデシリアライズを判定し得る。ソース・コードは、アプリケーション２０７のトップ・レベル・スクリプトであり得る。フロント・エンド３０３は、たとえば、所定の条件に基づいて、アプリケーション２０７のトップ・レベル・スクリプトをシリアライズ用にソース・コードとして選択し得る。

【００８１】

いくつかの例において、フロント・エンド３０３は、実行を待機する一次ソース・コー

ドの表示（たとえば、リンク、アドレス、ＵＲＬなど）を受信し得る。たとえば、一次ソース・コードは、ウェブ・ページ内のスクリプトであり得る。表示を受信することに応じて、図４Ａのブロック４０３に関して説明されたように、確認モジュール３０５は、図４Ａのブロック４０９において選択された一次ソース・コードおよび二次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認する。データ・ストア３１３内におけるリソース・キャッシュ内のキャッシュ・ミスに際し、フロント・エンド３０３は、データ・ストア３１３内におけるリソース・キャッシュからシリアライズ・コードを含むキャッシュ・データを取り出すことが可能である。

【００８２】

デシリアライジング・モジュール３１９は、取り出されたキャッシュされたシリアライズ・コードを実行可能コードにデシリアライズすることができる。デシリアライジング・モジュール３１９によってデシリアライズすることは、取り出されたキャッシュされたシリアライズ・コードを用いて、実行環境３３３に基づいて実行可能コード内の１組の参照を再記憶することを含み得る。フロント・エンド３０３は、実行環境３３３内における実行用の実行可能コードおよび参照を提供することができる。

【００８３】

なお、いくつかの例において、デシリアライジング・モジュール３１９によってデシリアライズすることは、たとえばデータ破損によって失敗する場合もある。そのような例において、フロント・エンド３０３は、キャッシュされたシリアライズ・コードをデータ・ストア３１３から削除することができる。したがって、実行を待機するソース・コードの表示が次に受信される際、確認モジュール３０５による確認は、図４Ａのブロック４０３に示されるキャッシュ・ミスに戻る可能性があり、図４に従ったシリアライズ・プロセスが行われる。

【００８４】

いくつかの例において、ブロック４０９の二次ソース・コードの選択は、ブロック４０７において取得された実行可能コードの実行から取得された実行結果に基づき得る。たとえば、ブロック４０７において取得された第１実行可能コードは、１回または複数回、二次コードの選択の前に、実行され得る。そのような例において、コード・キャッシング・プラットフォーム２０９は、第１実行コンテキスト内の第１実行可能コードの実行から実行結果を取得してもよく、実行結果に基づいて二次ソース・コードを選択してもよい。第１実行可能コードの実行の際、コード・キャッシング・プラットフォーム２０９は、参照および実行される二次ソース・コード（たとえば、機能）を決定するために、実行に関連するデータを収集し得る。

【００８５】

コード・キャッシング・プラットフォーム２０９は、ソース・コード全体をキャッシュする代わりに、キャッシング用の実行された二次ソース・コードを選択し得る。なぜならば、一次ソース・コードにおける様々な条件、クライアント・デバイス２００やユーザ入力等上の実行環境に関連する条件によると、実行され得る二次ソース・コード（たとえば、機能）もあれば、実行され得ないものもあるためである。コード・キャッシング・プラットフォーム２０９は、クライアント・デバイス２００上の一次ソース・コードの複数の実行に関連するデータを収集することによって、実行される二次ソース・コードを判定することができ、他の二次ソース・コードよりも頻繁に実行される二次ソース・コードを判定することができる。コード・キャッシング・プラットフォーム２０９は、判定に応じて、キャッシング用の二次ソース・コードを選択することができる。さらにコード・キャッシング・プラットフォーム２０９は、各機能呼出が機能の実行を含む一次ソース・コード内の機能呼出を判定し得る。コード・キャッシング・プラットフォーム２０９は、機能呼出の実行に関する情報を収集してもよいし、キャッシング用の実行時において実行されている機能を選択してもよい。

【００８６】

コード・キャッシング・プラットフォーム２０９は、様々な他の要因（たとえば、二次

ソース・コードのサイズ、二次実行可能コードのサイズ、一次ソース・コード内で二次ソース・コードが参照される回数、二次ソース・コードのコンパイル時間など）に基づいて二次ソース・コードを選択し得る。

【００８７】

図４Ｂは、１組の条件に基づいてコードをキャッシュ・コードにシリアライズするためのプロセスの例を示す。ブロック４２１において、フロント・エンド３０３は、実行を待機する一次ソース・コードの表示（たとえば、リンク、アドレス、ＵＲＬなど）を受信する。表示を受信することに応じて、ブロック４２３において、図４Ａに関して上述されたように、確認モジュール３０５は、一次ソース・コードに対応するキャッシュ・データがあるかデータ・ストア３１３内におけるリソース・キャッシュを確認する。ブロック４２３において確認がヒットを発見しない場合、戻された結果はキャッシュ・ミスであり得る（ブロック４２５に図示）。

【００８８】

ブロック４２７において、データ・ストア３１３内のキャッシュ・ミスの際、フロント・エンド３０３は、一次ソース・コードからコンパイルされた実行可能コードを取得することができる。たとえば、フロント・エンド３０３は、一次ソース・コードをコンパイル用にコンパイラ３２７に送ることができる。

【００８９】

ブロック４２９において、シリアライジング・モジュール３０９は、一次ソース・コードのサイズ、所定期間内に一次ソース・コードが実行される回数または頻度、一次ソース・コードのコンパイル時間、もしくは、これらの組み合わせに基づいて、実行可能コードをシリアライズ・コードにシリアライズする。たとえば、一次ソース・コードが週に一度、１０回未満実行される場合、シリアライジング・モジュール３０９は一次ソース・コードのシリアライズを行わない可能性がある。しかしながら、それ以上に（たとえば、週に一度、１０回以上）実行される一次ソース・コードはシリアライズされ得る。ブロック４３１において、フロント・エンド３０３は、シリアライズ・コードを、データ・ストア３１３内におけるリソース・キャッシュ内に、実行環境３３１または３３３内の実行用に提供されるキャッシュ・データとして記憶する。

【００９０】

上述の通り、主題の技術の態様は、異なる実行環境（たとえば、データが１つのウェブ・ブラウザ・セッションにおいてシリアライズされ、別のウェブ・ブラウザ・セッションにおいてデシリアライズされる、ウェブ・ブラウザ）内での実行用の実行可能コードをキャッシュすることの観点で説明されている。しかしながら、本願の技術は、ウェブ・ブラウザ環境に限定されない。主題の技術は、任意の実行環境内で実行可能コードをキャッシュすることに適用されてもよい。さらに、主題の技術は、図１の第１クライアント・デバイス１０１ａ～１０１ｎ（または図１のサーバ１０９ａ～１０９ｍ）中のデータをシリアライズおよびキャッシュすることと、図１の第２クライアント・デバイス１０１ａ～１０１ｎ（または図１のサーバ１０９ａ～１０９ｍ）中のキャッシュ・データを、第２クライアント・デバイス１０１ａ～１０１ｎ（またはサーバ１０９ａ～１０９ｍ）中においてキャッシュ・データがデシリアライズおよび実行されるように、転送することとに適用されてもよい。

【００９１】

上述のフィーチャおよびアプリケーションは、コンピュータ可読記憶媒体（コンピュータ可読媒体とも称される）上に記録される一組の命令として記述されるソフトウェアプロセスとして実装される。これらの命令が１以上の処理ユニット（たとえば、１以上のプロセッサ、プロセッサ・コア、他の処理ユニットを含み得る）によって実行されるとき、これらの命令によって、処理ユニットは当該命令に示されるアクションを行う。コンピュータ可読媒体の例として、ＣＤ－ＲＯＭ、フラッシュ・ドライブ、ＲＡＭチップ、ハード・ドライブ、ＥＰＲＯＭが含まれるが、コンピュータ可読媒体の例はこれらに限定されない。コンピュータ可読媒体は、無線または有線接続で送出される搬送波および電子信号を含

まない。

【００９２】

本明細書においては、用語「ソフトウェア」は、読出専用メモリ内のファームウェア、または、プロセッサによって処理を行うためのメモリに読み込まれる磁気ストレージを含むことが意図されている。また、いくつかの実施形態では、本願の技術の複数のソフトウェア技術は、区別がなされる本願の技術のソフトウェア態様でありつつ、より大きなプログラムの下位区分として実装される。いくつかの実施形態では、複数のソフトウェア技術は別個のプログラムとして実装される。そして、本明細書に記載されるソフトウェア態様を同時に実装する別個のプログラムは、いずれも本願の技術の範囲内にある。いくつかの実施形態では、ソフトウェア・プログラムは、インストールされて電子システムを操作する際、該ソフトウェア・プログラムの操作を実行する特定の機械の実装を規定する。

【００９３】

コンピュータ・プログラム（プログラム、ソフトウェア、ソフトウェア・アプリケーション、スクリプト、またはコードとも称される）は、任意の形式のプログラミング言語（たとえば、コンパイル済みまたはインタプリット済み言語、宣言型言語または手続型言語）で書かれてもよく、どのような形式で配置されてもよい（たとえば、スタンドアロン・プログラム、モジュール、コンポーネント、サブルーチン、オブジェクト、コンピューティング環境においての使用が相応しい他のユニットを含む）。コンピュータ・プログラムは、ファイルシステム内のファイルに対応してもよいが、これに限定されない。プログラムは、他のプログラムやデータを保持するファイルの一部分（マークアップ言語ドキュメントに記憶されるスクリプト）、対象プログラム専用の１つのファイル、または複数の調整ファイル（たとえば、モジュール、サブプログラム、コードを部分的に記憶するファイル）に記憶される。コンピュータ・プログラムは、１つのサイトに位置するか、もしくは複数のサイトに広がり、通信ネットワークによって相互接続される１つのコンピュータまたは複数のコンピュータ上で実行されるように配置される。

【００９４】

図５は、本願の技術のいくつかの実施形態が実装される一例の電子システム５００を概念的に示す。電子システム５００は、コンピュータ（たとえば、電話機、ＰＤＡ）であってもよいし、他の任意の電子デバイスであってもよい。このような電子システムは、様々な種類のコンピュータ可読媒体と、他の様々な種類のコンピュータ可読媒体に関するインターフェースとを含む。電子システム５００は、バス５０８、１または複数の処理ユニット５１２、システム・メモリ５０４、読出専用メモリ（ＲＯＭ）５１０、パーマネント・ストレージ・デバイス５０２、入力デバイス・インターフェース５１４、出力デバイス・インターフェース５０６、およびネットワーク・インターフェース５１６を含む。

【００９５】

バス５０８は、電子システム５００の多数の内部デバイスを通信可能に接続する全てのシステムと、周辺機器と、チップセットバスとを集合的に表す。たとえば、バス５００は、１または複数の処理ユニット５１２を、ＲＯＭ５１０と、システム・メモリ５０４と、パーマネント・ストレージ・デバイス５０２とに通信可能に接続する。

【００９６】

１または複数の処理ユニット５１２は、本開示のプロセスを実行するために、これらの様々なメモリユニットから実行対象の命令と処理対象のデータとを取り出す。１または複数の処理ユニットは、異なる実施形態において、単一のプロセッサであってもよいし、マルチコアプロセッサであってもよい。

【００９７】

ＲＯＭ５１０は、１または複数の処理ユニット５１２と電子システムの他のモジュールとによって必要とされる静的データと命令とを記憶する。一方、パーマネント・ストレージ・デバイス５０２は、読出／書込メモリデバイスである。このデバイスは、電子システム５００がオフであるときでも命令とデータとを記憶する不揮発性メモリユニットである。本開示のいくつかの実施形態では、パーマネント・ストレージ・デバイス５０２として

マスストレージデバイス（たとえば、磁気または光ディスク、および、対応するディスクドライブ）が用いられる。

【００９８】

他の実施形態では、パーマネント・ストレージ・デバイス５０２としてリムーバブルストレージデバイス（たとえば、フロッピー（登録商標）・ディスク、フラッシュ・ドライブ、対応するディスクドライブ）が用いられる。システム・メモリ５０４は、パーマネント・ストレージ・デバイス５０２と同様に、読出／書込メモリデバイスである。しかしながら、システム・メモリ５０４は、ストレージ・デバイス５０２とは異なり、揮発性読出／書込メモリ（たとえば、ランダムアクセスメモリ）である。システム・メモリ５０４は、ランタイムにおいてプロセッサが必要とする命令とデータとのいくつかを記憶する。いくつかの実施形態においては、本開示のプロセスは、システム・メモリ５０４、パーマネント・ストレージ・デバイス５０２、またはＲＯＭ５１０に記憶される。たとえば、様々なメモリユニットは、いくつかの実施形態に従ってウェブエレメントを提示するための命令を含む。１または複数の処理ユニット５１２は、いくつかの実施形態のプロセスを実行するために、これらの様々なメモリユニットから実行予定の命令と処理予定のデータとを取り出す。

【００９９】

さらにバス５０８は、入力デバイス・インターフェース５１４と出力デバイス・インターフェース５０６とに接続する。入力デバイス・インターフェース５１４によって、ユーザが情報を通信し、電子システムに対するコマンドを選択することが可能になる。入力デバイス・インターフェース５１４とともに用いられる入力デバイスは、たとえば、英数字キーボード、ポインティングデバイス（「カーソルコントロールデバイス」とも称される）を含む。出力デバイス・インターフェース５０６によって、たとえば、電子システム５００によって生成された画像の表示が可能になる。出力デバイス・インターフェース５０６とともに用いられる出力デバイスは、たとえば、プリンタ、ディスプレイデバイス（ブラウン管（ＣＲＴ）または液晶ディスプレイ（ＬＣＤ）など）を含む。いくつかの実施形態は、入力および出力デバイスの両方として機能するタッチスクリーンなどのデバイスを含む。

【０１００】

最後に、図５に示すように、バス５０８はさらに電子システム５００を、ネットワーク・インターフェース５１６を介して図示しないネットワークに接続する。このように、コンピュータは、コンピュータのネットワーク（たとえば、ローカル・エリア・ネットワーク（「ＬＡＮ」）、ワイド・エリア・ネットワーク（「ＷＡＮ」）、イントラネット）の一部、または、複数のネットワークのうちの１つのネットワーク（たとえば、インターネット）であってもよい。任意の、または、すべての電子システム５００の要素が、本開示とともに用いられる。

【０１０１】

これら上述の機能は、デジタル電子回路、コンピュータソフトウェア、ファームウェア、またはハードウェアにより実装されることができる。上述の技術は、コンピュータ・プログラム製品を用いて実装される。プログラマブル・プロセッサおよびコンピュータは、モバイルデバイスに含まれてもよいし、モバイルデバイスとしてパッケージされてもよい。プロセスおよびロジック・フローは、プログラマブル・プロセッサとプログラマブル論理回路とによって行われる。汎用および専用のデバイスと、クライアント・デバイスとは、通信ネットワークを通じて相互接続される。

【０１０２】

いくつかの実施形態は電子部品（たとえば、機械可読またはコンピュータ可読媒体（あるいは、コンピュータ可読記憶媒体、機械可読媒体、機械可読記憶媒体とも称される）におけるコンピュータ・プログラムの命令を記憶するマイクロプロセッサ、ストレージ、メモリ）を含む。このようなコンピュータ可読媒体のいくつかの例には、ＲＡＭ、ＲＯＭ、読出専用コンパクトディスク（ＣＤ－ＲＯＭ）、記録可能コンパクトディスク（ＣＤ－Ｒ

）、書換可能コンパクトディスク（ＣＤ－ＲＷ）、読出専用デジタル多用途ディスク（たとえば、ＤＶＤ－ＲＯＭ、２層ＤＶＤ－ＲＯＭ）、種々の記録可能／書換可能ＤＶＤ（たとえば、ＤＶＤ－ＲＡＭ、ＤＶＤ－ＲＷ、ＤＶＤ＋ＲＷ）、フラッシュメモリ（たとえば、ＳＤカード、ミニＳＤカード、マイクロＳＤカード）、磁気またはソリッド・ステート・ハード・ドライブ、読出専用および記録可能Ｂｌｕ－ｒａｙ（登録商標）ディスク、超高密度光ディスク、他の任意の光または磁気メディア、フロッピー・ディスクが含まれる。コンピュータ可読媒体は、少なくとも１つの処理ユニットにより実行可能であって、様々な操作を行うための複数組の命令を含むコンピュータ・プログラムを記憶できる。コンピュータ・プログラムまたはコンピュータコードの例には、たとえばコンパイラによって生成される機械コードと、インタプリタを用いてコンピュータ、電子部品、またはマイクロプロセッサによって実行される、より高レベルのコードを含むファイルと、が含まれる。

【０１０３】

上記議論はソフトウェアを実行するマイクロプロセッサまたはマルチコアプロセッサを指しているが、いくつかの実施形態は集積回路（たとえば、特定用途向け集積回路（ＡＳＩＣ）またはフィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ））によって実行される。いくつかの実施形態では、このような集積回路は回路上自体に記憶される命令を実行する。

【０１０４】

本明細書および本願の任意の請求項で用いられる通り、用語「コンピュータ」、「サーバ」、「プロセッサ」および「メモリ」はすべて、電子または他の技術分野のデバイスのことを指す。これらの用語から、人または人の集団は除外される。本明細書上では、用語「表示」または「表示する」は、電子デバイス上での表示を意味する。本明細書および本願の任意の請求項で用いられる通り、用語「コンピュータ可読媒体」「複数のコンピュータ可読媒体」は全体として、コンピュータによって読み込まれる形式の情報を記憶する有形の物理的な物体に限定される。これらの用語から、無線信号、有線ダウンロード信号、および他の任意の一時的な信号は除外される。

【０１０５】

ユーザとの対話を提供するため、本明細書に記載される主題の実施形態は、ユーザに対して情報を表示するためのディスプレイデバイス（たとえば、ＣＲＴ（ブラウン管）またはＬＣＤ（液晶ディスプレイ）モニタ）と、キーボードと、ユーザがコンピュータに対して入力を提供できるポインティングデバイス（たとえば、マウスまたはトラックボール）と、を有するコンピュータ上で実施可能である。他の種類のデバイスも同様に、ユーザとの対話を提供するために用いられる。たとえば、ユーザに対して提供されるフィードバックは、任意の形式の感覚フィードバック（たとえば、視覚フィードバック、聴覚フィードバック、触覚フィードバックなど）であってよいし、ユーザからの入力は任意の形式（音響、音声または触覚入力を含む）で受信されてよい。さらに、ユーザによって用いられるデバイスに対してドキュメントを送信し、該デバイスからドキュメントを受信することによって（たとえば、ユーザのクライアント・デバイス上でウェブ・ブラウザに対して、該ウェブ・ブラウザから受信したリクエストに応じてウェブ・ページを送信することによって）、コンピュータはユーザとの対話が可能になる。

【０１０６】

本明細書に記載される実施形態の主題は、バックエンド・コンポーネント（たとえば、データサーバ）、ミドルウェア・コンポーネント（たとえば、アプリケーション・サーバ）、フロントエンド・コンポーネント（たとえば、ユーザの本明細書に記載される主題の実施形態との対話を可能にするグラフィカル・ユーザ・インターフェースまたはウェブ・ブラウザを有するクライアントコンピュータ）、または、このようなバックエンド、ミドルウェア、およびフロントエンド・コンポーネントのうち任意の組み合わせを含むコンピューティング・システムにおいて実施可能である。該システムのコンポーネントは、デジタルデータ通信の任意の形式または媒体（たとえば、通信ネットワーク）によって相互接

続される。通信ネットワークの例には、ローカル・エリア・ネットワーク（「ＬＡＮ」）、ワイド・エリア・ネットワーク（「ＷＡＮ」）、インターネットワーク（たとえば、インターネット）、および、ピアツーピアネットワーク（たとえば、アドホック・ピアツーピア・ネットワーク）が含まれる。

【０１０７】

コンピューティング・システムには、クライアントおよびサーバが含まれる。クライアントおよびサーバは一般的には互いにリモートであり、通信ネットワークを介して対話し得る。クライアントおよびサーバの関係が生じるのは、各コンピュータ上で起動しており、互いにクライアント／サーバ関係を有するコンピュータ・プログラムを介してである。いくつかの実施形態においては、サーバがデータ（たとえば、ＨＴＭＬページ）をクライアント・デバイスに対して（たとえば、該クライアント・デバイスとの対話を行うユーザに対してデータを表示する目的、および、該ユーザからユーザ入力を受信する目的で）送信する。クライアント・デバイスにおいて生成されたデータ（たとえば、ユーザ対話の結果）は、サーバにおいてクライアント・デバイスから受信される。

【０１０８】

開示されるプロセスにおける工程の任意の特定の順番または階層は例示的アプローチの実例である、と理解される。デザインの嗜好に基づいて、プロセスにおける工程の特定の順番または階層は再配置されてもよい、図示される全ての工程が実行されてもよい、と理解される。いくつかの工程は同時に実行されてもよい。たとえば、ある状況においては、マルチタスキングおよびパラレル・プロセッシングは効果的である。さらに、上述の実施形態における様々なシステム・コンポーネントの分離については、全ての実施形態内においてそのような分離が必須であるとして理解されるべきではない。上述のプログラムコンポーネントおよびシステムは一般的に１つのソフトウェア製品に統合される、または、複数のソフトウェア製品にパッケージされる、と理解されるべきである。

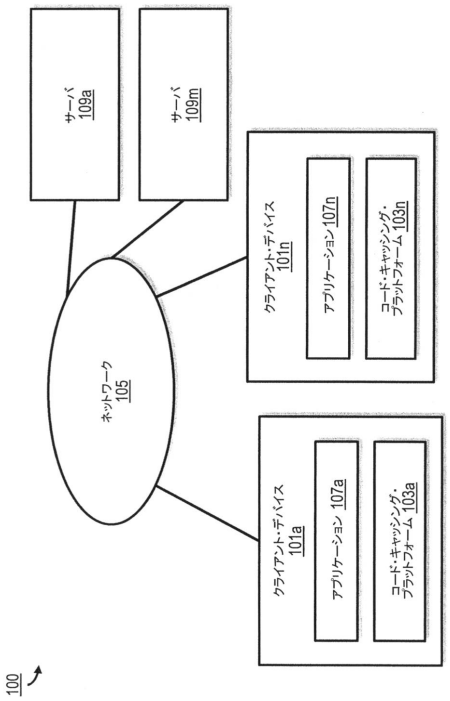
【０１０９】

上記記載は、どの当業者でも本明細書に記載されている様々な態様を実施できるようにするために提供される。これらの態様に対する様々な修正形態は当業者によって容易に識別され、本明細書に規定される包括的な原則は他の態様にも適用される。従って、本請求項は、本明細書に示される態様に限定される意図はなく、請求項の文言と一致する全ての範囲が付与され、単数である要素への参照は「１および１のみ」であることを（そのように特定的に記述されている場合を除いて）意味するのではなく、「１以上」を意味する。用語「いくつか」は、（特定的に記述する場合を除いて）１以上を指す。男性の指示代名詞（たとえば、「彼の」）は、女性および中性（たとえば、「彼女の」「その」）を含み、その逆も同様である。見出しおよび小見出しは、それらがある場合にしても、便宜上用いられているだけであって、本開示を限定するものではない。

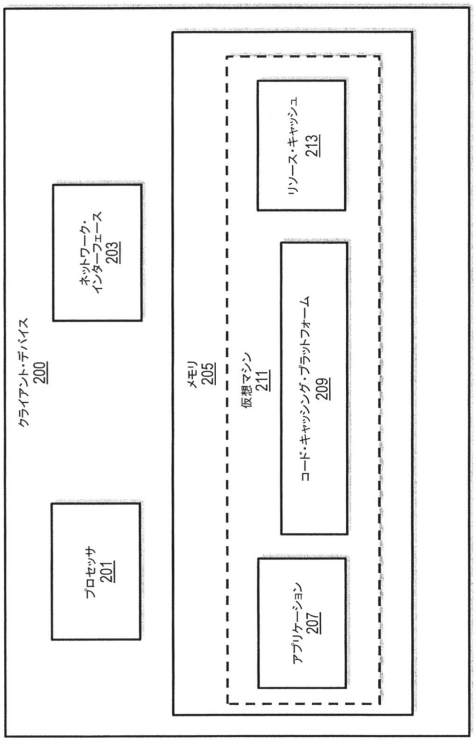
【０１１０】

たとえば「態様」という語句は、その態様が本願の技術に必要不可欠であること、その態様が本願の技術の全ての構成に適用されること、を暗示しない。態様に関する開示は、全ての構成または１以上の構成に適用される。たとえば態様という語句は、１以上の態様を指してもよいし、その逆も同様である。たとえば「構成」という語句は、そのような構成が本願の技術に必要不可欠であること、その態様が本願の技術の全ての構成に適用されること、を暗示しない。態様に関する開示は、全ての構成または１以上の構成に適用される。たとえば構成という語句は、１以上の構成を指してもよいし、その逆も同様である。

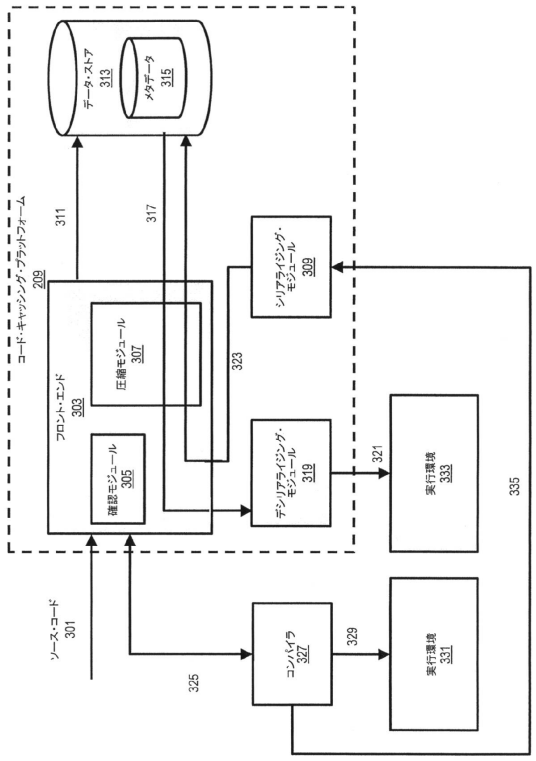
【図１】


サーバ
109a
サーバ
109m
ネットワーク
105
クライアント・デバイス
101n
アプリケーション 107n
コード・キャッシング・
プラットフォーム 103n
クライアント・デバイス
101a
アプリケーション 107a
コード・キャッシング・
プラットフォーム 103a
100

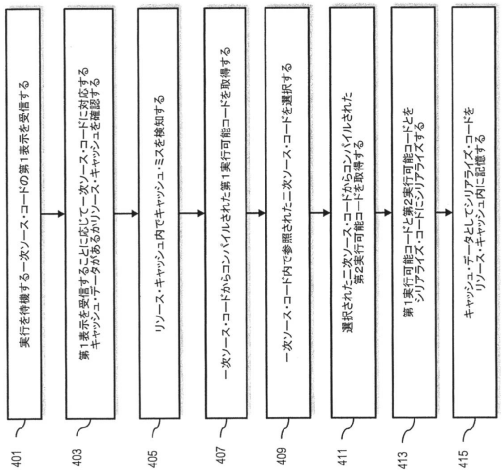
【図２】


クライアント・デバイス
200
プロセッサ
201
ネットワーク・
インターフェース
203
メモリ
205
仮想マシン
211
アプリケーション
207
コード・キャッシング・プラットフォーム
209
リソース・キャッシュ
213

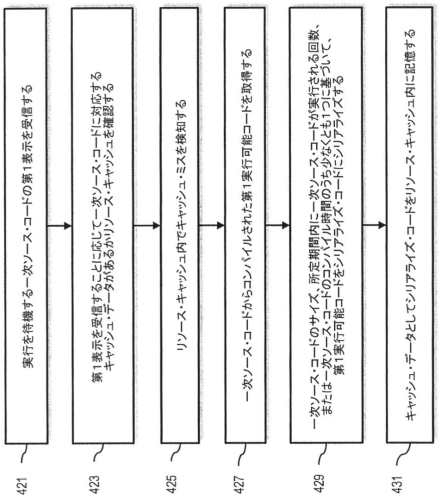
【図３】


ソース・コード
301
コード・キャッシング・プラットフォーム
209
フロント・エンド
303
護認モジュール
305
圧縮モジュール
307
デシリアライジング・
モジュール
319
シリアライジング・
モジュール
309
データ・ストア
313
メタデータ
315
コンパイラ
327
実行環境
331
実行環境
333
311
317
321
323
325
329
335

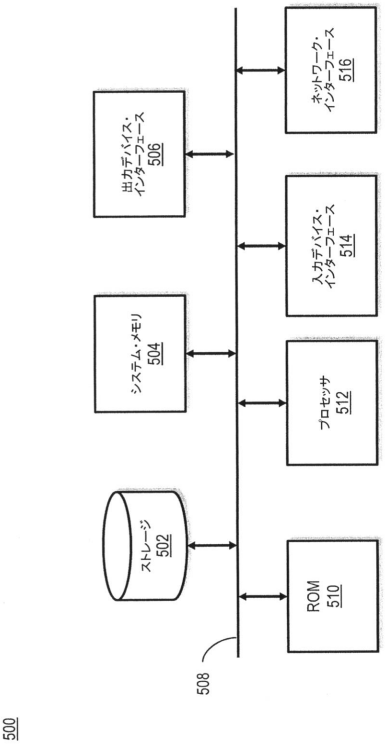
【図４Ａ】


実行を待機する一次ソース・コードの第1表示を受信する
第1表示を受信することに応じて一次ソース・コードに対応する
キャッシュ・データがあるかリソース・キャッシュを確認する
リソース・キャッシュ内でキャッシュ・ミスを検知する
一次ソース・コードからコンパイルされた第1実行可能コードを取得する
一次ソース・コード内で参照された二次ソース・コードを選択する
選択された二次ソース・コードからコンパイルされた
第2実行可能コードを取得する
第1実行可能コードと第2実行可能コードと
シリアライズ・コードにシリアライズする
キャッシュ・データとしてシリアライズ・コードを
リソース・キャッシュ内に記憶する
401
403
405
407
409
411
413
415

【図４Ｂ】


実行を待機する一次ソース・コードの第1表示を受信する
第1表示を受信することに応じて一次ソース・コードに対応するキャッシュ・データがあるかリソース・キャッシュを確認する
リソース・キャッシュ内でキャッシュ・ミスを検知する
一次ソース・コードからコンパイルされた第1実行可能コードを取得する
一次ソース・コードのサイズ、所定期間内に一次ソース・コードが実行される回数、または一次ソース・コードのコンパイル時間のうち少なくとも1つに基づいて、第1実行可能コードをシリアライズ・コードにシリアライズする
キャッシュ・データとしてシリアライズ・コードをリソース・キャッシュ内に記憶する
421
423
425
427
429
431

【図５】


ストレージ 502
システム・メモリ 504
出力デバイス・インターフェース 506
508
ROM 510
プロセッサ 512
入力デバイス・インターフェース 514
ネットワーク・インターフェース 516
500

フロントページの続き

(72)発明者　フォーゲルハイム、ダニエル
アメリカ合衆国　９４０４３　カリフォルニア州　マウンテン　ビュー　アンフィシアター　パークウェイ　１６００　グーグル　インコーポレイテッド内
(72)発明者　アイジンガー、ヨッヘン　マティアス
アメリカ合衆国　９４０４３　カリフォルニア州　マウンテン　ビュー　アンフィシアター　パークウェイ　１６００　グーグル　インコーポレイテッド内

審査官　杉浦　孝光

(56)参考文献　特開２００７－２３３４７２（ＪＰ，Ａ）
米国特許出願公開第２０１３／０２１２５６７（ＵＳ，Ａ１）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　　８／４１
Ｇ０６Ｆ　　９／４４５