(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2017-194988
(P2017-194988A)
(43)公開日　平成29年10月26日(2017.10.26)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 12/00	(2006.01)	G06F	12/00	546K	5B084
G06F 13/00	(2006.01)	G06F	13/00	540B	

審査請求 有　請求項の数 20　ＯＬ　（全 27 頁）

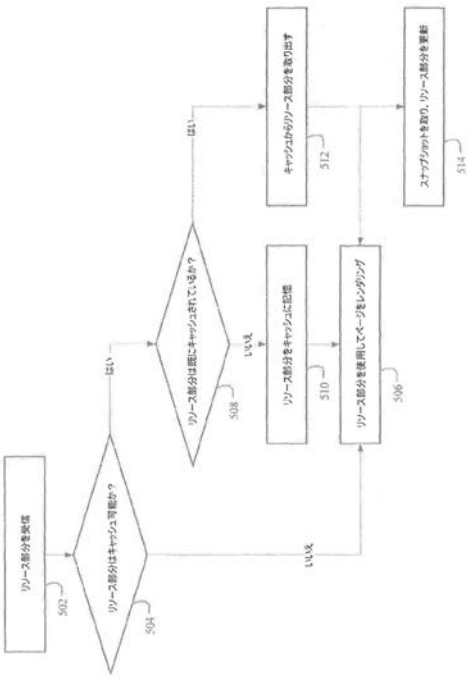
(21)出願番号　特願2017-114072 (P2017-114072)
(22)出願日　平成29年6月9日 (2017.6.9)
(62)分割の表示　特願2015-560286 (P2015-560286)の分割
原出願日　平成26年2月27日 (2014.2.27)
(31)優先権主張番号　13/782,937
(32)優先日　平成25年3月1日 (2013.3.1)
(33)優先権主張国　米国 (US)

(71)出願人　508178054
フェイスブック，インク．
アメリカ合衆国　カリフォルニア　９４０２５，メンロー　パーク，ウィロー　ロード　１６０１
(74)代理人　100105957
弁理士　恩田　誠
(74)代理人　100068755
弁理士　恩田　博宣
(72)発明者　ジャン、ジシャオ
アメリカ合衆国　９４０２５　カリフォルニア州　メンロー　パーク　ウィロー　ロード　１６０１

最終頁に続く

(54)【発明の名称】構造化文書のページレットのキャッシング

(57)【要約】　（修正有）
【課題】構造化文書のページレットをキャッシュし、キャッシュされたリソースを使用する方法を提供する。
【解決手段】ウェブ・ページを求める要求を受信する工程と、キャッシュに記憶されているリソース部分を識別する工程と、キャッシュに記憶されているリソース部分を使用して、要求されたウェブ・ページをレンダリングする工程を行うように命令する工程と、リソース部分をリフレッシュする工程と、差分データの生成と並行して、リソース部分をレンダリングするように命令する工程と、差分データを送信する工程と、差分データに基づいて、キャッシュに記憶されているリソース部分を更新するように、および、更新されたリソース部分をレンダリングするように、命令する工程と、を含む、リフレッシュ工程と、を備える。
【選択図】図５



【特許請求の範囲】
【請求項１】
サーバ・コンピューティング・システムが、ウェブ・ページを求める要求をクライアント・コンピューティング・デバイスから受信する工程と、
前記サーバ・コンピューティング・システムが、キャッシュに記憶されている１以上のリソース部分を識別する工程であって、各リソース部分は、要求された前記ウェブ・ページをレンダリングするために前記クライアント・コンピューティング・デバイスによる使用のための構造化文書の部分に対応している、工程と、
前記サーバ・コンピューティング・システムが、前記クライアント・コンピューティング・デバイスに、前記キャッシュに記憶されている前記１以上のリソース部分を使用して、要求された前記ウェブ・ページをレンダリングする、ページレンダリング工程を行うように命令する工程と、
前記サーバ・コンピューティング・システムが、前記１以上のリソース部分をリフレッシュする工程であって、
存在する場合、前記１以上のリソース部分について差分データを生成する工程であって、前記１以上のリソース部分についての差分データは、前記１以上のリソース部分に対して変更が行われたことを示す、工程と、
前記差分データの生成と並行して、前記１以上のリソース部分をレンダリングするように前記クライアント・コンピューティング・デバイスに命令する工程と、
前記差分データを前記クライアント・コンピューティング・デバイスに送信する工程と、
前記差分データに基づいて、前記キャッシュに記憶されている前記リソース部分を更新するように、および、更新された前記リソース部分をレンダリングするように、前記クライアント・コンピューティング・デバイスに命令する工程と、を含む、リフレッシュ工程と、を備える方法。
【請求項２】
前記リフレッシュ工程は、
存在する場合、前記１以上のリソース部分について更新されたデータを含む１以上のリソース更新部分を生成する工程と、
前記１以上のリソース更新部分を前記クライアント・コンピューティング・デバイスに送信する工程と、をさらに含む、請求項１に記載の方法。
【請求項３】
前記リフレッシュ工程は、
１以上の置換リソース部分を生成する工程であって、前記１以上の置換リソース部分は、存在する場合、前記１以上のリソース部分について前記１以上のリソース部分と更新されたデータとを含む、工程と、
前記１以上の置換リソース部分を前記クライアント・コンピューティング・デバイスに送信する工程と、をさらに含む、請求項１に記載の方法。
【請求項４】
前記１以上のリソース部分が前記キャッシュに最後に記憶された後、前記１以上のリソース部分の状態に影響するユーザ・イベントが発生したか否かを決定する工程と、
前記ユーザ・イベントが発生した場合、前記１以上のリソース部分の前記状態に影響する前記ユーザ・イベントに基づいて、前記キャッシュに記憶される前記１以上のリソース部分を再生成する工程と、をさらに備える、請求項１に記載の方法。
【請求項５】
前記キャッシュは、前記クライアント・コンピューティング・デバイスの永続的なストレージに記憶される、請求項１に記載の方法。
【請求項６】
前記１以上のリソース部分が暗号化フォーマットで前記キャッシュに記憶され、前記ページレンダリング工程は、前記１以上のリソース部分を解読する工程を備える、請求項５

に記載の方法。
【請求項７】
ユーザ・イベントに応答して前記キャッシュを消去する工程をさらに備える、請求項５に記載の方法。
【請求項８】
前記ユーザ・イベントは、前記ウェブ・ページが要求されたウェブ・サイトからログ・オフすることを含む、請求項７に記載の方法。
【請求項９】
１以上のプロセッサと、
前記プロセッサの１以上に結合され、命令を備える１以上のコンピュータ読み取り可能な非一時的記憶媒体と、を備えるシステムであって、前記プロセッサの１以上によって実行されると、前記命令は、前記システムに、
ウェブ・ページを求める要求をクライアント・コンピューティング・デバイスから受信する工程と、
キャッシュに記憶されている１以上のリソース部分を識別する工程であって、各リソース部分は、要求された前記ウェブ・ページをレンダリングするために前記クライアント・コンピューティング・デバイスによる使用のための構造化文書の部分に対応している、工程と、
前記クライアント・コンピューティング・デバイスに、前記キャッシュに記憶されている前記１以上のリソース部分を使用して、要求された前記ウェブ・ページをレンダリングする、ページレンダリング工程を行うように命令する工程と、
前記１以上のリソース部分をリフレッシュする工程であって、
存在する場合、前記１以上のリソース部分について差分データを生成する工程であって、前記１以上のリソース部分についての差分データは、前記１以上のリソース部分に対して変更が行われたことを示す、工程と、
前記差分データの生成と並行して、前記１以上のリソース部分をレンダリングするように前記クライアント・コンピューティング・デバイスに命令する工程と、
前記差分データを前記クライアント・コンピューティング・デバイスに送信する工程と、
前記差分データに基づいて、前記キャッシュに記憶されている前記リソース部分を更新するように、および、更新された前記リソース部分をレンダリングするように、前記クライアント・コンピューティング・デバイスに命令する工程と、を含む、リフレッシュ工程と、を行わせるように構成される、システム。
【請求項１０】
実行されると、前記命令は、
存在する場合、前記１以上のリソース部分について更新されたデータを含む１以上のリソース更新部分を生成する工程と、
前記１以上のリソース更新部分を前記クライアント・コンピューティング・デバイスに送信する工程と、
前記１以上のリソース部分と前記１以上のリソース更新部分とを使用して、前記キャッシュに前記リソース更新部分を記憶し、要求された前記ウェブ・ページをレンダリングするよう前記クライアント・コンピューティング・デバイスに命令する工程と、によって、前記リフレッシュ工程が行われるようにさらに構成される、請求項９に記載のシステム。
【請求項１１】
実行されると、前記命令は、
１以上の置換リソース部分を生成する工程であって、前記１以上の置換リソース部分は、存在する場合、前記１以上のリソース部分について前記１以上のリソース部分と更新されたデータとを含む、工程と、
前記１以上の置換リソース部分を前記クライアント・コンピューティング・デバイス

に送信する工程と、
前記１以上の置換リソース部分を使用して、前記キャッシュ内の前記１以上のリソース部分を前記１以上の置換リソース部分に置換し、要求された前記ウェブ・ページをレンダリングするよう前記クライアント・コンピューティング・デバイスに命令する工程と、によって、前記リフレッシュ工程が行われるようにさらに構成される、請求項９に記載のシステム。
【請求項１２】
実行されると、前記命令は、
前記１以上のリソース部分が前記キャッシュに最後に記憶された後、前記１以上のリソース部分の状態に影響するユーザ・イベントが発生したか否かを決定する工程と、
前記ユーザ・イベントが発生した場合、前記１以上のリソース部分の前記状態に影響する前記ユーザ・イベントに基づいて、前記キャッシュに記憶される前記１以上のリソース部分を再生成する工程と、が行われるようにさらに構成される、請求項９に記載のシステム。
【請求項１３】
前記キャッシュは、前記クライアント・コンピューティング・デバイスの永続的なストレージに記憶される、請求項９に記載のシステム。
【請求項１４】
前記１以上のリソース部分が暗号化フォーマットで前記キャッシュに記憶され、
実行されると、前記命令は、前記ページレンダリング工程が、前記１以上のリソース部分を解読する工程によって行われるようにさらに構成される、請求項１３に記載のシステム。
【請求項１５】
実行されると、前記命令は、ユーザ・イベントに応答して前記キャッシュを消去する工程が行われるようにさらに構成される、請求項１３に記載のシステム。
【請求項１６】
前記ユーザ・イベントは、前記ウェブ・ページが要求されたウェブ・サイトからログ・オフすることを含む、請求項１５に記載のシステム。
【請求項１７】
ソフトウェアを具体化する１以上のコンピュータ読み取り可能な非一時的記憶媒体であって、実行されると、前記ソフトウェアは、
サーバ・コンピューティング・システムが、ウェブ・ページを求める要求をクライアント・コンピューティング・デバイスから受信する工程と、
前記サーバ・コンピューティング・システムが、キャッシュに記憶されている１以上のリソース部分を識別する工程であって、各リソース部分は、要求された前記ウェブ・ページをレンダリングするために前記クライアント・コンピューティング・デバイスによる使用のための構造化文書の部分に対応している、工程と、
前記サーバ・コンピューティング・システムが、前記クライアント・コンピューティング・デバイスに、前記キャッシュに記憶されている前記１以上のリソース部分を使用して、要求された前記ウェブ・ページをレンダリングする、ページレンダリング工程を行うように命令する工程と、
前記サーバ・コンピューティング・システムが、前記１以上のリソース部分をリフレッシュする工程であって、
存在する場合、前記１以上のリソース部分について差分データを生成する工程であって、前記１以上のリソース部分についての差分データは、前記１以上のリソース部分に対して変更が行われたことを示す、工程と、
前記差分データの生成と並行して、前記１以上のリソース部分をレンダリングするように前記クライアント・コンピューティング・デバイスに命令する工程と、
前記差分データを前記クライアント・コンピューティング・デバイスに送信する工程と、

前記差分データに基づいて、前記キャッシュに記憶されている前記リソース部分を更新するように、および、更新された前記リソース部分をレンダリングするように、前記クライアント・コンピューティング・デバイスに命令する工程と、を含む、リフレッシュ工程と、が行われるように構成される、媒体。

【請求項１８】

前記ソフトウェアは、

前記１以上のリソース部分が前記キャッシュに最後に記憶された後、前記１以上のリソース部分の状態に影響するユーザ・イベントが発生したか否かを決定する工程と、

前記ユーザ・イベントが発生した場合、前記１以上のリソース部分の前記状態に影響する前記ユーザ・イベントに基づいて、前記キャッシュに記憶される前記１以上のリソース部分を再生成する工程と、が行われるようにさらに構成される、請求項１７に記載の媒体。

【請求項１９】

前記キャッシュは、前記クライアント・コンピューティング・デバイスの永続的なストレージに記憶される、請求項１７に記載の媒体。

【請求項２０】

前記１以上のリソース部分が暗号化フォーマットで前記キャッシュに記憶され、

前記ソフトウェアは、前記ページレンダリング工程が、前記１以上のリソース部分を解読する工程によって行われるようにさらに構成される、請求項１９に記載の媒体。

【発明の詳細な説明】

【技術分野】

【０００１】

本開示は、一般に、構造化文書（ウェブ・ページなど）をレンダリングすることに関し、より詳細には、構造化文書を効率的にレンダリングし、体感レンダリング時間を減少させるために、構造化文書のページレットをキャッシュすること、およびキャッシュされたリソースを使用することに関する。

【背景技術】

【０００２】

従来、クライアント・デバイスによって送信されたウェブ・ページまたは他の構造化文書を求める要求が、ウェブ・ページをホストするサーバまたはコンピューティング・システムによって受信されるとき、ホストするシステムは、通常、ハイパー・テキスト・マークアップ言語（ＨＴＭＬ）、拡張可能なマークアップ言語（ＸＭＬ）、または他のウェブ・ブラウザにサポートされる構造化文書の形式で、ベース・ウェブ・ページを生成する。生成された構造化文書は次いで、クライアント・デバイスでのレンダリングのために、ハイパーテキスト転送プロトコル（ＨＴＴＰ）または他の好適な接続を通じて、要求するクライアントに応答して送信される。構造化文書は、送信された文書内に埋め込まれる、１つもしくは複数のリソース（たとえば、ＪａｖａＳｃｒｉｐｔ（登録商標）スクリプトまたはリソース、カスケーディング・スタイル・シート（ＣＳＳ）リソース、画像、動画、その他）、またはそのようなリソースへの参照を含む。例として、ＨＴＭＬ文書に埋め込まれるリソースは、リソースのタイプに応じて、一般に、とりわけ、スクリプト要素、画像要素、またはオブジェクト要素内に含まれてよい、またはそれらの要素内で指定されてよい。リソースを参照する、または指定する要素は、ウェブ・ページを要求するクライアントに対してリソースのロケーションを識別するソース属性（たとえば、ｓｒｃ）を含む。通常、リソースを受信すると、クライアント・デバイスで稼働しているウェブ・ブラウザまたは他のクライアント・アプリケーションが、受信された構造化文書の文書オブジェクト・モデル（ＤＯＭ：ｄｏｃｕｍｅｎｔ　ｏｂｊｅｃｔ　ｍｏｄｅｌ）表現を構築し、文書に埋め込まれるリソース（これは、１つまたは複数の他の外部のロケーションにあってもよい）を要求する。

【０００３】

さらに、リモート・クライアントでウェブ・コンテンツを見るユーザが（たとえば、現

在レンダリングされているウェブ・ページ内のリンクをクリックすることによって、ブラウザ・アプリケーションの戻るボタンもしくは進むボタンをクリックすることによって、またはターゲット・ウェブ・ページのＵＲＬを入力することによって）、現在レンダリングされているウェブ・ページから、新しい（または「ターゲットの」）ウェブ・ページにナビゲートすることを所望するとき、ウェブ・コンテンツのレンダリングを担うブラウザは、新しいウェブ・ページを求める要求を作って、その要求を新しいウェブ・ページをホストするサーバに送信する。そのようにして、従来、ユーザが新しいウェブ・ページにナビゲートすることを要求する度に、ブラウザは、新しい全ウェブ・ページを求める要求をサーバに送信し、現在レンダリングされているページをアンロードし、サーバから受信された新しいウェブ・ページをその全体においてレンダリングする。

【発明の概要】

【発明が解決しようとする課題】

【０００４】

従来、この全ページをロードし、アンロードする方式は、ユーザが要求する各後続のページについて当てはまることになる。ウェブ・ページおよび基礎となるウェブ・ページに埋め込まれた一定のリソースは、ブラウザ・キャッシュに配置されてよく、ローカルに取り出されてよい。しかしながら、多くの動的な、または対話型のウェブ・ページは、それらが最後にレンダリングされた後、頻繁に変更されることがある、または更新されることがある、コンテンツおよび他のリソースを含む。従来、キャッシュされたページのいずれかの部分が変更される場合、全体のキャッシュされたページが無効にされ、キャッシュから出される。

【課題を解決するための手段】

【０００５】

開示される主題は、リモートのおよび／またはローカルなデータ・ストアから、キャッシュされたリソースへの更新を取り出すための非同期の技法と連携して、キャッシュされたリソースを使用するウェブ・ページおよび他の構造化文書を効率的にレンダリングすることに関する。特定の実施形態は、コンテンツの再ローディング（サーバからリモートでアクセスされたにせよ、および／または、キャッシュからローカルにアクセスされたにせよ）と、１つまたは複数の以前にレンダリングされたウェブ・ページに関連してダウンロードされたスクリプトの再実行とに関連付けられている、ブラウザ・オーバーヘッドを削減する、または排除する、ウェブ・ページまたは他の構造化文書をレンダリングするためのシステム、方法、および論理にさらに関する。特定の実施形態は、ブラウザまたは基礎となるクライアント・アプリケーションに、全体のウェブ・ページを不必要にネイティブに再レンダリングさせずに、ターゲット・ウェブ・ページをレンダリングするのに必須である新しいコンテンツおよびリソースのみを要求するための、さまざまな技法を利用する。

【０００６】

さまざまな実施形態において、キャッシュされたリソースが、キャッシュに記憶された後に変更されている場合、１つまたは複数の増分更新が、キャッシュされたリソースをリフレッシュするために生成されてよい。

【０００７】

さまざまな実施形態において、キャッシュされたリソースが、キャッシュに記憶された後に変更されている場合、１つまたは複数の新しい置換リソースが、キャッシュされたリソースをリフレッシュするために生成されてよい。

【図面の簡単な説明】

【０００８】

【図１】例示的なネットワーク環境を示す図。

【図２】図１の例示的なネットワーク環境の例示的なコンポーネントのブロック図。

【図３】ウェブ・ページを求める要求をサーブするための例示的な方法を示すフローチャート。

【図４Ａ】ウェブ・ページのためのページレットをサーブするための例示的な方法を示すフローチャート。
【図４Ｂ】ページレットをリフレッシュするための例示的な方法を示すフローチャート。
【図４Ｃ】ページレットをリフレッシュするためのさらなる例示的な方法を示すフローチャート。
【図５】ページレットをレンダリングするための例示的な方法を示すフローチャート。
【図６】例示的なコンピュータ・システム・アーキテクチャを示す図。
【発明を実施するための形態】
【０００９】
特定の実施形態は、リモートのおよび／またはローカルなデータ・ストアから、キャッシュされたリソースへの更新を取り出すための非同期の技法と連携して、キャッシュされたリソースを使用するウェブ・ページおよび他の構造化文書を効率的にレンダリングすることに関する。特定の実施形態は、コンテンツの再ローディング（サーバからリモートでアクセスされたにせよ、および／または、キャッシュからローカルにアクセスされたにせよ）と、１つまたは複数の以前にレンダリングされたウェブ・ページに関連してダウンロードされたスクリプトの再実行とに関連付けられている、ブラウザ・オーバーヘッドを削減する、または排除する、ウェブ・ページまたは他の構造化文書をレンダリングするためのシステム、方法、および論理にさらに関する。特定の実施形態は、ブラウザまたは基礎となるクライアント・アプリケーションに、全体のウェブ・ページを不必要にネイティブに再レンダリングさせずに、ターゲット・ウェブ・ページをレンダリングするのに必須である新しいコンテンツおよびリソースのみを要求するための、非同期ＪａｖａＳｃｒｉｐｔおよびＸＭＬ（ＡＪＡＸ：Ａｓｙｎｃｈｒｏｎｏｕｓ　ＪａｖａＳｃｒｉｐｔ　ａｎｄ　ＸＭＬ）技法を利用する。特定の実施形態は、クライアント・デバイス上で動作するブラウザのコンテキスト内で実行され得るプロセスに関し、プロセスは、ウェブ・ページを求めるブラウザからの要求を遮り、１つまたは複数のキャッシュされたリソースにアクセスし、１つまたは複数のキャッシュされたリソースへの増分更新を求める要求を送信して、ウェブ・ページを更新された形式でレンダリングする。さまざまな例示的な実施形態において、１つまたは複数の説明されるウェブ・ページは、ソーシャル・ネットワーキング・システム、またはウェブ・サイト内で実行されるソーシャル・ネットワーキング・サービスに関連付けられていてよい。しかしながら、開示される主題の実施形態は、任意のタイプのネットワーク・アドレス指定可能なリソースまたはウェブ・サイトによってホストされる、構造化文書の取り出しおよびレンダリングへの適用を有する。本明細書で使用されるとき、「ユーザ」は、個人、グループ、またはエンティティ（ビジネスまたはサードパーティ・アプリケーションなど）であってよい。
【００１０】
特定の実施形態は、多数のネットワーク・アドレス指定可能なシステムを含む、インターネットなどのワイド・エリア・ネットワーク環境において動作することができる。図１は、さまざまな例示的な実施形態がその中で動作することができる例示的なネットワーク環境を示す。ネットワーク・クラウド６０は、一般に、本明細書で説明されるシステムおよびホストがその上で通信することができる、１つまたは複数の相互接続されたネットワークを表現する。ネットワーク・クラウド６０は、パケット・ベースのワイド・エリア・ネットワーク（インターネットなど）、プライベート・ネットワーク、無線ネットワーク、衛星ネットワーク、セルラー・ネットワーク、ページング・ネットワーク、その他を含む。図１が示すように、特定の実施形態は、ソーシャル・ネットワーキング・システム２０、および１つまたは複数のクライアント・デバイス３０を備えるネットワーク環境において動作することができる。クライアント・デバイス３０は、ネットワーク・サービス・プロバイダ、無線キャリア、または任意の他の好適な手段を通じて、ネットワーク環境に動作可能に接続される。
【００１１】
例示的な一実施形態において、ソーシャル・ネットワーキング・システム２０は、本明

細書で説明されるように、ユーザが、互いに通信する、または別のやり方で対話する、およびユーザ・プロフィールなどのコンテンツにアクセスするのを可能にするコンピューティング・システムを備える。ソーシャル・ネットワーキング・システム２０は、さまざまな例示的な実施形態において、１つまたは複数の物理サーバ２２と、データ・ストア２４とを備える、ネットワーク・アドレス指定可能なシステムである。１つまたは複数の物理サーバ２２は、例として、一組のルータおよび／またはネットワーキング・スイッチ２６を通じて、コンピュータ・ネットワーク６０に動作可能に接続される。例示的な一実施形態において、１つまたは複数の物理サーバ２２によってホストされる機能は、ウェブ・サーバまたはＨＴＴＰサーバ、ＦＴＰサーバ、ならびに、限定されるものでないが、コモン・ゲートウェイ・インターフェース（ＣＧＩ：Ｃｏｍｍｏｎ Ｇａｔｅｗａｙ Ｉｎｔｅｒｆａｃｅ）スクリプト、ＰＨＰハイパー・テキスト・プリプロセッサ（ＰＨＰ）、アクティブ・サーバ・ページ（ＡＳＰ）、ハイパー・テキスト・マークアップ言語（ＨＴＭＬ）、拡張可能なマークアップ言語（ＸＭＬ）、Ｊａｖａ（登録商標）、ＪａｖａＳｃｒｉｐｔ、非同期ＪａｖａＳｃｒｉｐｔおよびＸＭＬ（ＡＪＡＸ）、その他を使用して実装されるウェブ・ページおよびアプリケーションを含む。

【００１２】

物理サーバ２２は、ソーシャル・ネットワーキング・システム２０の動作を対象とした機能をホストすることができる。例として、ソーシャル・ネットワーキング・システム２０は、１つまたは複数のクライアント・デバイス３０における１人または複数のユーザが、情報を見る、および投稿するだけでなく、そのウェブ・サイトを通じて互いに通信することを可能にする、ウェブ・サイトをホストする。サーバ２２は、たとえばソーシャル・ネットワーキング・システム２０をホストする、非常に多くのサーバ、ならびに他のコンテンツ配信サーバ、データ・ストア、およびデータベースを含むが、これ以降これらのサーバ２２をサーバ２２と称す。データ・ストア２４は、ソーシャル・ネットワーキング・システムの動作に関する、およびソーシャル・ネットワーキング・システムの動作をイネーブルにする、コンテンツおよびデータを、デジタル・データ・オブジェクトとして記憶することができる。データ・オブジェクトは、特定の実装形態において、通常、データ・ファイル、データベース、またはレコードにおいて記憶された、または具体化されたデジタル情報のアイテムである。コンテンツ・オブジェクトは、テキスト（たとえば、ＡＳＣＩＩ、ＳＧＭＬ、ＨＴＭＬ）、画像（たとえば、ｊｐｅｇ、ｔｉｆ、およびｇｉｆ）、グラフィックス（ベクトル・ベースまたはビットマップ）、オーディオ、動画（たとえば、ｍｐｅｇ）、または他のマルチメディア、およびそれらの組合せを含む、多くの形式を取ることができる。コンテンツ・オブジェクト・データはまた、実行可能なコード・オブジェクト（たとえば、ブラウザ・ウインドウまたはフレーム内で実行可能なゲーム）、ポッドキャスト、その他を含む。論理的には、データ・ストア２４は、リレーショナル・データベースおよびオブジェクト指向データベースなどの、多様な別個のデータベースおよび統合データベースのうちの１つまたは複数に対応し、データベースは、１つまたは複数の物理システムに記憶された論理的に関係したレコードまたはファイルの統合された集合として、情報を維持する。構造的には、データ・ストア２４は、一般に、大規模クラスのデータ・ストレージおよびマネジメント・システムのうちの１つまたは複数を含む。特定の実施形態において、データ・ストア２４は、１つまたは複数のデータベース・サーバ、マス・ストレージ・メディア、メディア・ライブラリ・システム、ストレージ・エリア・ネットワーク、データ・ストレージ・クラウド、その他などのコンポーネントを含む、任意の好適な物理システムによって実装されてよい。例示的な一実施形態において、データ・ストア２４は、１つまたは複数のサーバ、データベース（たとえば、ＭｙＳＱＬ）、および／またはデータ・ウェアハウスを含む。

【００１３】

データ・ストア２４は、ソーシャル・ネットワーキング・システム２０の異なるユーザおよび／またはクライアント・デバイス３０に関連付けられているデータを含む。特定の実施形態において、ソーシャル・ネットワーキング・システム２０は、システム２０のユ

ーザごとにユーザ・プロフィールを維持する。ユーザ・プロフィールは、ソーシャル・ネットワークのユーザを記述するデータを含み、データは、たとえば、正式な名前（人の名前、ミドルネーム、および名字、商標、ならびに／または、ビジネス・エンティティの会社名、その他）、職歴、学歴、趣味または選好、地理的なロケーションなどの、経歴の、人口統計の、および他のタイプの記述的な情報、ならびに追加的な記述的データを含む。例として、ユーザ・プロフィールは、ユーザの誕生日、交際状態、居住市、その他を含む。システム２０は、異なるユーザ間の１つもしくは複数の関係、または接続を記述するデータをさらに記憶する。関係情報は、同様の、または共通の職歴、グループ・メンバーシップ、趣味、または学歴を有するユーザを示すことができる。ユーザ・プロフィールはまた、他のユーザに対するユーザの情報へのアクセスを左右するプライバシー設定を含む。

【００１４】

クライアント・デバイス３０は、一般に、コンピュータ・ネットワーク上で通信する（たとえば、リモートで）ための機能を含むコンピュータまたはコンピューティング・デバイスである。クライアント・デバイス３０は、好適なコンピューティング・デバイスの中でもとりわけ、デスクトップ・コンピュータ、ラップトップ・コンピュータ・携帯情報端末（ＰＤＡ）、車内もしくは車外のナビゲーション・システム、スマート・フォンもしくは他のセルラー・フォンもしくはモバイル・フォン、またはモバイル・ゲーム・デバイスであってよい。クライアント・デバイス３０は、ウェブ・ブラウザ（たとえば、マイクロソフト・ウィンドウズ（登録商標）インターネット・エクスプローラ、モジラ・ファイアフォックス、アップル・サファリ、グーグル・クローム、およびオペラ、その他）などの、１つまたは複数のクライアント・アプリケーションを実行して、コンピュータ・ネットワーク上でコンテンツにアクセスし、コンテンツを見ることができる。特定の実装形態において、クライアント・アプリケーションは、クライアント・デバイス３０のユーザが、ソーシャル・ネットワーキング・システム２０によってホストされるリソースなどの、取り出されるべき固有のネットワーク・リソースのアドレスを入力することを可能にする。これらのアドレスは、ユニフォーム・リソース・ロケータ、すなわち、ＵＲＬであってよい。その上、ページまたは他のリソースが取り出されると、クライアント・アプリケーションは、ユーザが他のリソースへのハイパーリンク上を「クリックする」ときに、他のページまたはレコードへのアクセスを提供することができる。例として、そのようなハイパーリンクは、ウェブ・ページ内に配置されていてよく、ユーザに別のページのＵＲＬを入力しそのページを取り出す自動化されたやり方を提供する。

【００１５】

それ自体が多数の埋め込まれたリソースを含むことがある、ウェブ・ページ、またはウェブ・ページ内に埋め込まれたリソースは、データ・レコードを含むことができ、データ・レコードは、たとえば、プレーン・テキスト情報、または、ソフトウェア・プログラム、もしくは他のコード・オブジェクト、グラフィックス、画像、オーディオ信号、動画、その他などの、より複雑なデジタルでエンコードされたマルチメディア・コンテンツである。ウェブ・ページを作成するための１つの普及したマークアップ言語が、ハイパー・テキスト・マークアップ言語（ＨＴＭＬ）である。他の汎用ウェブ・ブラウザにサポートされる言語および技術は、拡張可能なマークアップ言語（ＸＭＬ）、拡張可能なハイパー・テキスト・マークアップ言語（ＸＨＴＭＬ）、ＪａｖａＳｃｒｉｐｔ、カスケーディング・スタイル・シート（ＣＳＳ）、およびしばしば、Ｊａｖａを含む。例として、ＨＴＭＬは、ページ内に埋め込まれ得る、テキストおよびリンク、ならびに画像、ウェブ・アプリケーションおよび他のオブジェクトのための構造セマンティクスを表すことによって構造化文書を作成するように、ページ・デベロッパーをイネーブルにする。一般に、ウェブ・ページは、静的文書としてクライアントに配布されてよい。しかしながら、ページに埋め込まれたウェブ要素の使用を通して、対話型の経験が、ページまたはページのシーケンスにより達成され得る。クライアントにおけるユーザ・セッションの間、ウェブ・ブラウザは、そのページをホストするウェブ・サイトから受信された、または取り出されたページおよび関連付けられているリソースだけでなく、潜在的には、他のウェブ・サイトからの

リソースを、解釈し、表示する。

【００１６】

より具体的には、ＨＴＭＬは、ウェブ・アプリケーション、画像、または動画を含むオブジェクトまたはリソースを、ウェブ・ページなどの構造化文書内に埋め込むように、デベロッパーをイネーブルにする。一般に、ＨＴＭＬ構造化文書は、構造化文書コンテンツ内でタグ（山括弧で囲まれた）からなるＨＴＭＬ要素の形式で書かれ、タグは、文書がウェブ・ブラウザによってどのように解釈されるべきか、かつ最終的にはユーザのディスプレイにどのように提示されるべきかについて、構造化文書をレンダリングするウェブ・ブラウザへの標識の役割をする。例として、ＨＴＭＬ要素は、ヘッディング、パラグラフ、ハイパーテキスト・リンク、埋め込みメディア、および多様な他の構造体を表現する。ＨＴＭＬは、従来のウェブ・ブラウザなどのＨＴＭＬプロセッサの挙動に影響するＪａｖａＳｃｒｉｐｔなどの言語のスクリプト、ならびに、テキストおよび他のコンテンツの見た目およびレイアウトを定義するカスケーディング・スタイル・シート（ＣＳＳ）を含むか、またはロードする。ＨＴＭＬ要素は、ＨＴＭＬのための基本的なコンポーネントであり、２つの基礎プロパティ、すなわち、属性およびコンテンツを有する。各要素である属性およびコンテンツは、ＨＴＭＬ要素が有効であるとみなされるために従わなければならない、一定の制約を有する。ＨＴＭＬ要素は、通常、開始タグ（たとえば、〈要素名〉）、および終了タグ（たとえば、〈／要素名〉）を有する。要素の属性は、開始タグに保有され、コンテンツは、タグとタグの間に配置される（たとえば、〈要素名　属性＝“値”〉コンテンツ〈／要素名〉）。

【００１７】

例として、ＨＴＭＬ要素は、構造要素（たとえば、テキストまたは他のコンテンツの目的を記述する）、提示要素（たとえば、その機能に関わらず、テキストまたは他のコンテンツの見た目を記述する）、およびハイパーテキスト要素（たとえば、文書の一部を別の文書へのリンクにする）を含む。ほとんどの要素は、いくつかの共通の属性のうちのいずれかを取ることができる。例として、ｉｄ属性は、文書規模の一意の識別子を要素に提供し、クラス属性は、同様の要素を分類するやり方を提供し、タイトル属性は、サブテキストの説明を要素に結び付けるために使用される。ＨＴＭＬはまた、スクリプト・データおよびスタイルシート・データなどの、要素コンテンツについてのいくつかのデータタイプ、ならびに、例としてはＩＤ、名前、ＵＲＩもしくはＵＲＬ、番号、長さの単位、言語、メディア記述子、色、文字エンコーディング、日付および時間、その他を含む、属性値についての非常に多くのタイプを定義する。

【００１８】

文書構造要素は、ルート要素（開始タグ〈ｈｔｍｌ〉および終了タグ〈／ｈｔｍｌ〉によってそれぞれ定義される）、ヘッド要素（開始タグ〈ｈｅａｄ〉および終了タグ〈／ｈｅａｄ〉によってそれぞれ定義される）、およびボディ要素（開始タグ〈ｂｏｄｙ〉および終了タグ〈／ｂｏｄｙ〉によってそれぞれ定義される）を含む。ルート要素タグ〈ｈｔｍｌ〉および〈／ｈｔｍｌ〉は、それぞれ、ＨＴＭＬ文書の冒頭および終了の範囲を定める。所与のＨＴＭＬ文書のすべての他のＨＴＭＬ要素は、ルート要素内に含まれる。ヘッド要素タグ〈ｈｅａｄ〉および〈／ｈｅａｄ〉は、一般に、ＨＴＭＬ文書についての情報およびメタデータを処理するためのコンテナを定義する。ヘッド要素コンテナ内に見出される例示的な文書ヘッド要素は、限定としてではなく例として、ＨＴＭＬ文書におけるすべての相対ｈｒｅｆおよび他のリンクについてのベース・ユニフォーム・リソース・ロケータ（ＵＲＬ）を指定するベース要素（開始タグ〈ｂａｓｅ〉および終了タグ〈／ｂａｓｅ〉によってそれぞれ定義される）、他の文書（たとえば、外部ＣＳＳファイルのための）へのリンクを指定するリンク要素（開始タグ〈ｌｉｎｋ〉および終了タグ〈／ｌｉｎｋ〉によってそれぞれ定義される）、ＨＴＭＬ文書についての追加的なメタデータを指定するために使用されてよいメタ要素（開始タグ〈ｍｅｔａ〉および終了タグ〈／ｍｅｔａ〉によってそれぞれ定義される）、文書ヘッダ内に汎用オブジェクトを含めるために使用されるオブジェクト要素（開始タグ〈ｏｂｊｅｃｔ〉および終了タグ〈／ｏｂｊｅｃｔ〉に

よってそれぞれ定義される）、スクリプト命令（たとえば、ＪａｖａＳｃｒｉｐｔ）のためのコンテナ、またはｓｒｃ（ソース）属性を有する外部スクリプトへのリンクとしての役割をするスクリプト要素（開始タグ〈ｓｃｒｉｐｔ〉および終了タグ〈／ｓｃｒｉｐｔ〉によってそれぞれ定義される）、文書のためのスタイルを指定し、スタイル命令（たとえば、インラインＣＳＳルールのための）のためのコンテナとしての役割をするスタイル要素（開始タグ〈ｓｔｙｌｅ〉および終了タグ〈／ｓｔｙｌｅ〉によってそれぞれ定義される）、ならびに、文書タイトルを定義するタイトル要素（開始タグ〈ｔｉｔｌｅ〉および終了タグ〈／ｔｉｔｌｅ〉によってそれぞれ定義される）を含む。

【００１９】

ボディ要素〈ｂｏｄｙ〉は、ＨＴＭＬ文書の表示可能なコンテンツのためのコンテナを表現する。例示的なボディ要素は、限定としてではなく例として、ブロック要素（たとえば、とりわけ、基本的なテキスト要素およびリスト要素）、インライン要素（たとえば、アンカー要素およびフレーズ要素）、および画像要素およびオブジェクト要素を含む。ボディ要素内に位置付けられるスクリプト要素は、文書にスクリプトを置くために使用されてよい（たとえば、スクリプト要素は、ブロックまたはインライン・コンテンツを動的に生成するための命令を保有することができる）。画像要素（開始タグ〈ｉｍｇ〉および終了タグ〈／ｉｍｇ〉によってそれぞれ定義される）は、文書の中に画像を挿入するために使用されてよい。例として、画像要素は、画像が配置されるＵＲＬを指定するｓｒｃ属性を含む。オブジェクト要素（開始タグ〈ｏｂｊｅｃｔ〉および終了タグ〈／ｏｂｊｅｃｔ〉によってそれぞれ定義される）は、含まれるタイプの属性において指定されたタイプの文書の中にオブジェクトを挿入するために使用されてよい。別の頻繁に使用されるＨＴＭＬ要素は、ボディ要素の代替として使用されてよいフレームセット要素である。

【００２０】

一般に、ウェブ・アプリケーションは、ネットワーク上でウェブ・ブラウザもしくは他のクライアント・アプリケーションを通じてアクセスされ得るアプリケーション、または、ウェブ・ブラウザにサポートされる言語においてコード化され、実行可能なアプリケーションをレンダリングするためにウェブ・ブラウザに依存するコンピュータ・ソフトウェア・アプリケーションである。ウェブ・アプリケーションは、主として、ウェブ・ブラウザの遍在性、クライアント（ときに、シン・クライアントと称す）としてのリモート・コンピューティング・デバイスで起動されたウェブ・ブラウザを使用する利便性、およびソフトウェアを配信してリモート・クライアント上にインストールすることなく、ウェブ・アプリケーションを更新し、維持する対応する能力の結果として、大衆性を得てきた。しばしば、ウェブ・アプリケーションを実装するために、ウェブ・アプリケーションは、関連付けられたウェブ・サイトのバックエンド・サーバにおいて提供される１つまたは複数のリソースへのアクセスを要する。加えて、ウェブ・アプリケーションは、しばしば、他のアプリケーションに関連付けられている追加的なリソースへのアクセスを要する。

【００２１】

ソーシャル・ネットワーキング・システム２０は、リモート・クライアント３０のユーザが、ユーザ・セッションの間にそれを用いて対話することができる、数多くの特徴を含む。特定の実施形態において、これらの特徴は、ウェブ・アプリケーションとして実装されてよく、サーバ２２、ならびに他の外部サーバまたはデータ・ストアに要求されたＪａｖａＳｃｒｉｐｔリソースおよびＣＳＳリソースを利用することができる。ウェブ・アプリケーションまたはリソースは、フレームもしくはｉＦｒａｍｅ、セクションもしくは「ｄｉｖ」、その他などにおいて、リモート・クライアントにサーブされるさまざまなウェブ・ページに埋め込まれてよい。特定の実施形態において、ソーシャル・ネットワーキング・システム２０は、ユーザがソーシャル・ネットワーキング・システム２０にアクセスしている間にそれを用いて対話することができる、異なる種類のアイテムのためのいくつかのオブジェクトを、データ・ストア２４に維持する。例示的な一実施形態において、これらのオブジェクトは、ユーザ・プロフィール、アプリケーション・オブジェクト、およびメッセージ・オブジェクト（ウォール投稿、電子メール、および他のメッセージのため

など）を含む。一実施形態において、オブジェクトは、その関連付けられたアイテムのインスタンスごとに、システム２０によって記憶される。本明細書で論じられるこれらのオブジェクトおよびアクションは、例証目的のみのために提供され、無限の数の変形および特徴が、ソーシャル・ネットワーキング・システム２０上で提供され得ることが認識されてよい。

【００２２】

上で説明されたように、ウェブ・ページまたは基礎となる構造化文書は、論理的に、視覚的に、または別のやり方で、セグメント化されてよい、またはセクションに分離されてよい。例として、ウェブ・ページをエンコードするのに使用される構造化文書は、開始および終了ＨＴＭＬ〈ｄｉｖ〉タグによって表される、１つまたは複数のブロック・レベル要素を含む。バックグラウンドとして、汎用ウェブ・ブラウザの場合、ウェブ・ページの表示可能な要素は、ブロックまたはインラインのいずれかとしてレンダリングされてよい。すべての要素は文書シーケンスの一部であると同時に、ブロック要素は、その親要素内で、行をまたいで分かれない矩形のオブジェクトのように見え、周囲の要素とは独立して設定され得る、ブロック・マージン、ならびに幅および高さのプロパティを有する。反対に、インライン要素は、文書テキストのフローの一部として扱われる。インライン要素は、マージン、幅、または高さ設定を有することはできず、行をまたいで分かれる。インライン要素は、ボディ要素の内部に直接置かれることはできない。インライン要素は、ブロック・レベル要素内に完全にネストされなければならない。

【００２３】

別の例として、ウェブ・ページはまた、ＨＴＭＬフレーム要素（たとえば、開始タグ〈ｆｒａｍｅ〉および終了タグ〈／ｆｒａｍｅ〉によってそれぞれ表される）を使用して構造的に指定されるように、１つまたは複数のフレームに分割されてもよい。フレームは、ウェブ・ブラウザ表示ウィンドウがセグメントに分割されるのを可能にし、セグメントの各々は、異なる文書を見せることができる。別のフレーム要素は、インライン・フレーム要素（開始タグ〈ｉｆｒａｍｅ〉および終了タグ〈／ｉｆｒａｍｅ〉によってそれぞれ表す）である。インライン・フレームは、別のＨＴＭＬ構造化文書をフレーム内に置く。オブジェクト要素とは異なり、インライン・フレームは、他の要素によって定義されるリンクのための「ターゲット」フレームであってよい。

【００２４】

以下では、とりわけ、ブロック・レベル要素、フレーム、またはインライン・フレームなどの、ウェブ・ページの、またはウェブ・ページをエンコードするために使用される構造化文書の任意の論理的、構造的、または視覚的なセクションもしくは部分を、以下では各々「ページレット」と称す。

【００２５】

クライアント・デバイス（たとえば、クライアント・デバイス３０）のユーザが、少なくとも部分的にソーシャル・ネットワーキング・システム２０によってホストされる特定のウェブ・ページ（以下、「ターゲット構造化文書」とも称す）を見たいと所望するとき、ユーザのウェブ・ブラウザ、または他のクライアント・サイドの文書レンダリング・エンジン、または好適なクライアント・アプリケーションは、要求を作って、ソーシャル・ネットワーキング・システム２０に送信する。要求は、一般に、ＵＲＬまたは他の文書識別子、ならびにメタデータまたは他の情報を含む。例として、要求は、ユーザＩＤなどの、ユーザを識別する情報、ならびに、ユーザのクライアント・コンピューティング・デバイス３０上で稼働するウェブ・ブラウザまたはオペレーティング・システムを識別する、または特徴付ける情報を含む。要求はまた、ユーザのクライアント・デバイスの地理的なロケーション、またはユーザのクライアント・デバイスの論理ネットワークロケーションを識別する、ロケーション情報を含む。要求はまた、要求がいつ送信されたかを識別するタイムスタンプを含む。

【００２６】

次に、ウェブ・ページを求める要求をサーブするための方法を、図２のブロック図およ

び図３のフローチャートを参照して説明する。例示的な一実施形態において、方法は、３０２で、少なくとも部分的にソーシャル・ネットワーキング・システム２０によってホストされるウェブ・ページを求める、クライアント・デバイス３０でのクライアント・アプリケーションからの要求を、ソーシャル・ネットワーキング・システム２０におけるサーバ２２または他のコンピューティング・システムが受信することにより始まる。上で説明されたように、要求は、一般に、ウェブ・ページ・ロケーションに対応したＵＲＬまたは他の文書識別子、およびメタデータまたは他の情報を含む。例として、要求は、ユーザＩＤなどの、要求を行うクライアント・アプリケーションのユーザを識別する情報、ならびにユーザのクライアント・コンピューティング・デバイス３０上で稼働するクライアント・サイドの文書レンダリング・アプリケーション（たとえば、ウェブ・ブラウザ）２０２またはオペレーティング・システムを識別する、または特徴付ける情報を含む。要求はまた、ユーザのクライアント・デバイスの地理的なロケーション、またはユーザのクライアント・デバイスの論理ネットワークロケーションを識別する、ロケーション情報を含む。要求はまた、要求がいつ送信されたかを識別するタイムスタンプを含む。認証プロセス２０４が、３０４で、要求を行うユーザが、要求されたウェブ・ページを受信するために認可されるか否か（たとえば、ユーザがログインに成功しているか、およびユーザは、ユーザがアクセスまたは管理権限を有するページを要求しているか）を、最初に決定する。

【００２７】

特定の実施形態において、ウェブ・ページ・サービング・プロセス２０６が次いで、要求を分析し、３０６で、マークアップ言語コード（たとえば、ＨＴＭＬ、ＸＭＬ、または他の好適なマークアップ言語コード）を使用して、ベース（骨組み）構造化文書を生成する。特定の実施形態において、ベース構造化文書は、ヘッド要素（たとえば、ＨＴＭＬヘッド要素）と、ボディ要素（たとえば、ＨＴＭＬボディ要素）とを含む。特定の実施形態において、ヘッド要素は、要求するクライアント３０においてページ・アセンブル・プロセス２０８を実装するための実行可能なコード・セグメントを含む。特定の実施形態において、ページ・アセンブル・プロセス２０８を実装するための実行可能なコード・セグメントは、ＪａｖａＳｃｒｉｐｔコード・セグメントであり、ＪａｖａＳｃｒｉｐｔ関数ライブラリを含む。ヘッド要素はまた、１つもしくは複数の初期リソース（たとえば、ＪａｖａＳｃｒｉｐｔまたはＣＳＳ）、または外部ロケーション（たとえば、サードパーティ・サイト）からダウンロードされる、そのようなリソースへの参照（たとえば、選択されたリソースの位置を突き止めるための対応するソース（ｓｒｃ）識別子を有するスクリプト要素、画像要素、またはオブジェクト要素の形式での）を含む。特定の実施形態において、ベース構造化文書に含まれる初期リソースの数を最小限にすること（たとえば、要求するクライアント３０へのベース構造化文書の送信を早めるために）が望ましい。特定の実施形態において、ボディ要素は、要求するクライアント３０によって受信されると、１つまたは複数のプレース・ホルダをレンダリングするための、マークアップ言語コードを含む。特定の実施形態において、ページ・サービング・プロセス２０６は、以下でより詳細に説明されるように、対応する第２の応答においてクライアント３０に送信されることになる、１つまたは複数の（一般的には複数存在する）ページレットの各々のためのプレース・ホルダを、ベース構造化文書の中に追加する。ボディ要素はまた、要求するクライアント３０によって受信されると、レンダリングされることになるリソース、リソースへの参照、またはコンテンツさえも含む（しかしながら、特定の実施形態において、リソース、リソースへの参照、またはコンテンツは、ベース構造化文書において最小限にされる）。特定の実施形態において、３０８で、ウェブ・ページ・サービング・プロセス２０６は次いで、ベース構造化文書を含む初期応答または初期応答部分（以下では、「第１の応答」または「第１の応答部分」とも称す）を作って、要求するクライアント３０に送信する（いくつかの実施形態においては、初期／第１の応答／応答部分の送信に先立って、たとえば、プリ・フェッチする命令などの任意のデータが、クライアントに送信されることに留意されたい）。初期応答は、ＨＴＴＰ上で、または任意の他の好適な接続上で、要求するクライアント３０に送信されてよい。特定の実施形態において、初期応答がクライア

ント３０に送信される接続は、永続的な伝送制御プロトコル（ＴＣＰ）接続である。以下で説明するように、このことは、ベース構造化文書を受信し、要求されたウェブ・ページの残りが生成されている間に、ベース構造化文書の処理をし始め、初期応答におけるリソースを初期化する、またはダウンロードするように、クライアント３０をイネーブルにする。

【００２８】

特定の実施形態において、３０６でのベース構造化文書の生成、または３０８での初期応答の送信の後に、または少なくとも部分的にそれと並行してさえ、ウェブ・ページ・サービング・プロセス２０６は、３１０で、ページ生成プロセス２１０を起動し、ページ生成プロセス２１０に、要求されたウェブ・ページの残りを生成するように命令する。特定の一実施形態において、ページ生成プロセス２１０は、複数のサブページ生成プロセスを管理するマスタ・プロセスを備え、サブページ生成プロセスは、各々が、ウェブ・ページの割り当てられたサブ部分を生成するように構成されている。すなわち、特定の実施形態において、１つまたは複数のサブページ生成プロセスの各々は、他のサブページ生成プロセスと並行して、割り当てられたページレットを生成することを始める。特定の一実施形態において、各サブページ生成プロセスがその割り当てられたページレットの生成を完了するとき、ページ生成プロセス２１０は、そのページレットをページ・サービング・プロセス２０６に渡し、ページ・サービング・プロセス２０６は次いで、３１２で、二次応答または二次応答部分（以下では、「第２の応答」または「第２の応答部分」とも称す）としてページレットを作って、要求するクライアント３０に送信する。代替実施形態において、ページ・サービング・プロセス２０６は、ページ生成プロセス２１０を備えることができ、各二次部分を、イン・プロセスで順次生成することができ、それにより、一般に、１つまたは複数の他のページ生成プロセスを起動することを伴わなくてもよい。ステップ３１２は、単一のステップ３１２として説明されているが、連続して、または並行してクライアント３０に送られる、およびいくつかの実施形態では、各ページレットが生成されるとクライアント３０に送られる、複数の対応する二次応答における複数のページレットを作って、送信することを一般に含んでもよいことに留意されたい。上で説明されたように、各二次応答は、初期応答が送信されたのと同じ永続的なＴＣＰまたは他の好適な接続上で、クライアント３０に送信されてよい。一実施形態において、ベース構造化文書のボディ要素は、終了タグで閉じられていないことがあり、したがって、各二次応答は、後で生成され、後で送信されるベース構造化文書のボディ要素の一部として、送信されてもよい。別の実施形態において、二次応答のうちの１つまたは複数は、１つまたは複数の独立したＨＴＭＬ応答として、送信されてもよい。

【００２９】

特定の実施形態において、要求するクライアント３０、具体的には文書レンダリング・アプリケーション２０２（たとえば、ウェブ・ブラウザ）が初期応答を受信するとき、文書レンダリング・アプリケーション２０２は、初期応答およびベース構造化文書を処理し、ベース構造化文書のモデル表現をクライアント３０のメモリに生成する。例として、文書レンダリング・アプリケーション２０２は、ベース構造化文書の文書オブジェクト・モデル（ＤＯＭ）表現２１２を生成することができる。当業者によって認識されるように、文書レンダリング・アプリケーション２０２がＤＯＭ表現２１２を生成するとき、文書レンダリング・アプリケーション２０２は、マークアップ言語コードを、ＤＯＭ要素またはＤＯＭノードのＤＯＭツリーまたはＤＯＭ階層に本質的に変換し、ＤＯＭ要素またはＤＯＭノードの各々は、可能性の中でもとりわけ、リソース、リソースのための参照、コンテンツを含むか、または保有することができる。特定の実施形態において、文書レンダリング・アプリケーション２０２が、ベース構造化文書において指定されたプレース・ホルダの各々のために１つまたは複数のＤＯＭノードを生成するように、ベース構造化文書が構成される。以下では、これらのＤＯＭノードを「プレース・ホルダＤＯＭノード」と称す。一般に、各プレース・ホルダＤＯＭノードは、表示されることになるいずれのリソース、リソースのための参照、またはコンテンツも含まないが、以下でより詳細に説明される

ように、これらのプレース・ホルダＤＯＭノードは、文書レンダリング・アプリケーション２０２によってレンダリングされ、一般的に表示されるときに、レンダリングされたベース構造化文書における箇所を本質的に確保し、そこで、続いて受信される二次応答の中で、関連付けられたページレットにおいて続いて受信されるリソース、リソースのための参照、またはコンテンツが、表示される、初期化される、または実行されることになる。

【００３０】

特定の実施形態において、ページ・アセンブル・プロセス２０８を実装するためのコード・セグメント（たとえば、ＪａｖａＳｃｒｉｐｔ）は、文書レンダリング・アプリケーション２０２によって処理されると、初期化される。初期化されると、ページ・アセンブル・プロセス２０８は次いで、上で説明された、続いて受信される二次応答を待つことができる。初期応答に含まれる任意の他のリソース、またはリソースへの参照も、文書レンダリング・アプリケーション２０２によって受信されると、処理される、初期化される、実行される、またはダウンロードされる。

【００３１】

特定の実施形態において、二次応答の各々が文書レンダリング・アプリケーション２０２によって受信されるとき、ページ・アセンブル・プロセス２０８は、各対応するページレットのコンテンツを、キュー２１４（ページ・アセンブル・プロセス２０８または文書レンダリング・アプリケーション２０２によってアクセス可能な、一時的な、または他の好適なメモリロケーションで実装されてよい）に置くことができる。続いて、ページ・アセンブル・プロセス２０８は、受信された二次応答ごとに、ＤＯＭ表現２１２において対応する（１つまたは複数の）プレース・ホルダＤＯＭノードを識別し、対応箇所プレース・ホルダＤＯＭノードで、または対応箇所プレース・ホルダＤＯＭノードの代わりに、ページレットのコンテンツを、ＤＯＭ表現２１２の中に動的に挿入する、または交換することができる。特定の実施形態において、ページ・アセンブル・プロセス２０８は、文書レンダリング・アプリケーション２０２と連携して、ページレットのコンテンツを、ＤＯＭ表現２１２の中に動的に挿入する、または交換することができる。すなわち、ページ・アセンブル・プロセス２０８からの呼び出しに応答して、文書レンダリング・アプリケーション２０２が、コンテンツを実際に挿入する、または交換することができる。この挿入または交換は、ＤＯＭ表現２１２における追加的なＤＯＭノードの生成（たとえば、ページレットに対応したプレース・ホルダＤＯＭノードのレベルでの、またはそれ以下のレベルでの、ＤＯＭ表現２１２におけるより低次のＤＯＭノードの生成）を伴うことに留意されたい。各二次応答のコンテンツがＤＯＭ表現２１２の中に挿入されるとき、挿入されたコンテンツまたはリソースは次いで、文書レンダリング・アプリケーション２０２によるレンダリングのために、レンダリングされ、実行され、初期化され、またはダウンロードされて、ユーザに表示される。

【００３２】

特定の実施形態において、デバイス・キャッシュ・コンポーネント２２０は、たとえば、クライアント３０の一時的なメモリまたは永続的なストレージ内で、文書レンダリング・アプリケーション２０２によって受信されたページレットをキャッシュすることになり、その結果、以下でより詳細に論じられるように、ユーザが、たとえば、現在レンダリングされているウェブ・ページを再ロードする、または、現在レンダリングされているウェブ・ページから離れ、別の１つまたは複数のウェブ・ページへとナビゲートして、次いで以前にレンダリングされたウェブ・ページに戻るのを所望するときに、これらのリソースのうちの１つまたは複数が利用され得る。いくつかの実施形態において、デバイス・キャッシュ・コンポーネント２２０は、ページレットがキャッシュ可能であるか否か、ページレットが以前にキャッシュされているか否か、ページレットはいつ以前にキャッシュされたか、および／または、キャッシュされたページレットがサーバ２２上で更新されており、したがってクライアント３０上で更新される予定であってよいか否かなどの、情報を記憶することができる。

【００３３】

特定の実施形態において、各ページレットは、クライアント３０によって受信されると、ページ・アセンブル・プロセス２０８を呼び出すように構成された関数呼び出しを含むか、または関数呼び出しのフォーマットである。特定の実施形態において、ページ・アセンブル・プロセス２０８は、続いて受信される二次応答における関数呼び出しによって呼び出されることになるか、または呼び出されうる（たとえば、ＪａｖａＳｃｒｉｐｔ）関数を含む関数ライブラリ（たとえば、ＪａｖａＳｃｒｉｐｔ関数ライブラリ）を含む。特定の実施形態において、各ページレットは、対応する関数呼び出しを実装するためのスクリプトを含む任意のマークアップ言語コード（たとえば、ＨＴＭＬ）を含み、それが、ページ・アセンブル・プロセス２０８の関数ライブラリによって要求される関数を識別する。特定の実施形態において、各関数呼び出しは、例として、ＨＴＭＬまたは他のマークアップ言語コード、ＪａｖａＳｃｒｉｐｔリソースのアレイ、ＣＳＳリソースのアレイ、および、ページレットが表示され、ページレットにおいて識別されたリソースがダウンロードされた後で実行されることになる、（たとえば、ＪａｖａＳｃｒｉｐｔ）スクリプトまたはコード・セグメントのリストを含むレジスタを含む、１つまたは複数のコールバック関数にラッピングされている１つまたは複数の引数を含む。

【００３４】

特定の実施形態において、各関数呼び出しは、ページレットのいくつかのプロパティを記述するいくつかのメタデータをさらに含み、それらはページレットの相対優先度を決定するために、ページ・アセンブル・プロセス２０８によって使用されてよい。これは、二次応答において受信されたページレットの各々の相対表示順序をスケジュールするように、ページ・アセンブル・プロセス２０８をイネーブルにする。例として、いくつかのページレットは、クリティカルではないこと、または他の特徴もしくはコンテンツほどクリティカルではないことを予め決定された特徴をイネーブルにすることができ、したがって、これらのページレットは、他のよりクリティカルなページレットの後でレンダリングされてよい。単純な一実装形態において、各関数呼び出しは、「遅延フラグ」をさらに含む。この遅延フラグが真に設定されるとき、これは、ページ・アセンブル・プロセス２０８に、そのページレットが相対的にクリティカルでないこと、ならびに、ＤＯＭ表現２１２の中へのその挿入および文書レンダリング・アプリケーション２０２による続くレンダリングが、他のより高い優先度ページレットが挿入されてレンダリングされるまで、遅延されてもよいことを命令する。より具体的には、かつ以下でより詳細に説明されるように、各ページレット・コールバック関数は、クライアント３０で受信されるとき、他のより高い優先度ページレット・コールバック関数が処理される、または実行されるまで、キュー２１４に保持されてよい。しかしながら、いくつかの実施形態では、より高い優先度ページレットがキュー２１２に存在しない場合、より低い優先度ページレット・コールバック関数が処理されてよい。

【００３５】

特定の実施形態において、上で説明されたように、二次応答において受信された特定のページレットのコンテンツをＤＯＭ表現２１２の中に挿入するために、ページ・アセンブル・プロセス２０８は、ＤＯＭ表現２１２において、対応するプレース・ホルダＤＯＭノードを識別しなげればならない。特定の実施形態において、これは、ページ・サービング・プロセス２０６により、各プレース・ホルダを有する識別子（ＩＤ）パラメータを、ベース構造化文書に含めることによって達成される。同様に、ページ生成プロセス２１０によって生成された各ページレットは、ページレットのコンテンツが、その中に挿入されることになる、またはそこで交換されることになるＤＯＭ表現２１２において、一致する、または対応するプレース・ホルダＤＯＭノードを識別するために、ページ・アセンブル・プロセス２０８によって使用可能なＩＤパラメータを含む。構造化文書の生成および配布に関するさらなる詳細は、その全体において本願明細書に援用する、米国特許出願第１２／７５４，５４９号に見出されてよい。

【００３６】

次にページレットをサーブするための方法が、図４Ａのフローチャートを参照して説明

される。上で説明されたように、ウェブ・ページ要求に応答して、３０６でのベース構造化文書の生成、または３０８での初期応答の送信の後に、またはそれと並行して、ウェブ・ページ・サービング・プロセス２０６は、３１０で、ページ生成プロセス２１０を起動し、ページ生成プロセス２１０に、要求されたウェブ・ページの残りを生成するように命令する。特定の実施形態において、要求されたウェブ・ページが以前にレンダリングされている場合、およびしたがって、ウェブ・ページのレンダリングの使用のためにクライアント３０によってキャッシュされている、またはキャッシュに記憶されている対応するページレットを有する場合（「キャッシュ・ヒット」）、ウェブ・ページ・サービング・プロセス２０６は、クライアント３０に、キャッシュされたリソースを使用して、少なくとも部分的にページレットをレンダリングするように命令する。例として、図４Ａを参照すると、クライアント・デバイス３０が以前にレンダリングされているウェブ・ページを要求するとき、４０２で、ウェブ・ページ・サービング・プロセス２０６は、キャッシュ・ヒットが存在するか否かを最初に決定する。すなわち、要求されたウェブ・ページが以前にクライアント３０によってレンダリングされ、キャッシュに記憶されている場合、ウェブ・ページ・サービング・プロセス２０６は、ウェブ・ページのレンダリングに使用されることになる、キャッシュに記憶された１つもしくは複数のページレットまたはページレット・リソースを識別することができる。例示的な一実施形態において、キャッシュ・ヒットが存在しない場合、本明細書で説明されたように、ウェブ・ページ・サービング・プロセス２０６は、ページ生成プロセス２１０を起動し、ページ生成プロセス２１０に、要求されたウェブ・ページの残りを生成するように命令する。

【００３７】

特定の実施形態において、ウェブ・ページ・サービング・プロセス２０６は、ウェブ・ページを求める要求がいつサーバに送信されたか、１つまたは複数の応答がいつサーバ２２から提供されたか、および／または、ウェブ・ページがいつクライアント３０によってレンダリングされたかを示すタイムスタンプを記憶することができる。キャッシュに記憶された１つまたは複数のページレットが識別された後で、ウェブ・ページ・サービング・プロセス２０６は、キャッシュされたページレットのうちのいくつかまたはすべてが、期限切れである、したがって古すぎることを決定することができる。たとえば、４０４で、ウェブ・ページ・サービング・プロセス２０６は、ウェブ・ページを求める現在の要求のタイムスタンプと、ウェブ・ページを求めた以前の要求が行われたタイムスタンプとを比較して、現在の要求についてのタイムスタンプと以前の要求についてのタイムスタンプとの間で経過した時間が、予め決定されたしきい値時間を超えているか否かを決定することができる。予め決定されたしきい値時間は、任意の所望される時間であってよく、たとえば、限定されるものでないが、５分であってよい。

【００３８】

特定の実施形態において、システム・キャッシュ・コンポーネント２１８は、追加的なデータを書き込むなどの、デバイス３０のキャッシュに記憶されているページレットに実施される、１つまたは複数の状態変化動作を監視、および／または記録することができる。例として、システム・キャッシュ・コンポーネント２１８は、状態変化動作（一実施形態において、埋め込まれたスクリプトにおいてオブジェクトとして具体化される）を、動作が実施されたページレットについて記憶された対応するデータ・オブジェクト内に埋め込まれたスクリプトによって登録されたコールバック関数に関連付けて、記憶することができる。このようにして、ユーザが、動作が実施されたページからナビゲートして離れ、後で動作が実施されたページにナビゲートして戻るとき、システム・キャッシュ・コンポーネント２１８は、ウェブ・ページ生成２０６に、コールバック関数を二次応答部分としてクライアント３０に送るように命令することができ、次いでクライアント３０は、キャッシュされたページのコンテキストにおいて動作を再実行する。状態変化動作が以前に実施されているページについてのキャッシュ・ヒットに応答して、本明細書で論じられるように、ページ生成プロセス２０８は、対応するデータ・オブジェクトにアクセスし、ページについてのキャッシュされたリソースを再ロードする、または再実行することができる

。いくつかの実施形態において、ウェブ・ページ・サービング・プロセス２０６またはシステム・キャッシュ・コンポーネントは、受信された状態変化情報に少なくとも部分的に基づいて、キャッシュされたページレットが古すぎることを決定することができる。

【００３９】

特定の実施形態において、４０４で、キャッシュされたページレットが古すぎないことが決定された場合、４０６で、図５を参照して以下でさらに論じられるように、ウェブ・ページ・サービング・プロセス２０６は、キャッシュから、ページレットのうちのいくつかまたはすべてをレンダリングするようクライアント３０に命令する。たとえば、ウェブ・ページ・サービング・プロセス２０６は、ページレットをレンダリングするために使用されることになるリソースのうちのいくつかまたはすべてがキャッシュに記憶されていることを示す二次応答を、クライアント３０に送ることができる。

【００４０】

特定の実施形態において、４０８で、クライアント３０がキャッシュからページレットのうちのいくつかまたはすべてをレンダリングすると、図４Ｂおよび図４Ｃを参照して以下でさらに論じられるように、ウェブ・ページ・サービング・プロセス２０６は、ページ生成プロセス２１０に、ページレットをリフレッシュするように命令する。いくつかの実施形態において、ウェブ・ページ・サービング・プロセス２０６は、ページレットをリフレッシュするためのクライアント３０による要求に応答して、ページレットがリフレッシュされる。

【００４１】

特定の実施形態において、キャッシュされたページレットが古すぎることが決定された場合、４１０で、ウェブ・ページ・サービング・プロセス２０６は、ページ生成プロセスに、いずれのキャッシュされたリソースも利用せずに、新しいページレットとしてページレットを生成するように命令する。４１２で、ウェブ・ページ・サービング・プロセスは、たとえば、本明細書で論じられたように、クライアント３０への二次応答の一部としてページレットをキャッシュに送ることによって、キャッシュからではなく、４１０で生成されたページレットを使用してウェブ・ページをレンダリングするようクライアント３０に命令する。

【００４２】

次に、ページレットをリフレッシュするための例示的な方法が、図４Ｂのフローチャートを参照して説明される。ウェブ・ページ・サービング・プロセス２０６は、ページ生成プロセス２１０に、キャッシュから少なくとも部分的にレンダリングされているページレットを、リフレッシュするように命令する。例として、クライアント３０は、ページ、より具体的にはページレットが、クライアント３０によって最後にレンダリングされた後、変更されている対応するアプリケーションによって必要とされるリソースについて、増分更新を要求することができる。特定の実施形態において、ページレット関数呼び出しは、要求内の対応するアプリケーションをレンダリングするために必要とされるリソースについての対応するタイムスタンプを含む。特定の実施形態において、ウェブ・ページ・サービング・プロセス２０６および／またはシステム・キャッシュ・コンポーネント２１８は、対応するタイムスタンプに基づいて、リソースがクライアント３０に最後に送信され、レンダリングされた後、対応するリソースに対する更新があったか否かを決定することができる。

【００４３】

特定の実施形態において、４２０で、たとえば、要求と共に送信されたタイムスタンプ以降にそれらのリソースが変更している場合、ウェブ・ページ・サービング・プロセス２０６は、ページ生成プロセス２１０に、キャッシュされたページレットの１つまたは複数のリソースについて、クライアント３０への１つまたは複数の増分更新を生成するように命令する。すなわち、ウェブ・ページが一体としてみなされ、かつウェブ・ページにおけるいずれの変更もクライアント・サイドのキャッシュからページをフラッシュすることをもたらす、従来のキャッシング手順とは対照的に、個々のページレットが、一体のウェブ

・ページが再要求されることなく、その対応する関数呼び出しを通じて、増分更新を要求することができる。例として、リソースが変更されている、または更新されている場合、ページ生成プロセス２１０は、クライアント３０でキャッシュされたリソースを更新するために使用される増分データ（または、基礎となるデータへの変更を示す差分データ）を生成することができる。

【００４４】

特定の実施形態において、４２２で、ページ生成プロセス２１０からの更新されたリソース（存在する場合）の生成時に、ウェブ・ページ・サービング・プロセス２０６は、本明細書で論じられたように、たとえば二次応答部分の一部として、更新されたリソースをクライアント３０に送ることができる。４２４で、ウェブ・ページ・サービング・プロセス２０６は、更新されたリソースおよびキャッシュされたリソースを使用してページをレンダリングするようにクライアント３０に命令する。本明細書で論じられたように、たとえば、二次応答部分は、更新されたリソースを、ページ・アセンブル・プロセス２０８によってページの中に挿入するための命令を含む。例として、これは、更新されたＨＴＭＬまたは他のコンテンツを、対応するページレットまたはレンダリングされたページの他のセクションの中に挿入すること、または、ページのレンダリングに使用するために更新されたスクリプトを実行することを伴ってもよい。このようにして、キャッシュされたリソースは、更新されたリソースの生成前に、または少なくとも部分的にそれと並行して、レンダリングされ、ユーザに表示されてよい。更新されたリソースが受信されると、ページ・アセンブル・プロセスは、更新されたリソースを含むページレットをレンダリングすることができる。加えて、二次応答部分は、更新されたリソースを、クライアント３０のキャッシュにキャッシュするための命令を含む。

【００４５】

次に、ページレットをリフレッシュするためのさらなる例示的な方法が、図４Ｃのフローチャートを参照して説明される。キャッシュされたページレットのリソースのために１つまたは複数の増分更新を生成するのではなく、４２６で、ウェブ・ページ・サービング・プロセス２０６は、ページ生成プロセス２１０に、１つまたは複数の新しい置換リソースを生成して、キャッシュされたリソースを新しいバージョンと置換するように命令する。これは、本明細書で説明したように、要求と共に送信されたタイムスタンプ以降にキャッシュされたリソースが変更している場合、および／または、クライアント３０からの要求時に、キャッシュされたリソースがウェブ・ページをレンダリングするために使用される度に発生してよい。

【００４６】

特定の実施形態において、４２８で、ページ生成プロセス２１０からの新しい置換リソース（存在する場合）の生成時に、ウェブ・ページ・サービング・プロセス２０６は、本明細書で論じられたように、たとえば、二次応答部分の一部として、更新されたリソースをクライアント３０に送ることができる。４３０で、ウェブ・ページ・サービング・プロセス２０６は、新しい置換リソースを使用してページをレンダリングするようクライアント３０に命令する。本明細書で論じられたように、たとえば、二次応答部分は、新しい置換リソースを、ページ・アセンブル・プロセス２０８によってページの中に挿入するための命令を含む。例として、これは、対応するページレットもしくはレンダリングされたページの他のセクションのＨＴＭＬもしくは他のコンテンツを置換することによって、対応するページレットを置換すること、または、ページのレンダリングに使用するための置換スクリプトを実行することを伴ってもよい。このようにして、キャッシュされたリソースは、置換リソースの生成前に、または少なくとも部分的にそれと並行して、レンダリングされ、ユーザに表示されてよい。置換リソースが受信されると、ページ・アセンブル・プロセスは、置換リソースを使用してページレットをレンダリングすることができる。加えて、二次応答部分は、新しい置換リソースを、クライアント３０のキャッシュの中の対応するページレットの箇所にキャッシュするための命令を含む。

【００４７】

次に、ページレットをレンダリングするための方法が、図５のフローチャートを参照して説明される。特定の実施形態において、５０２で、ページレットが二次応答において受信されるとき、対応する関数呼び出しが、ページ・アセンブル・プロセス２０８を呼び出す。５０４で、ページ・アセンブル・プロセスは、単独で、またはデバイス・キャッシュ・コンポーネント２２０と通信して、受信されたページレットがキャッシュ可能なページレットであるか否かを決定するためのチェックを実施することができる。たとえば、関数呼び出しは、ページレットがキャッシュ可能なリソースを含むことを示すために、フラグなどの命令を含む。例として、キャッシュ可能なリソースは、ナビゲーション・メニュー、ニュース・フィード情報、または、ユーザがウェブ・ページを離れ、その後そのウェブ・ページに戻るときに一般に再ロードされる他のリソースなどの、リソースを含む。いくつかのページレットは、キャッシュ可能なリソースを含まないことがある。たとえば、いくつかのページレットでは、ユーザが同じページに戻る度に異なる広告をレンダリングすることが望ましいことがある、広告を表示するページレットの場合などの、ページがロードされる度に新しいリソースを要求することが望ましいことがある。

【００４８】

特定の実施形態において、５０４で、デバイス・キャッシュ・コンポーネント２２０が、ページレットがキャッシュ可能でないことを決定した場合、５０６で、ページ・レンダリング・コンポーネント２０８が、本明細書で説明されたように、ページレットをキャッシュせずに、かつページレットをレンダリングするためにキャッシュされたリソースを全く使用せずに、ページレットのレンダリングを進行することができる。

【００４９】

特定の実施形態において、５０４で、デバイス・キャッシュ・コンポーネント２２０が、ページレットがキャッシュ可能であることを決定した場合、５０８で、デバイス・キャッシュ・コンポーネント２２０は、受信されたページレットが以前にキャッシュされているか否かを決定するためのチェックを実施することができる。たとえば、関数呼び出しは、ページレットが以前にキャッシュされていることを示すために、フラグなどの命令を含む。加えて、または代替として、デバイス・キャッシュ・コンポーネント２２０は、各キャッシュされたページレットを、ページレットごとに一意の識別情報で登録し、一意の情報から、ページレットが以前にキャッシュされているか否かを決定することができる。関数呼び出しおよび／またはデバイス・キャッシュ・コンポーネント２２０はまた、ページレットがいつキャッシュされたかについての情報を含む。

【００５０】

特定の実施形態において、５０８で、デバイス・キャッシュ・コンポーネント２２０が、受信されたページレットが以前にキャッシュされていないことを決定した場合、デバイス・キャッシュ・コンポーネント２２０は、ページレットをクライアント３０のキャッシュに記憶することができる。例として、受信されたページレットをキャッシュに記憶することは、ページレットを、マス・ストレージ・ドライブ上などの、クライアント３０の永続的なメモリに記憶することを含む。受信されたページレットを記憶することはまた、ページレットを暗号化して、暗号化されたページレットをキャッシュに記憶することを含む。ウェブ・ブラウジング・セッションが終了した後、ページレット・キャッシュ・データが永続的なメモリの中にとどまるのであれば、ページレットを暗号化することは、ページレットをコピーすること、改ざんすること、またはリバース・エンジニアリングすることを防ぐことができる。ページレットを暗号化することは、暗号化ライブラリ、たとえば、限定されるものでないが、Ｓｔａｎｆｏｒｄ Ｊａｖａｓｃｒｉｐｔ Ｃｒｙｐｔｏ Ｌｉｂｒａｒｙを使用して、ページレット・データを暗号化することを含む。加えて、または代替として、ページレットを記憶することは、ページレットを、クライアント３０のシステム・メモリまたはオン・チップ・プロセッサ・キャッシュ・メモリなどの、一時的なメモリに記憶することを含む。５０６で、ページ・レンダリング・コンポーネント２０８は次いで、本明細書で説明されたように、ページレットのレンダリングを進行することができる。

【００５１】
特定の実施形態において、５０８で、デバイス・キャッシュ・コンポーネント２２０が、受信されたページレットが以前にキャッシュされていることを決定した場合、５１２で、デバイス・キャッシュ・コンポーネント２２０は、キャッシュされたページレット、またはキャッシュされたページレットの部分に対応したキャッシュされたリソースを、クライアント３０のキャッシュから取り出すことができる。キャッシュされたページレットが暗号化されている場合、キャッシュされたリソースを取り出すことは、ページ生成プロセス２０８によってキャッシュされたリソースが利用され得る前に、それらを解読することを含む。キャッシュされたリソースが受信されると、５０６で、ページ・レンダリング・コンポーネント２０８は、本明細書で説明されたように、キャッシュされたリソースを使用してページレットのレンダリングを進行することができる。

【００５２】
特定の実施形態において、ページレットをレンダリングした後に、または少なくとも部分的にそれと並行して、５１４で、デバイス・キャッシュ・コンポーネント２２０は、ページレットのスナップショットを取ることができる。そのようなスナップショットは、受信されたページレットごとに、ＤＯＭ情報および／または関数呼び出しデータのスナップショットを取ることを含む。スナップショットは、たとえば、ページレットを更新するための要求の一部として、サーバ２２に送られてよい。キャッシュされたリソースを使用してレンダリングされたページレットは、本明細書で論じられたように、ページレット・リソースがキャッシュされた後に追加されている新しい情報を含むように、リフレッシュされてよい。このようにして、キャッシュされたページレットは、レンダリングされ、ユーザに表示されてよく、キャッシュされたページレットをレンダリングした後に、または少なくとも部分的にそれと並行して、ページレットは、存在する場合、新しい情報を含むように、リフレッシュされてよい。したがって、ページレット・リソースが、キャッシュから取り出されるのではなく、すべてサーバからダウンロードされた場合に比べ、ユーザは、ページレットが早くロードされるのを体感することができる。

【００５３】
特定の実施形態において、５０６で、本明細書で論じられたように、受信されたページレットおよび／またはキャッシュされたリソースは、ページ・アセンブル・プロセス２０８に送られており、ページ・アセンブル・プロセス２０８は次いで、対応するページレット・コールバック関数をキュー２１４に入れることができ、キュー２１４は既に、以前の二次応答において受信された、以前に受信されたページレットに対応したページレット・コールバック関数を保有していることがある。ページレットをレンダリングすること、およびページレットをウェブ・ページの中に挿入することを含む、ページレット・コールバック関数を取り扱うことに関するさらなる詳細は、たとえば、その全体において本願明細書に援用する、米国特許出願第１２／７５４，５４９号に見出されてよい。

【００５４】
特定の実施形態において、デバイス３０のキャッシュは、定期的に、または一定のイベントの後で、消去されてよい。たとえば、デバイス・キャッシュ・コンポーネント２２０は、キャッシュを消去するためのユーザからの命令時に、キャッシュを消去するように構成されてよい。加えて、または代替として、デバイス・キャッシュ・コンポーネントは、ユーザがウェブ・サイトからログ・オフするとき、またはユーザがウェブ・サイトにログ・オンした後一定の時間期間が経過したとき、またはキャッシュが作成されたときなどの一定のイベントの後で、キャッシュを消去するように構成されてもよい。

【００５５】
本明細書で説明されたように、説明されたプロセスまたは方法のうちのいずれかは、実行されたときに、１つまたは複数のプロセッサに、上で説明された動作を実装させるように動作可能な、有形のデータ記憶媒体上もしくは有形のデータ記憶媒体内で具体化される、またはエンコードされる、一連のコンピュータ可読命令として実装されてよい。より小さなデータセットの場合、上で説明された動作は、単一のコンピューティング・プラット

フォームまたはノード上で実行され得る。例として、特定の実施形態において、図２および図３を参照して上で説明された段階的な生成プロセスは、サーバ２２で実行される単一のサーバ・プロセスによって実装されてよい。すなわち、上で説明されたウェブ・ページ生成プロセスおよびサービング・プロセスは、サーバ２２上で実装されてよい。より大規模なシステムおよびその結果としてのデータセットの場合は、パラレル・コンピューティング・プラットフォームが使用されてもよい。

【００５６】

図１は、１つのマスタ・サーバ２２ａおよび２つのスレーブ・サーバ２２ｂからなる、例示的な分散コンピューティング・システムを示す。いくつかの実施形態において、分散コンピューティング・システムは、商品サーバの高可用性クラスタを備え、そこでは、スレーブ・サーバが通常ノードと呼ばれる。２つのノードのみが図１に示されているが、いくつかの実施形態において、ノードの数は、百、または千、あるいはそれ以上さえも優に超えることがある。普通、高可用性クラスタにおけるノードは、冗長であり、その結果、１つのノードが特定のアプリケーションを実施中に異常終了した場合、クラスタ・ソフトウェアが、１つまたは複数の他のノード上でアプリケーションを再開することができる。

【００５７】

多数のノードはまた、大規模データベースのパラレル処理を促進する。いくつかの実施形態において、２２ａなどのマスタ・サーバは、クライアントからジョブを受信し、次いでそのジョブから生じるタスクを、サーバ２２ｂなどのスレーブ・サーバ、すなわちノードに割り当て、スレーブ・サーバ、すなわちノードが、マスタからの命令時に、割り当てられたタスクを実行する実際の作業を行い、タスク間でデータを移動させる。いくつかの実施形態において、上で論じられたように、クライアント・ジョブは、ＨａｄｏｏｐのＭａｐＲｅｄｕｃｅ機能を呼び起こすことになる。

【００５８】

同様に、いくつかの実施形態において、サーバ２２ａなどのマスタ・サーバは、大規模データベースのパラレル処理をサポートする分散ファイル・システムを左右する。特に、マスタ・サーバ２２ａは、ファイル・システムの名前空間、およびノードへのブロック・マッピング、ならびに、サーバ２２ｂなどのスレーブ・サーバ、すなわちノード上に実際に記憶されたファイルへのクライアント・アクセスを管理する。続いて、いくつかの実施形態において、スレーブ・サーバは、クライアントからの読み出し要求および書き込み要求を実行する実際の作業を行い、マスタ・サーバからの命令時に、ブロック作成、削除、および複製を実施する。

【００５９】

上述したプロセスおよび機構は、広く多様な物理システムによって、かつ広く多様なネットワーク環境およびコンピューティング環境において実装され得るが、以下で説明されるサーバまたはコンピューティング・システムは、限定する目的ではなく、教説の目的のために、例示的なコンピューティング・システム・アーキテクチャを提供する。

【００６０】

図６は、サーバ２２ａ、２２ｂ、またはクライアント・デバイス３０を実装するために使用され得る、例示的なコンピューティング・システム・アーキテクチャを示す。一実施形態において、ハードウェア・システム６００は、プロセッサ６０２と、キャッシュ・メモリ６０４と、有形なコンピュータ可読媒体上に記憶され、本明細書で説明された機能を対象とする、１つまたは複数の実行可能なモジュールおよびドライバを備える。加えて、ハードウェア・システム６００は、高性能入力／出力（Ｉ／Ｏ）バス６０６と、標準Ｉ／Ｏバス６０８とを含む。ホスト・ブリッジ６１０が、プロセッサ６０２を高性能Ｉ／Ｏバス６０６に結合し、一方で、Ｉ／Ｏバス・ブリッジ６１２が、２つのバス６０６および６０８を互いに結合する。システム・メモリ６１４、および１つまたは複数のネットワーク／通信インターフェース６１６は、バス６０６に結合する。ハードウェア・システム６００は、ビデオ・メモリ（図示せず）、およびビデオ・メモリに結合された表示デバイスをさらに含む。マス・ストレージ６１８およびＩ／Ｏポート６２０が、バス６０８に結合す

る。ハードウェア・システム６００は、オプションで、キーボードおよびポインティング・デバイス、ならびにバス６０８に結合された表示デバイス（図示せず）を含む。一括して、これらの要素は、広範なカテゴリのコンピュータ・ハードウェア・システムを表現するように意図されており、システムは、限定はされないが、米国カリフォルニア州サンタ・クララのインテル・コーポレーション（Ｉｎｔｅｌ　Ｃｏｒｐｏｒａｔｉｏｎ）によって製造されたｘ８６互換プロセッサ、および米国カリフォルニア州サニーベールのアドバンスト・マイクロ・デバイス（ＡＭＤ：Ａｄｖａｎｃｅｄ　Ｍｉｃｒｏ　Ｄｅｖｉｃｅｓ）インコーポレイティッドによって製造されたｘ８６互換プロセッサ、ならびに任意の他の好適なプロセッサに基づいた、汎用コンピュータ・システムを含む。

【００６１】

ハードウェア・システム６００の要素が、以下でより詳細に説明される。特に、ネットワーク・インターフェース６１６は、ハードウェア・システム６００と、イーサネット（登録商標）（たとえば、ＩＥＥＥ８０２．３）ネットワーク、バックプレーン、その他などの幅広いネットワークのうちのいずれかとの間に、通信を提供する。マス・ストレージ６１８は、サーバ２２ａ、２２ｂで実装される上で説明された機能を実施するように、データおよびプログラミング命令のための持続的なストレージを提供し、一方で、システム・メモリ６１４（たとえば、ＤＲＡＭ）は、プロセッサ６０２によって実行されるときに、データおよびプログラミング命令のための一時的なストレージを提供する。Ｉ／Ｏポート６２０は、ハードウェア・システム６００に結合され得る追加的な周辺デバイス同士の間に通信を提供する、１つまたは複数のシリアルおよび／またはパラレル通信ポートである。

【００６２】

ハードウェア・システム６００は、多様なシステム・アーキテクチャを含むことができ、ハードウェア・システム６００のさまざまなコンポーネントは、再配列されてもよい。たとえば、クライアント３０のキャッシュ６０４は、プロセッサ６０２とオン・チップで実装されてもよい。代替として、キャッシュ６０４は、クライアント３０のシステム・メモリ６１４で実装されてもよい。このようにして、プロセッサ・チップ上で、またはシステム・メモリ６１４で実装されるキャッシュ６０４は、一般に、一時的なストレージで実装されてよい。さらなる代替として、キャッシュ６０４は、永続的なストレージ・デバイスであってよい、クライアント３０のマス・ストレージ６１８で実装されてもよい。

【００６３】

さらに、開示された主題の一定の実施形態は、上のコンポーネントのすべてを要さなくてもよいし、含まなくてもよい。たとえば、標準Ｉ／Ｏバス５０８に結合されて示される周辺デバイスは、高性能Ｉ／Ｏバス６０６に結合してもよい。その上、いくつかの実施形態において、ハードウェア・システム６００のコンポーネントが単一のバスに結合されている、単一のバスのみが存在してもよい。さらに、ハードウェア・システム６００は、追加的なプロセッサ、ストレージ・デバイス、またはメモリなどの、追加的なコンポーネントを含む。

【００６４】

一実装形態において、本明細書で説明された実施形態の動作は、分散コンピューティング環境において、個々に、または一括して、ハードウェア・システム６００によって稼働される一連の実行可能なモジュールとして実装される。特定の一実施形態において、一組のソフトウェア・モジュールおよび／またはドライバが、ネットワーク通信プロトコル・スタック、パラレル・コンピューティング機能、ブラウジングおよび他のコンピューティング機能、最適化プロセス、その他を実装する。上述した機能モジュールは、ハードウェア、コンピュータ可読媒体上に記憶された実行可能なモジュール、またはその両方の組合せによって実現されてよい。たとえば、機能モジュールは、プロセッサ６０２などの、ハードウェア・システムのプロセッサによって実行されることになる、複数の、または一連の命令を備えることができる。最初に、一連の命令は、マス・ストレージ６１８などのストレージ・デバイス上に記憶されてよい。しかしながら、一連の命令は、ディスケット、

ＣＤ－ＲＯＭ、ＲＯＭ、ＥＥＰＲＯＭ、その他などの任意の好適な記憶媒体上に明白に記憶されてもよい。さらに、一連の命令は、ローカルに記憶される必要はなく、ネットワーク／通信インターフェース６１６を通じて、ネットワーク上のサーバなどのリモート・ストレージ・デバイスから受信されることも可能である。命令は、マス・ストレージ６１８などのストレージ・デバイスから、メモリ６１４の中にコピーされ、次いでプロセッサ６０２によってアクセスされ、実行される。

【００６５】

オペレーティング・システムは、ソフトウェア・アプリケーション（図示せず）へのデータの入力、およびソフトウェア・アプリケーションからのデータの出力を含む、ハードウェア・システム６００の動作を管理し、制御する。オペレーティング・システムは、システム上で実行されているソフトウェア・アプリケーションと、システムのハードウェア・コンポーネントとの間に、インターフェースを提供する。リナックス（登録商標）（ＬＩＮＵＸ（登録商標））オペレーティング・システム、米国カリフォルニア州クパチーノのアップル・コンピュータ・インコーポレイティッド（Ａｐｐｌｅ　Ｃｏｍｐｕｔｅｒ　Ｉｎｃ．）から利用可能なアップル・マッキントッシュ・オペレーティング・システム、ＵＮＩＸ（登録商標）オペレーティング・システム、Ｍｉｃｒｏｓｏｆｔ（登録商標）Ｗｉｎｄｏｗｓ（登録商標）オペレーティング・システム、ＢＳＤオペレーティング・システム、その他などの、任意の好適なオペレーティング・システムが使用されてよい。当然ながら、他の実装形態も可能である。たとえば、本明細書で説明された機能が、ファームウェアで、または特定用途向け集積回路上で実装されてもよい。

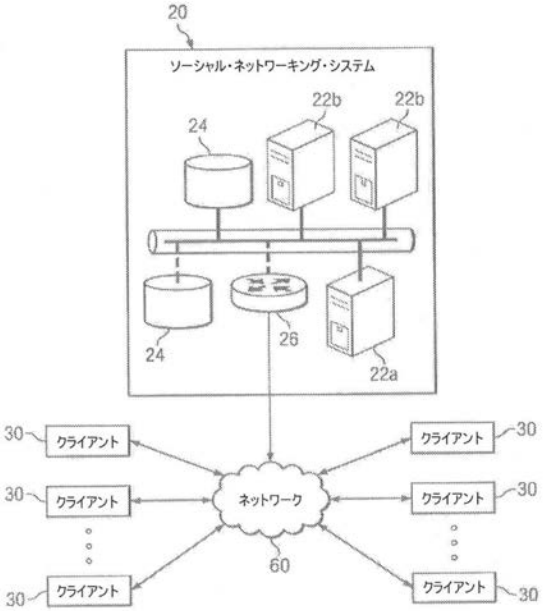
【００６６】

さらに、上で説明された要素および動作は、記憶媒体上に記憶された命令からなってもよい。命令は、処理システムによって取り出され、実行されてよい。命令のいくつかの例は、ソフトウェア、プログラム・コード、およびファームウェアである。記憶媒体のいくつかの例は、メモリ・デバイス、テープ、ディスク、集積回路、およびサーバである。命令は、処理システムによって実行されるときに、処理システムに、開示された主題と一致して動作するよう指示するように動作できる。用語「処理システム」は、単一の処理デバイス、または相互動作できる処理デバイスのグループを指す。処理デバイスのいくつかの例が、集積回路および論理回路である。当業者は、命令、コンピュータ、および記憶媒体に精通している。

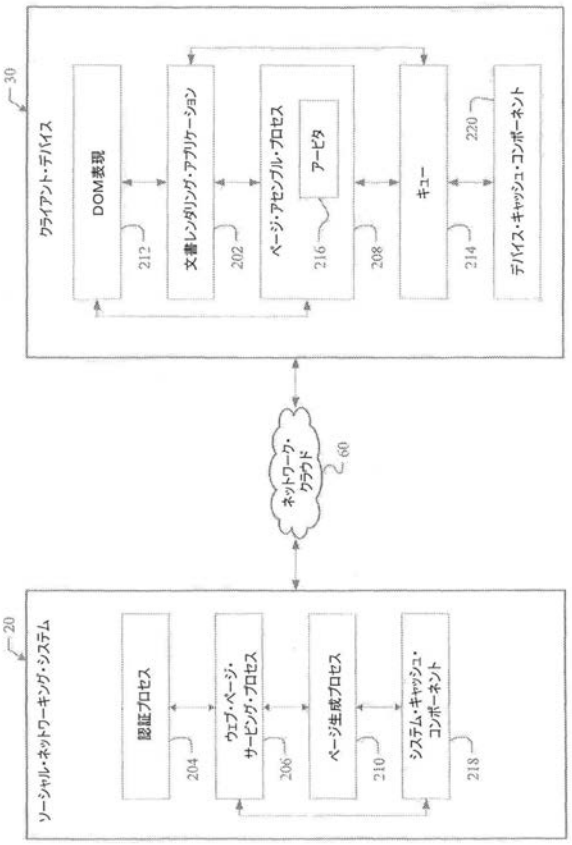
【００６７】

本開示は、当業者が理解するであろう、本明細書の例示的な実施形態に対するすべての変更形態、代用形態、変形形態、改変形態、および修正形態を包含する。同様に、適切である場合、添付の特許請求の範囲は、当業者が理解するであろう、本明細書の例示的な実施形態に対するすべての変更形態、代用形態、変形形態、改変形態、および修正形態を包含する。例として、本開示の実施形態は、ソーシャル・ネットワーキング・ウェブサイトと関連して動作するように説明されてきたが、開示された本主題のさまざまな実施形態は、ウェブ・アプリケーションをサポートする任意の通信施設と関連して使用されてもよい。さらに、いくつかの実施形態において、用語「ウェブ・サービス」と「ウェブ・サイト」とは、区別なく使用されることがあり、加えて、サーバに直接ＡＰＩ呼び出しを行うモバイル・デバイス（たとえば、セルラー・フォン、スマート・フォン、パーソナルＧＰＳ、携帯情報端末、パーソナル・ゲーム・デバイス、その他）などのデバイス上の、カスタムＡＰＩまたは一般化されたＡＰＩに及ぶ。

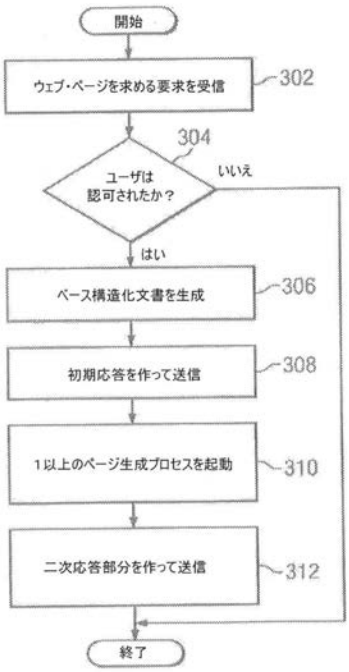
【図１】


20
ソーシャル・ネットワーキング・システム
22b
22b
24
24
26
22a
30
クライアント
ネットワーク
60

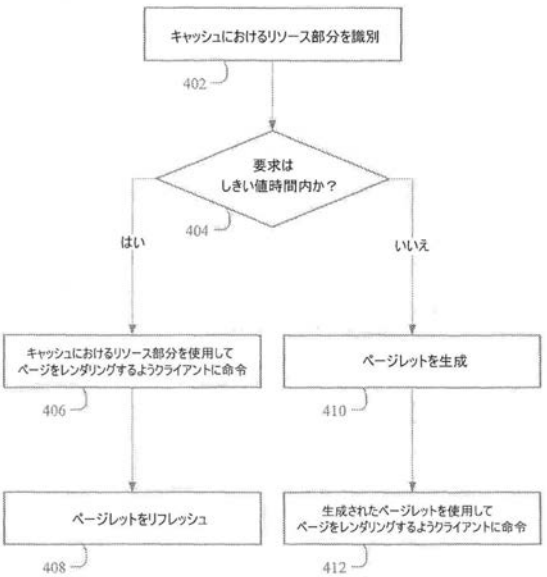
【図２】


30
クライアント・デバイス
DOM表現
212
文書レンダリング・アプリケーション
202
ページ・アセンブル・プロセス
アービタ
216
208
キュー
214
220
デバイス・キャッシュ・コンポーネント
ネットワーク・クラウド
60
20
ソーシャル・ネットワーキング・システム
認証プロセス
204
ウェブ・ページ・サービング・プロセス
206
ページ生成プロセス
210
システム・キャッシュ・コンポーネント
218

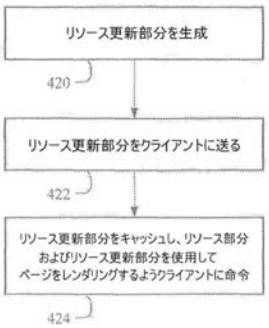
【図３】


開始
ウェブ・ページを求める要求を受信
302
304
ユーザは認可されたか？
いいえ
はい
ベース構造化文書を生成
306
初期応答を作って送信
308
1以上のページ生成プロセスを起動
310
二次応答部分を作って送信
312
終了

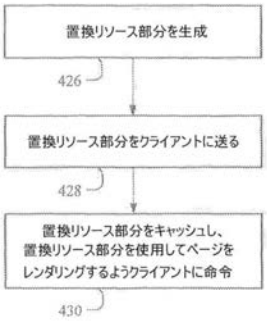
【図４Ａ】


キャッシュにおけるリソース部分を識別
402
要求はしきい値時間内か？
404
はい
いいえ
キャッシュにおけるリソース部分を使用してページをレンダリングするようクライアントに命令
406
ページレットを生成
410
ページレットをリフレッシュ
408
生成されたページレットを使用してページをレンダリングするようクライアントに命令
412

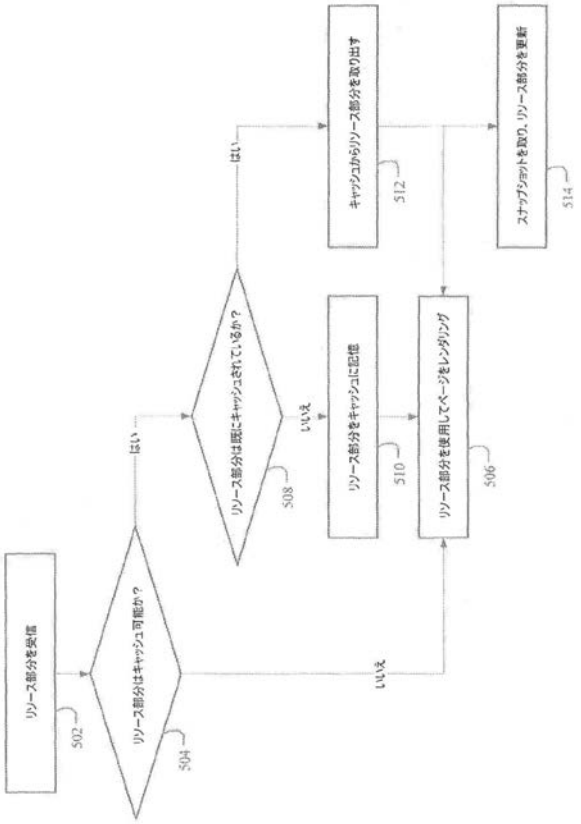
【図４Ｂ】


リソース更新部分を生成
420
リソース更新部分をクライアントに送る
422
リソース更新部分をキャッシュし、リソース部分
およびリソース更新部分を使用して
ページをレンダリングするようクライアントに命令
424

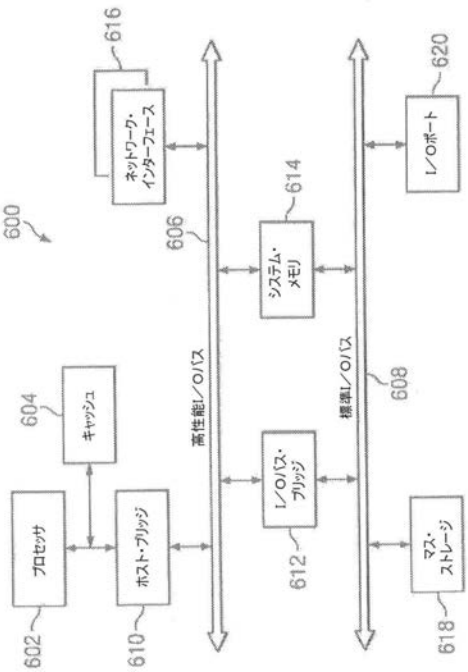
【図４Ｃ】


置換リソース部分を生成
426
置換リソース部分をクライアントに送る
428
置換リソース部分をキャッシュし、
置換リソース部分を使用してページを
レンダリングするようクライアントに命令
430

【図５】


リソース部分を受信
502
リソース部分はキャッシュ可能か？
504
はい
いいえ
リソース部分は既にキャッシュされているか？
508
はい
いいえ
キャッシュからリソース部分を取り出す
512
リソース部分をキャッシュに記憶
510
スナップショットを取り、リソース部分を更新
514
リソース部分を使用してページをレンダリング
506

【図６】


600
プロセッサ
602
キャッシュ
604
ホスト・ブリッジ
610
高性能I／Oバス
606
システム・メモリ
614
I／Oバス・ブリッジ
612
標準I／Oバス
608
ネットワーク・インターフェース
616
I／Oポート
620
マス・ストレージ
618

フロントページの続き

(72)発明者　ウェイ、シャオリアン
アメリカ合衆国　９４０２５　カリフォルニア州　メンロー　パーク　ウィロー　ロード　１６０１

Ｆターム(参考) 5B084 AA01 AA11 AB04 BB12 CB03 CB04 CC04 CD10 CD25