(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4318854号
(P4318854)

(45)発行日　平成21年8月26日(2009.8.26)　(24)登録日　平成21年6月5日(2009.6.5)

(51)Int.Cl.		F I		
G06F 17/30	(2006.01)	G06F	17/30	110F
G06F 12/00	(2006.01)	G06F	12/00	546A
G06F 13/00	(2006.01)	G06F	13/00	353C
G06F 15/00	(2006.01)	G06F	15/00	310E

請求項の数 15　(全 22 頁)

(21)出願番号	特願2000-527067 (P2000-527067)
(86)(22)出願日	平成10年12月23日 (1998.12.23)
(65)公表番号	特表2002-500404 (P2002-500404A)
(43)公表日	平成14年1月8日 (2002.1.8)
(86)国際出願番号	PCT/GB1998/003900
(87)国際公開番号	WO1999/034571
(87)国際公開日	平成11年7月8日 (1999.7.8)
審査請求日	平成17年11月1日 (2005.11.1)
(31)優先権主張番号	08/996,884
(32)優先日	平成9年12月23日 (1997.12.23)
(33)優先権主張国	米国(US)

(73)特許権者　390028587
ブリティッシュ・テレコミュニケーションズ・パブリック・リミテッド・カンパニー
ＢＲＩＴＩＳＨ　ＴＥＬＥＣＯＭＭＵＮＩＣＡＴＩＯＮＳ　ＰＵＢＬＩＣ　ＬＩＭＩＴＥＤ　ＣＯＭＰＡＮＹ
イギリス国，イーシー１エー・７エー　ジェイ，ロンドン，ニューゲート・ストリート　８１
(74)代理人　100058479
弁理士　鈴江　武彦
(74)代理人　100084618
弁理士　村松　貞男
(74)代理人　100092196
弁理士　橋本　良郎

最終頁に続く

(54)【発明の名称】ＨＴＴＰセッション制御

(57)【特許請求の範囲】

【請求項１】

ネットワーク上でクライアントにアクセスするために接続されたネットワークサーバであって、該クライアントは１つのセッション要求および１つのセッション応答としてセッションを定めたセッションプロトコルにしたがってネットワークサーバへアクセスし、該ネットワークサーバが、

ページ情報を記憶するメモリと、

要求を出しているクライアントからページ要求を受取り、要求に応答してある記憶されたページ情報を使用してページを生成し、要求を出しているクライアントに生成されたページをネットワーク上で送信するページコンパイラとを含み、

ここにページコンパイラは生成されたページにページスクリプト機能を取付け、生成されたページとともにページスクリプト機能を送信し、スクリプト機能は、複数の一定の間隔および異なる長さを有する１以上の間隔で、要求を出しているクライアントからサーバへ送られる多数の追加のメッセージを生成するために実行可能な命令を含むネットワークサーバ。

【請求項２】

要求を出しているクライアントに生成されたページおよびページスクリプト機能を送信した後に、アクセスを要求する他のクライアントのために、サーバが記憶されたページ情報の少なくとも幾つかへのアクセスを阻止し、アクセスの阻止は、要求を出しているクライアントがもはやページスクリプト機能を実行していないことを示す追加のメッセージの

不在まで継続する、請求項1記載のネットワークサーバ。
【請求項３】
要求を出しているクライアントが生成されたページを使用しているときのみ、スクリプト機能が追加のメッセージを生成する請求項1または２記載のネットワークサーバ。
【請求項４】
要求を出しているクライアントが継続的に生成されたページを使用しているときのみ、スクリプト機能が追加のメッセージを生成する請求項３記載のネットワークサーバ。
【請求項５】
ページコンパイラが、サーバから要求を出しているクライアントへ送られる応答メッセージを生成するスクリプトプロセスをさらに含み、追加のメッセージはサーバがスクリプトプロセスを実行するための要求を含み、それにより、サーバが追加のメッセージを受取るたびごとに応答メッセージを生成して送る請求項１ないし４の何れか１項記載のネットワークサーバ。
【請求項６】
応答メッセージが空のメッセージである請求項４記載のネットワークサーバ。
【請求項７】
全ての受取ったページ要求を記録するアクセスログをさらに含む請求項１ないし６の何れか１項記載のネットワークサーバ。
【請求項８】
アクセスログが、全ての前記ページ要求および追加のメッセージが受取られたときの記録を維持し、ネットワークサーバが、アクセスログを解析して、記録に基いて要求を出しているクライアントが要求されたページを使用し続けるか否かを判断するプロセッサをさらに含む請求項７記載のネットワークサーバ。
【請求項９】
メモリがデータベースを含み、ページ情報が要求を出しているクライアントによってデータベース内で検索および変更できるデータポイントを含む、請求項１ないし８の何れか１項記載のネットワークサーバ。
【請求項１０】
データベースが前記データポイントの日付および時刻フィールドを含み、前記日付および時刻フィールドの各々が、対応するデータポイントに最後にアクセスした要求を出しているクライアントはだれかと、前記最後のアクセスが行われたのはいつかとを識別する情報を含む請求項９記載のネットワークサーバ。
【請求項１１】
実行可能な命令が、連続した追加のメッセージ間の一定の予め定められた間隔で、要求を出しているクライアントからサーバに送られる多数の追加のメッセージを生成するように配置される、請求項１ないし１０の何れか１項記載のネットワークサーバ。
【請求項１２】
追加のメッセージが生成される間隔が不規則である請求項１ないし１０の何れか１項記載のネットワークサーバ。
【請求項１３】
各セッションを１つのセッション要求と1つのセッション応答として定めたセッションプロトコルを使用する、クライアントおよびサーバのネットワーク上で送るためのページを、生成する方法であって、
要求を出しているクライアントからページ要求を受取ることと、
ページ情報を検索して、要求されたページを組立てることと、
ページ情報を要求されたページへコンパイルすることと、
ネットワーク上で要求を出しているクライアントにページスクリプト機能とともにページを送信することとを含み、該ページスクリプト機能は、複数の一定の間隔および異なる長さを有する１以上の間隔で、要求を出しているクライアントから送られる多数の追加のメッセージを生成するために実行可能な命令を含む方法。

【請求項１４】
検索するステップが、ページ情報の少なくとも幾つかのためのデータベースにアクセスするステップを含む請求項１３記載の方法。
【請求項１５】
ページスクリプト機能が、前記複数の一定の間隔および異なる長さを有する１以上の間隔で、追加のメッセージをこれ以上は送らなくなるまで、要求されているページの他の要求を出しているクライアントをロックアウトするステップをさらに含む請求項１４記載の方法。
【発明の詳細な説明】
【０００１】
発明の属する技術分野
本発明は、インターネット上で行われるようなネットワーク通信、とくにインターネット上のＨＴＴＰ通信に対するセッション制御に関する。
【０００２】
従来の技術
インターネットは、遠隔のクライアントが通信するサーバの集合を含む分散形ネットワークである。ワールドワイドウエブはインターネット上で使用可能なサーバの集合に対する名前である。したがってウエブはクライアントと、インターネット上で使用できるいくつかのサーバとの間の通信路を形成する。
【０００３】
ワールドワイドウエブ上のクライアントは、通常ウエブブラウザである。これらは、インターネット上でサーバに対して要求を出し、サーバによって戻されたページを処理および表示するプログラムである。代わってワールドワイドウエブサーバはブラウザからの要求を処理して、要求を出しているブラウザへドキュメントを供給するプログラムである。これらのドキュメントは通常はハイパーテキストマークアップ言語（ＨＴＭＬ）というブラウザおよびサーバによってワールドワイドウエブ上で一般的に認識されている言語である。サーバによって戻されるウエブページを読取るために、ブラウザはスクリプティング言語で戻されたページを表示することができる。
【０００４】
普及しているスクリプティング言語は、いわゆるジャバスクリプトおよびビジュアルベーシックスクリプトである。ハイパーテキストマークアップ言語（ＨＴＭＬ）は、戻されたページをどのように表示するかをブラウザへ知らせるのに使用されるコマンドのフォーマットである。ＨＴＭＬはさらに、代わりのグラフィックまたはスクリプティング言語のようなシーン情報の背後を特定することができる。サーバとブラウザとの間でウエブページを通信するときに、採用される通信プロトコルはいわゆるハイパーテキスト転送プロトコル（ＨＴＴＰ）である。これはワールドワイドウエブによって使用される通信プロトコルであり、ウエブサーバとウエブブラウザとの間の通信を調和させる。
【０００５】
普通、ウエブブラウザはウエブサーバから“スタティックな(static)”のワールドワイドウエブページを要求して受取る。スタティックなページは、ページの日付が予め書き込まれて固定されることを特徴としている。しかしながら時にはウエブサーバは、ダイナミックな(dynamic)内容をブラウザへ伝えるプログラムを実行する。例えばサーバが、ユーザがブラウザを介してアクセスおよび変更することができる情報のデータベースと通信するとき、サーバは、現在のままの状態(as-current condition)におけるデータベース情報を含むウエブページを伝えることができる。したがって、ユーザがブラウザを介してウエブサーバと接触して、データベースエントリにアクセスして変更するときは、その後にこのサーバにアクセスする別のブラウザは修正されたデータベース情報を検索することになる。ウエブサーバがプログラムを実行して、ダイナミックな内容を伝えることを可能にする機構は、共通のゲートウエイインターフェイスと呼ばれている。
【０００６】

現在、ウエブ上の全てのサーバはアクセスログを含む。このログは、インターネット上でブラウザがウエブサーバに対して行われた全てのアクセスの詳細を含むファイルである。このログに含まれる詳細としては、アクセスの日付、アクセスの時間、要求者のコンピュータ識別子（アドレス）、要求される情報、ユーザへ送られた情報量、およびおそらくは転送状態がある。さらに、ウエブ上の各ブラウザ（クライアント）はキャッシュメモリを含んでおり、該キャッシュメモリは一定数の検索されたウエブページを保持しているので、これらのページは、ユーザによって迅速に検索されるインターネット上でサーバから再びロードされる必要はない。

【０００７】

２つのコンピュータ（例えば、サーバとブラウザ）が一緒に接続されている時間は“セッション”と呼ばれる。一般的な用語では、セッションは通常、クライアントがサーバと通信を始めた時間から、クライアントが通信を終了する時間まで続く。しかしながらウエブ上では、セッションは僅かに異なる定め方がされている。インターネット以外のクライアント／サーバのデータ転送では、クライアントとサーバはそれら自身の間の接続を、クライアントがサーバから情報を要求する間、およびサーバが情報をクライアントへ提供する間設定する。このような場合に、クライアントとサーバとの間の“セッション”は、クライアントとサーバとの間で接続が実行されるときから、クライアントまたはサーバかセッションを終了するときまで行われる。セッション中にクライアントとサーバとの間ではデータ転送およびデータ要求を何度か行うことができる。しかしながらウエブ上では、ハイパーテキスト転送プロトコルおよびワールドワイドウエブ構造における制限によって、ウエブ上のクライアントおよびサーバは“セッション”の基準外の概念を厳守する。ウエブ上でセッションは、ページ（データファイル、イメージファイル、など）が要求され、ユーザ（ブラウザ）へダウンロードされる時間のみから構成される。したがってウエブクライアントとウエブサーバとの間の各セッションは、1ページをクライアントからサーバへ要求し、1ページをサーバからクライアントへ転送する時間になる。したがってサーバのログに記録されるセッション情報では、ページ要求の日付、ページ要求の時刻、ページ要求を行うコンピュータ、要求されたページを識別するファイル情報、ページを送るために要求を出しているコンピュータへ送られるバイト数、およびおそらくは転送状態を識別する。

【０００８】

したがってインターネットは、１つのクライアントコンピュータを別のサーバコンピュータへ接続して、例えばデータベースへアクセスするといった典型的（クラシック）な場合とは異なる。典型的な場合、クライアントは、サーバへ接続されたままで、データベース記録を検査する。このような場合サーバはクライアントによるデータベースアクセスの継続期間を記録することができる。セッションはデータベースとの間で接続を開始および終了する継続時間によって定められるので、これらの記録は、ユーザが記録を調べるのに費やした時間に関する統計的な解析に使用することができる。さらにサーバは、データベース記録を“ロック”して、このクライアントが調べ終わるまで他のクライアントがこのデータベース記録にアクセスするのを妨げることもできる。

【０００９】

セッションはクライアントとサーバとの間の合計接続時間によって定められず、単一のページ要求および単一のページ伝送によって定められるので、インターネット接続には同様のことが当てはまらない。インターネットの場合、サーバは、ユーザが特定の記録を使用して調べ終えたときを知る方法も、ユーザが特定の記録を調べた時間を知る方法もない。従来の方法では、手操作の機構を使用して、ユーザが記録を完了したことをサーバに知らせることによって、これに対処する試みを行うが、このようなシステムはクライアント依存型であり、サーバ単独のときは実行できない。したがってサーバが、ユーザが記録を完了した正確な時を知るか、またはユーザが記録をどのくらい長く調べたかを正確に知ることが重要なとき、このウエブ構造は効果的な解決案を与えなかった。

【００１０】

インターネット環境においてどのようにしてこの問題が表面に現れるかの例は2人のユーザがウエブを介してデータベース内の情報にアクセスすることを試みるときに発生する。例えば、ユーザＡは倉庫にある特定の製品Ｘに対する在庫の製品から情報を要求すると仮定する。倉庫ではウエブ上のサーバを使用して、種々の製品の在庫量に関してユーザに情報を伝える。さらにユーザＡが在庫品情報を要求するとき、現在倉庫で２００ユニットの製品Ｘが得られると仮定する。製品Ｘの在庫品を識別するページに対するユーザＡからの要求に対する倉庫のサーバは、そのデータベースを検査して、在庫品を“２００”個取り出して、この量を識別するページをユーザへ送る。次に別のユーザであるユーザＢが製品Ｘに対する同じ在庫品情報を要求すると仮定する。倉庫サーバはユーザＢからページ要求を受取り、そのデータベースを調べて、製品Ｘの量が２００個であることを知り、製品Ｘの在庫品が“２００”の入手可能なユニットであることを識別するページを送る。したがってユーザＡが別のページ要求において、２００の入手可能なユニットから１５０ユニットを注文することを倉庫サーバへ伝えるときに問題が発生する。倉庫サーバはユーザＡからの注文に応答して、１５０ユニットの低減をそのデータベースに書き込む。次に、ユーザＢは倉庫サーバへ注文要求を送ることによって、倉庫サーバから１００ユニットを要求する試行を行う。倉庫サーバが使用可能な在庫が２００ユニットしかないときに、２５０ユニットの注文を受領することによって問題が発生する。

【００１１】

上述の例では、倉庫サーバがユーザＢのページ要求に応答して、ユーザＡが同じページを完了したことを知るまで、ユーザＢへページを送らないことが重要である。ユーザＡがいくつかの機構を使用して、製品Ｘの在庫品ページの使用を完了したことを倉庫サーバへ手操作で知らせるとき、倉庫サーバはユーザＢへの同じ製品在庫品ページの伝送が時宜を得ていて、正確であるか否かを知ることはできない。

【００１２】

ウエブにおける制限されたセッションの定義が問題を生じる別の例は、セッション制御が統計的な目的に対して要求されるときである。これは、例えばウエブに広告を出している会社が誰かがそのページを調べた時間または広告を掲載しているページを知りたいときに発生する。このウエブセッションの定義、すなわち１セッションは1ページを要求および伝送するのみであることを使用して、ユーザが手操作の機構を使わずにサーバのページまたは広告を調べた時間を知ることはできない。したがってユーザがページを調べた時間をサーバが知るには必ず、クライアントが特定のページの使用を終了したことを識別しなければならない。

【００１３】

図１はインターネットセッションの現状を示している。図１において、システム１は第１のクライアントから第ｎのクライアントの中から何人かのクライアントを含む。クライアントは、インターネット接続10を通って、サーバ12を含む多数のサーバにアクセスする。図1に示したように、サーバ12は“Ａ”および“Ｂ”によって示されたセッション中にクライアントから要求を受取る。

【００１４】

セッションＡ中に、第１のクライアントはインターネット10を通ってサーバ12へ、特定の識別されたページを得たいことを知らせる（“give me a page”と表示される）。サーバ12は、ページをアセンブルして、インターネット10を介して第１のクライアントへそれを送ることによって（“here’s your page”と表示される）、この要求に応答する。この対の通信（“give me a page”および“here is your page”）は、全セッションＡの間に第１のクライアントとサーバ12との間の全通信である。同様に、第ｎのクライアントも、セッション“Ｂ”として示したようにサーバ12からページを要求することができる。セッションＢでは第ｎのクライアントはさらにサーバ12からページを要求し、サーバ12からそのページを受取る。

【００１５】

上述のように、第１のクライアントおよびクライアント番号ｎがサーバ12から同じページ

をセッションＡおよびセッションＢ中にほぼ同時に要求するときに問題が発生し、したがってサーバ12において記録をオーバーラップして変更してもよい。したがってこのような場合に、サーバ12は、クライアントから要求をオーバーラップすることに応答してデータベース16を誤って変更することがある。さらにサーバ12がアクセスログ14にセッション時刻を記録していても、セッションにおいてページ要求およびページ伝送はそれぞれ独立して開始および終了するので、サーバ12がセッションＡおよびセッションＢが継続する時間を識別できないという問題も発生する。

【００１６】

発明が解決しようとする課題

現在の定義は、クライアントがサーバによって送られるページにアクセスする時刻および時間を識別できるようにするインターネット上でのセッション制御を与えている。本発明はこれをするのに、クライアントに向けて送られる各ページに“ハートビートプログラム(heartbeat program)”を識別する情報を添付して、このハートビートプログラムは、所定の間隔でサーバへビートを送るようにし、その間に送られたページが調べられる。最も簡単な例では、サーバはページをクライアントへ戻し、クライアントは自分がページを調べ終えるまで毎分サーバへビート（活性状態表示信号）を送り戻す。

【００１７】

ページによって戻されたこれらのビートを記録し、検索中にページによって戻されたビート数を調べることによって、サーバはクライアントが特定のページを見ていたおおよその時間を識別することができる。例えば上述の例で、５ビートが戻されるとき、サーバはクライアントがページを５分を越え６分未満の間調べていたと結論することができる。さらに、クライアントは６分間の最後にページが解放されたことを知り、もとのクライアントとオーバーラップせずに新しいクライアントによってアクセスすることができる。

【００１８】

本発明は、インターネット上での純粋なセッション制御を与えていないが、ページがアクセスされた時刻および経過した時間を識別する良好な情報をサーバへ与えて、オーバーラップを避け、統計的な情報を生成する。さらに本発明は、ビート（うなり）周波数と統計的サンプリングの許容差とのバランスをとることによって使用できる。したがって毎分行われるビートが、インターネット容量の負担を増すが、例えばビートが５分ごとに発生するときよりも正確な結果を与える。

【００１９】

発明の実施の形態

クライアント、例えば図１の第１のクライアントが、サーバ12へページ要求を送るときはいつでも、サーバ12はアクセスログ14において要求に対するログエントリを生成する。典型的なログエントリを次に示す：

elara.planet.bt.co.uk--[12/may/1994:10:10:11-400]’’GET/icons/
blank.xbm HTTP/1.0’’200 509

上述のログエントリにおいて、“elara.planet.bt.co.uk”はクライアントコンピュータのアドレスである。ブラケット内には要求の日付および時刻が示され、要求のタイプ（上述の例では“400”）によって終了している。ブラケット情報の後には、指示されたファイル（“icons/blank.xbm”）を検索するコマンドが記載される。その後にはプロトコル（HTTP/1.0）、さらにその後には戻りコード（200）、および要求されるファイルのバイト数（509）が続く。

【００２０】

上述の例では、サーバ12のアクセスログ14へ挿入されたエントリはクライアント“elara.planet.bt.co.uk”によって１９９４年５月１２日、１０時１０分１０秒にＨＴＴＰフォーマットでサーバ12から“icons/blank.xbm”と呼ばれるファイルを検索する要求である。クライアントがセッション（図１のＡまたはＢ）中にサーバ12を使ってページを要求する度に、サーバ12のアクセスログ14中に類似のエントリが行われる。上述のやり方ではアクセスログ14の展開は従来技術において知られており、クライアントがサーバ12からペー

ジを要求する種々のセッションの数および時間の正確な記録になる。しかしながらこれは、クライアントが特定のページにアクセスする継続期間を記録しない。

【００２１】

過去には、ウエブ報告を行ういくつかのプログラムが、サーバによって与えられるページ上で特定のクライアントのヒット履歴を調べることによってクライアントが特定のページ上にどのくらい長く留まっていたかについての統計的な推測を行っている。しかしながら従来技術のアクセスログ方法からこの統計的な推測が、特定のページにおいてクライアントが費やした継続期間の有効な指標であるか否かを知る方法はない。

【００２２】

本発明は、一連の潜在的なセッション終了要求を生成することによってウエブ上でのセッション情報が欠落していることを無視（バイパス）する。最も簡単な実施形態では本発明は、サーバ12上に置かれた簡単なＣＧＩスクリプトと一緒に、サーバ12によって報告される各ページ内の小さいスクリプト言語機能のみを必要とする。サーバ12によって与えられる各ページに置かれるスクリプト言語機能によって、ページは所定の間隔でサーバ12へ向けたビートメッセージを生成する。サーバ12上に置かれた簡単なＣＧＩスクリプトはこれらのビートを受取り、それらを記録し、後でより詳しく記載する方法でそれらに応答する。しろうと表現では、ページによって与えられたビートは本質的に、クライアントがページを利用し続け、サーバからの応答が“no change to the current pages（現在のページを変更しない）”という応答であることを知らせる。このやり方では、このセッション制御システムは、本質的にユーザに見えない。

【００２３】

図２は本発明の例示的な実施形態にしたがってクライアントおよびサーバが本発明のセッション制御情報を交換することを示している。なお図２ではクライアントからサーバへの通信は右矢印によって示され、サーバからクライアントへの通信は左矢印によって示されている。さらに図２では、通信は図２の上から下へ年代（時間）順に配置されている。なお図３、４、５、６、８、および９も同様に各図の上から下へ実行される年代順のシーケンスで位置付けられている。

【００２４】

図２では、クライアントとサーバとの間の最初の2つの通信は、図１に関して記載された従来技術の状況と同一である。したがってクライアントは、サーバへ“give me page X（ページＸを出力してください）”という要求においてセッション（例えば、図１のセッションＡ）を開始する。次にサーバは上述の例示的なログエントリにしたがってログ14へ要求をログする。次にサーバは応答“here is your page（出力しました）”において要求されたページでクライアントへ応答する。このページは、後のビートを生成するスクリプト言語機能を含む点で従来技術のページとは異なる。例示的なスクリプト言語機能の例を後で詳しく記載する。

【００２５】

クライアントがページを受取ると、スクリプト機能は所定の時間X_0の間待ち、次にサーバへの要求を生成して、本質的にサーバへ“Ｉ’ｍ here”、すなわちクライアントがページＸを受取ってから時間X_0を経た後、このクライアントが依然としてページＸを使用していることを伝える。このサーバへの最初の応答後、第１のビートとしてページ受取りが参照される。第１のビートに応答して、サーバはビートをログして、アクセスログに入り、本質的にクライアントへナルページを送ることによって応答し、これは“ＯＫ、私はページＸをまだ開いていると理解します(0kay-I understsnd you are still on page X.)”という意味であると解釈できる。

【００２６】

最初の時間間隔X_0に続いて第１のビートがサーバへ送られるときに、このページは別の時間間隔（X_1）の間待って、第２のビートをサーバへ送る。サーバはこのビートをログして、応答する。プロセスは間隔X_2およびX_3の間継続する。最後に、期間X_3の後にビートが送られた後で、サーバは、クライアントからのビートが停止したために、クライ

アントがページＸを置き去りにしたことに気付く。

【００２７】

何れのイベントでも、図２の例示的実施形態におけるサーバはアクセスログ14へ十分な情報をログして、クライアントがページＸを要求して、少なくとも（$X_0+X_1+X_2+X_3$）と同じ長さの間ページ上に留まることを示す。さらに、図２の実施形態においてビートの継続期間（X_1，X_2，およびX_3）は均一の長さをもつので（これは後出の実施形態に記載されているように本発明の必要条件ではない）、サーバログはクライアントがサーバ上に（X_0+3X_1）ないし（X_0+4X_1）の間居残っていたことを知るのに十分な情報を含んでいる。

【００２８】

図３および４は、異なる種類のデータを累積するのに有益な本発明のさらに別の実施形態を示している。図３および４は、図２に示したタイプのダイヤグラムから、簡潔にするためにサーバおよびサーバ応答を削除した簡略化したダイヤグラムである。図２、３、および４の実施形態では、ページによってサーバへ与えられたビートの機構および継続期間が異なる。図２において、ビート（X_1、X_2、X_3）は均一の間隔で構成されている。図３において、第１のビートは間隔X_4で行われ、その後第２のビートはＸ４の継続期間の約４倍の間では行われない。次に第３、および第４、などのビートが第１のビートと同じ継続期間で発生する。したがって、図３においては$4X_4=X_5=4X_6$、などである。図３の実施形態は統計の判断に関係付けることができ、このとき広告者は、ユーザが広告（Ｘ４）を吸収するのに十分長くページ上に留まっているか否かを知りたいが、ユーザが留まっている時間が（$T=X_{5\text{のように}}$）相当に長くない限り、ユーザが留まっている時間にはあまり関心がなく、$T=X_5$のときは、広告者は、ユーザがページ上に留まっている時間を正確に知ることに関心があり、これは周波数の増加したビート（$T=X_6$）によって示される。

【００２９】

本発明のさらに別の例示的な実施形態は図４に示されており、ビート周波数は時間で増加する。図４の例は、例えば２人のユーザが同じデータベースにアクセスして、サーバは第１のユーザをポーリングして、第１のユーザがデータベースとの接続を終了して、第２のユーザがこのデータベースにアクセスできるようになったときを判断したい場合に関連している。この場合に、サーバは第１のビートを要求する前にユーザに一定量の（より長い）時間（$T=X_7$）を与えることを期待する。その後サーバはより頻繁にビートを要求して、ユーザがシステムから離れるときを判断する。したがってこの図４の実施形態は、ユーザがページから出ていく前にそのページに相当な時間を費やすことを期待されることを知るときに適している。例えば上述の従来の技術の段落で記載した実施形態において、２人のユーザが倉庫サーバ上の同じ製品Ｘの在庫品のデータベースにアクセスする場合に、ユーザが在庫品記録にアクセスし、購入をするデータベース内で約３または４分を費やすことが分かっているとき、倉庫サーバはX_7ビートを約３分間、X_8ビートを約１．５分間、X_9ビートを４０秒間、などに設定して、ユーザが約３ないし４分間の範囲内で実際にデータベースを離れるときを精密に短くすることができる。

【００３０】

図５は、サーバ12の構造の例示的実施形態を示している。なおページ要求およびビート信号が図５の左側の上から下へ年代（時間）順に示されている。クライアントからページ要求が受取られるとき、サーバ12はプロセッサ20においてページ要求を受取り、次にデータベース16にアクセスして、ページを見付けるかまたはコンパイルする。次にプロセッサ20は“ページ戻し”伝送においてクライアントへページを送る。既に記載したように、このページは所定の間隔でビート信号を生成するスクリプト言語機能を含む。図５に示したように、第1のビート、すなわち“第１のビート”は第１の間隔の後に戻される。このビートはプロセッサ20によって受取られ、ビート信号をログ14に記録する。次にプロセッサ20はＣＧＩプロセス21にアクセスし（後で記載するように、第１のビートの信号で参照される）、プロセッサ20に知らせて、“ＯＫ”信号をサーバへ送る。

【００３１】
その後、サーバ12のユーザは遠隔のコンピュータ22を介してプロセッサ20にアクセスし、ログ14にアクセスして、ページ要求および次のビート情報を取除く。これらから、コンピュータ22は統計のために継続時間情報をコンパイルすることができる。
【００３２】
サーバユーザがコンピュータ22を介してログ14へアクセスするとき、本発明の例示的な実施形態にしたがって、サーバユーザは図６に示したアクセスログに類似したアクセスログ14を調べることになる。図６の例は、図２に示した通信シーケンスに類似しており、ビート周波数（Ｘ１，Ｘ２，Ｘ３，など）は均一の継続期間をもつ。後で詳しく記載するように、図６のアクセスログ14において識別された情報から、サーバのユーザは特定のページにアクセスした特定のクライアントが９分を越え１０分未満の間それを調べていたことを知らせる。
【００３３】
これは、図６のアクセスログ14に示した各行を調べることによって確認することができる。第1の行では、プロセッサ20はアクセスログ14に、“Index.html.”というタイトルを付けられたページに対して“pc59”として識別されるクライアントからのページ要求を記録した。このページ要求は１９９６年１０月９日、１６時５１分５６秒に行われた。図６のアクセスログ14の第１の行はコマンド“GET”で識別されるページ要求である。
【００３４】
図６において“GET”要求の行の後の５行は、それぞれページによってサーバへ戻される５つのビートである。これらの行から分かるように、ビートは同じＰＣ、すなわち“pc59”から同じ日（October 9, 1996）に戻される。ページ要求から５分後にハートビートが始まり（図２を参照すると、X_0は５分に相当する）、その後１分ごとにビートを生成する（図２を参照すると、$X_1 = X_2 = X_3$＝１分）。したがってスクリプトは、このページ（Index.html）において、第１のビートを送る前に５分間待つように伝えられる。その後スクリプトは毎分１つのビートをサーバへ送ることができる。図６に示した情報から、pc59のユーザが１６時５１分５６秒にindex.htmlページを受取り、少なくとも１７時００分５６秒まで、しかし１７時０１分５６秒を超えずにそこに留まると判断することができる。このことは、１７時００分５６秒に最初に完了した後に毎分ビートが送られるようにスケジュールされていることから分かる。
【００３５】
もちろん、図６のアクセスログの例は単に、使用できるビートシーケンスの１つの例である。図６において示したビートによって与えられるものよりも細かい細分性で、クライアントユーザが、ページ上に留まっているか否かを知りたいとき、スクリプトは、例えば1分毎にではなく、１０秒毎にビートを生成するように容易に変更することができる。さらに（図６の例におけるページ要求から５分後に行われる）最初のビートも同様に変更することができる（例えば、1分毎に変更する）。いずれにしても本発明は、ビート間隔の機構または時間に限定されない。
【００３６】
他のタイプのビートスクリプトも本発明の技術的範囲内で生成および構想することができる。例えばスクリプトはビートをクロックタイムに基いて（例えば、各正時（１時間ごと）に）、または特定のアルゴリズムに基いて（例えば、図４に示したように、$X_n = 2X_n + 1$）与えることができる。さらに別のタイプのより高度なスクリプトでは指定された時刻（例えば、午後２時）に単一のビートを生成して、午後２時にページ上に存在すると推定されるユーザが実際に指定された時刻にそこにいるか否かを判断することができる。多くの他のタイプのビートスクリプトを構想することができるので、本発明は特定のタイプのビートシーケンスに限定されない。
【００３７】
図７は、本発明にしたがって使用することができる１つの例示的なタイプのスクリプトを示している。もちろん、本発明は図７に記載した特定のスクリプトに限定されないが、本

明細書および請求の範囲に記載した目的および機能が構想（予見）される他のタイプのスクリプトを含む。図７の例において、例えば図５のサーバ12はより詳しく示されている。図５の実施形態と図7の実施形態の１つの相違点は、データメモリが異なることである。図５では、サーバ12はデータベース16にアクセスし、データベース16からデータを引き出して、ページをコンパイルする。他方で、図７ではサーバ12のプロセッサ20はファイルサーバおよびファイルディレクトリ17からページにアクセスする。サーバ12がクライアントへ送るページを見付けて準備するとき、図５、図７、およびその他の実施形態は本発明の技術的範囲内で構想される。

【００３８】

図７の例では、クライアント（図示されていない）はサーバ12から“index.html”というタイトルを付けられたページを要求する。インデックスindex.htmlページにおいて、クライアントユーザへディスプレイされるページ情報を与えるページデータに加えて、index.HTMLページの第１ないし第１０行目に示したスクリプト機能をが含まれる。図７に示したように、index.HTMLページ上の例示的なスクリプト機能を次に示す：

【数１】

```
1. <SCRIPT LANGUAGE="JavaScript">
2. function beat( ) {
3. setTimeout("beat( )", 60000);
4. document.beatform.submit( );
5. }
6. </SCRIPT>
7. <body onLoad='setTimeout("beat( )", 300000)'>
8. <form name="beatform" method=POST action="/cgi-bin/nph-beat">
9. <input type=hidden name=whoami value="index">
10. </form>
```

【００３９】

スクリプト機能は、既に記載したように、図２ないし６の例示的な実施形態にしたがって、クライアントからサーバ12へビート信号を与える。スクリプトにおいて、特定の行が次の機能を実行する：

第１行目は、スクリプトが周知のジャバスクリプトフォーマットにおいて書き込まれることを特定する。もちろん、他のスクリプト言語も知られていて、使用でき、代わりの例を次に示す。

【００４０】

第２行目は、“beat”というタイトルを付けられた機能を規定している。

【００４１】

第３行目は、規定のビート機能が６万ミリ秒（６０秒）ごとにコールアップされるの規定のビート機能を要求する。

【００４２】

第４行目は、“beatform”というタイトルを付けられたフォームをサーバ12へ送る命令である。

【００４３】

第５行目は、機能の定義を終了する。

【００４４】

第６行目は、ジャバスクリプトセクションを終了する。

【００４５】
第７行目は、index.HTMLページが検索されてから３０万ミリ秒（５分）後に“beat”というタイトルを付けられた機能を実行する。これは第１のビート（図２のK_0）を設定する。

【００４６】
第８行目は、サーバへ提出するフォームを規定する。フォームが提出されるとき、サーバ12上で“nph-beat”と呼ばれるプログラムを実行する。

【００４７】
第９行目は、フォームが少なくとも１つのフィールドをもつことを特定する。

【００４８】
第１０行目は、フォーム形成の最後を特定する。

【００４９】
index.htmlページがクライアントによって受取られると、５分後にビート機能を実行し、“beatform”を６０秒ごとにサーバへ送る。ビートフォームがサーバ12によって受取られると、サーバ12はこのビートフォーム（index.htmlスクリプトの第８行目）で動作を読み取り、“nph-beat”スクリプトを実行すべきことを知る。プロセッサ20はＣＧＩプロセス21（別個のソフトウエア構成要素であっても、プロセッサ20に組込んでもよい）を使用して、図７に示したサーバＣＧＩスクリプトを実行する。このサーバＣＧＩスクリプトはＵＮＩＸフォーマットで示されているが、後述のように代わりのフォーマットであってもよい。図７に示したように、ＣＧＩサーバスクリプトは非常に簡単である。ＣＧＩサーバスクリプトの1行目は、スクリプトがＵＮＩＸシェルプログラムであることを示している。第２行目はＨＴＴＰヘッダを、index.htmlページを要求したクライアントのブラウザへ送る。このヘッダは、ブラウザによって実行される必要のある動作がないことを意味する“no content(ノーコンテンツ)”を指定する。最後にＣＧＩスクリプトの第３行目はヘッダに続く空の行を送る。

【００５０】
図７に示した例示的スクリプトを使用して、クライアントはindex.htmlページを要求して、ページは示されたスクリプトに到達し、ページを受取ってから5分後に、ページは“beat(ビート)”機能を実行する。“beat”機能を実行しているとき、ページは１分ごとに“beatform”ドキュメントをサーバ12へ送る。このドキュメントはサーバに“nph-beat”というタイトルを付けられたＣＧＩプロセスを探すように命令する。サーバがこのプロセスを見付けると、このプロセスはサーバ12がＨＴＴＰヘッダおよびこれに続く空の行をクライアントへ向けて発行するようにする。この空の行は本質的に図２に示した“ＯＫ”の記述に相当し、一方でindex.htmlページによって送られる“beatform”は本質的に図２に示した“Ｉ’ｍ still here.”の記述に相当する。

【００５１】
図７に示したＣＧＩスクリプトに相当するＣ言語を次に示す：
【数２】

```
1. #include<stdio.h>
2. main ( ) {
3. printf("HTTP/1.0 204 No Content/n/n");
4. }
```

【００５２】
上記のコードにおいて、第1行目はＣ言語によって要求される；
第２行目は主なプログラムを規定している；
第３行目はＨＴＴＰヘッダをブラウザへ送り、このブラウザによって実行される必要のあ

る動作がないことを指定する；
第４行目はプログラムの定義を終了する。
【００５３】
多くの現在のワールドワイドウエブサーバ上では、ＣＧＩスクリプト名の名前はコンテンツヘッダを戻すために、“ nph ”で始まらなければならないことに注意すべきである。
【００５４】
上述の例のスクリプトは、図７の例示的実施形態に適しており、ここではサーバ12は、例えばファイルディレクトリ17からページを検索している。サーバ12がデータベースから情報を検索している場合、図５に示したように、クライアントによって変更することができ、スクリプトはわずかにより複雑である。上述の従来の技術の段落に記載したように、データベースアプリケーションに関係して解決された問題は非常に簡単で容易に統計的な情報を得るというだけでなく、2人のクライアントが同じデータベースに同じ時刻にアクセスしないことを確実にすることも含む。データベースアプリケーションの好ましい実施形態において、ビートは定常状態になる（すなわち、均一に定期的になる）。さらに、ページフォームでデータベースから戻されるフィールドは、ある種の自己識別をもつ（すなわち、戻される各ページは独特の識別フィールドをもつ）ので、サーバ12はクライアントユーザがいる場所が分かる。
【００５５】
データベースアプリケーションにおいて、ＣＧＩプロセス21におけるサーバスクリプトは、特定のデータベースアプリケーション用に設計されている。とくに、スクリプトは１）データベースが位置している場所、および２）特定のデータベース内におけるフィールドの取扱い方法を知らなければならない。さらにデータベースアプリケーションにおいて、サーバによって戻されるウエブページはサーバによってダイナミックに生成されなければならず、したがってデータベースからのフィールドが検索され、戻されたデータベース情報に基いて生成される。理想的にはウエブページは、適切な識別フィールドを含むビート機能によって戻されるフォーム（またはドキュメント）と一緒に、ビートのスクリプト機能を含む。
【００５６】
好ましい実施形態において、ページを生成して受理するサーバＣＧＩスクリプトは次の機能を実行する：
１．誰かによってページが使用されているとき、ページを別のユーザへ送ることを断る；
２．ビートのフォームを適切な識別フィールドでマークを付す；
３．必要なときは、データベース情報でデータフォームを埋める；
４．ビートが“中断している”ときは、戻されたフォームを受理することを断り、その間に他の者がこのページを参照する。
【００５７】
サーバ12上でデータベース情報を評価している第４のクライアントユーザの例を図８に示す。最初にクライアントユーザは“give me page X.（ページＸを出力してください）”でページ要求を行う。この要求は、プロセッサ20によって受取られ、データベース16へアクセスする。図８に示した例において、ページＸは在庫品記録であり、特定の製品に対する在庫品の量“Ｍ”を要求する。この場合に、プロセッサ20はコマンド“give me record ‘Ｍ’ so Ｉ can make page X.（記録‘Ｍ’を出力してください。それで私はページＸを作れます。）”で記録Ｍを検索するようにデータベース16に要求する。
【００５８】
データベース16内では、各記録は日付／時刻識別（ＤＴＩ）記録を含み、ＤＴＩ記録は特定の記録に対するログとして働く。ＤＴＩ情報は別のフィールドとしてデータベース内に記録される。したがって第４のクライアントユーザはページＸを要求して、データ記録“Ｍ”を求め、第４のユーザによる記録Ｍに対する要求は、データベース16内の記録Ｍに対するＤＴＩフィールドにおいて記録される。ＤＴＩフィールドは、図６のアクセスログ14内に示されたエントリに類似しているが、図８では簡潔にするために簡略形式を使用した

。図８の例では、第１のビートおよびそれに続く各ビートは1分間隔で行われ、最初のページ要求は１５時３６分に行われている。その結果記録Ｍは、第４のクライアントユーザが記録Ｍを要求するページを１５時３６分に要求したことを示す“user no. 4 at 15:36”の第１のアクセス記録エントリと関係するＤＴＩフィールドをもつ。

【００５９】

プロセッサ20が記録Ｍを要求して、ページＸを作った後で、データベース16は記録Ｍをプロセッサ20へ戻し、プロセッサ20は、ＣＧＩスクリプト（後でより詳しく記載する）と記録“Ｍ”情報とを含むページＸ（フォーム）を構築する。その後、ページスクリプト機能は、第４のクライアントユーザがビートをサーバ12へ毎分送り、それがページ上に残っていることを知らせる。これらのビートは“still on page X.（ページＸ上に依然として存在している）”という記述として図８の上から下へ年代順のシーケンスで示されている。後でより詳しく記載するように、第４のクライアントユーザからのビートはサーバ12へのページ情報を識別する。プロセッサ20は第４のクライアントユーザからビートを受取ると、ＣＧＩプロセス21をコールアップする。このプロセス21は別のスクリプト機能であり、サーバ12が“ＯＫ”を意味する空のラインで応答するようにする。さらにビートがプロセッサ20およびＣＧＩプロセス21によって受取られる度ごとに、データベース16はビートエントリをログすることによって記録Ｍに対するＤＴＩフィールドにおいて変更される。したがって例えば、データベース16内の記録Ｍが、第4のユーザがページＸ上に１５時３６分から（その後毎分）少なくとも１５時３９分まで（終了時間は第４のクライアントユーザがページ上に残ることによってサーバ12へビートを送り続ける時間に依存することが多い）ページＸ上に存在していたことを示す（部分的な）ＤＴＩフィールドをもつように示されている。

【００６０】

最後に、第４のクライアントユーザは要求をサーバ12へ送り、実際にサーバ12に記録“Ｍ”を変更するように要求してもよい。これは、例えば第４のクライアントユーザが特定の在庫品における一定数の在庫品“Ｍ”を購入する要求を出すときに行うことができる。第４のクライアントユーザは、記録Ｍに対する変更を示すフォームを戻すことによってこれを行う。このフォームはプロセッサ20によって受取られ、プロセッサ20はデータベース16に記録Ｍが修正されたことを知らせる。データベース16はこれにしたがって記録Ｍを修正する。その後、第４のクライアントユーザはページＸから離れて、ビートをサイレントにする。これによりサーバ12は別のクライアントユーザが現在記録Ｍを使用できることを知る。これは図９により詳しく示されており、第１のクライアントはページ４を要求し、ページ４を受取り、１（またはそれ以上の）ビートをサーバ12へ送り、ページ４上に残っていた時間を知らせる。ある点で、第１のクライアントはページ４を離れ、続いて第１のクライアントとサーバ12との間で沈黙が起こり、サーバ12に第１のクライアントがこのページから離れたことを知らせる。次に第２のクライアントは、要求“ｃ”において第４のクライアントにページを要求することができる。通信“Ｄ”として図９に示されたサーバ12からの応答は、スクリプト機能に依存し、第１のクライアントによって受取られた最後のビートと第２のクライアントによるページ４への要求との間の時間Ｔｃに依存することになる。サーバ12から第２のクライアントへの応答“Ｄ”への種々の可能性の例を後でさらに詳しく記載する。

【００６１】

第１に、表“Ｔ”からデータを表示するウエブページにおける例示的ページスクリプトを後で示す。このページはサーバＣＧＩスクリプトによって生成され、このサーバＣＧＩスクリプトでは要求された記録番号を得て、表“Ｔ”から読み取って、ユーザがこの記録を変更できるようにする。ユーザがこの記録をスクリプトへ戻すとき、記録はデータベースへ再び書き込まれ、次の記録（Ｍ＋１）がこの場所で検索される。この例は、単に本発明の1つの実施形態であって、これに限定されない：

【数３】

```
1. <html>
2. <head>
3.     <META HTTP-EQUIV="Expires" CONTENT="Thu,01 Dec 1994
16:00:00 GMT">
4. <script language="JavaScript">
5. function beat ( ) {
6. setTimeout("beat( )", 60000);
7. document.beatform.submit ( );
8. }
9. </script>
10. <title>Record m From Table T</title>
11. </head>
12. <body onLoad="beat ( ) ">
13.     <form name="beatform" method=POST action="/cgi-bin/nph-
dbbeat">
14. <input type=hidden name="whoAmI" value="T/m">
15. </form>
16. [introductory text would go here]
17. <form name="dataform" method=POST action="/cgi-bin/dbupdate">
18. <input type=hidden name="whoAmI" value="T/m">
19. [data fields, filled with information from table T record m go here]
20. </form>
21. ["footer" text would go here]
22. </body>
23. </html>
```

【００６２】

第４行目では、この例示的スクリプトはジャバスクリプトフォーマットで書き込まれているが、他のフォーマットも可能である。ある関連するスクリプト行の説明を次に示す：第３行目ではページをサーバから再びロードする。これは、誰かが第１のページを処理し、次に別のページへジャンプし、第１のページへ戻るといった１つの可能な問題に応答する。第１のページが最近のものであるときは、ブラウザはこのページをキャッシュメモリから取出す試行をして、最近のページは、人々が同じページにしばしば戻るという仮定を保ち続ける。ページがキャッシュから引き出されるとき、他の者がその間にこのページを検索していてもビート機能が再開する。“DBUPDATE”と呼ばれるＣＧＩスクリプトは、ページを保存することを拒否することによってこれを却下する。その代わりに、メッセージはユーザに戻されて、ユーザがそれを最早処理しないことを知らせる。追加の警告として、各生成されたページには、ページをダウンロードする前に有効期限が切れるようにする有効期限長が与えられている。有効期限の切れたページはキャッシュされない。他の代わりの技術を使用して、ページがキャッシュされるのを妨げて、この問題に対処してもよい

。
【００６３】
第５行目は、６０秒毎に“ビートフォーム”を提出するビート機能である。
【００６４】
第１０行目は、戻される表情報のタイトル、例えば“information for Mike Jones from the payroll table（ペイロール表からMike Jonesへの情報）”、または他の有益な情報を示す。
【００６５】
第１２行目は、ページがロードされたときにビートを開始することを示す。
【００６６】
第１３行目は、ビートが行われたときは必ずＣＧＩスクリプト“nph-dbbeat”を実行する。次に“whoAml”と呼ばれるファイルが、記録および表を特定する値と一緒に送られる。
【００６７】
第１６行目は、ユーザが調べるものを記述しているテキスト、またはおそらくは会社のロゴである。
【００６８】
第１７行目は、記録に関係するデータの全てを含む別のフォームである。
【００６９】
記録がペイロール記録であるときは、この記録は名前、従業員識別子、給与、税金情報、などを含んでもよい。完了すると、フォームは“DBUPDATE”へ提出され、情報を保存し、記録Ｍ＋１のデータをもつ新しいページを戻す。ジャバスクリプトを使用して、一連の“提出”ボタンを生成し、例えば最初の記録、最後の記録、前の記録、次の記録、または特定の記録を戻すことができる。この例では“次の記録”を戻す。
【００７０】
第１８行目は、フォーム内に含まれる“whoAml”フィールドである。
【００７１】
第２１行目は閉テキストを閉じることであり、おそらくは共同するページ、ヘルプページ、などをリンクしている。
【００７２】
第２２行目は、ページの本文を終了する。
【００７３】
第２３行目は、ページを終了する。
【００７４】
この例によって、２つのＣＧＩスクリプトがファイルと共に仕事をすることが必要とされる：一方のＣＧＩスクリプトはビートを処理し、他方のＣＧＩスクリプトはウエブページをデータベースから更新して処理する。ビートのスクリプトは、図８に示したように日付／時刻／識別子（ＤＴＩ）スタンプを指定された位置に押さなければならない。ＤＴＩスタンプは先のＤＴＩスタンプに上書きするか、またはそれらをログしてもよい。ＤＴＩスタンプが上書き可能であるとき、上書き条件の許可基準は次のものを含む：
１．ＤＴＩを提出したコンピュータは、先にそれを特定したのと同じコンピュータである；または、
２．現在の日付／時刻は先のＤＴＩスタンプの“タイムアウト”期間を超えている。
【００７５】
ビート機能用の例示的なＣＧＩスクリプトは、標準のＣ言語機能に関連して、以下に疑似（シュード）コードで示され、ここでは判読性を助けている：
【数４】

1. Identify the computer sending the request via getenv("REMOTE_ADDR"). Save this as "ipAddress."

2. Get the valueof the "whoAmI" parameter. This can be done using standard CGI utilities. Save this as "whoAmI."

3. Open a file named by the value of "whoAmI" for reading.

4. If this file exists:

5. read the contents. Save the ip address as "lastIP," and the date/time as "lastDT."

6. if the lastIP is the same as the whoAmI, or if the (time now) - lastDT is more than the timeout period, then:

7. re-open the file for writing.

8. Insert whoAmI and the current date/time.

9. else, this person is trying to steal someone else's heartbeat. Ignore him.

10. else, the file does not exist:

11. open the file for writing.

12. Insert whoAmI and the current date/time.

13. close the file.

14. write "HTTP/1.0 204 No Content\n\n" to stdout, to inform the client that the script has been processed.

15. terminate the script.

【００７６】
上の例に対して必要とされる第２のスクリプトは、データベースからウエブページを更新し、伝送するスクリプトである。この例では、これは“DBUPDATE”スクリプトと呼ばれ、この例は以下に、標準のＣ言語機能を参照して、疑似コードで示され、ここでは判読性を助けている：
【数５】

1. Identify the computer sending the request via getenv("REMOTE_ADDR"). Save this as "ipAddress."
2. Get the valueof the "whoAml" parameter. This can be done using standard CGI utilities. Save this as "whoAml."
3. Open a file named by the value of "whoAml" for reading.
4. If this file exists:
    5. read the contents. Save the ip address as "lastIP," and the date/time as "lastDT."
    6. if the lastIP is the same as the whoAml, or if the (time now) - lastDT is more than the timeout period, then:
        7. re-open the file for writing.
        8. Insert whoAml and the current date/time.
    9. else, this person is trying to steal someone else's heartbeat. Ignore him.
10. else, the file does not exist:
    11. update the current database record.
    12. build a new page with the heading "Record <m> has been successfully updated" and data for the next record (if any)
13. close the file.
14. terminate the script.

【００７７】
図９の実施形態では、各クライアントがサーバ12から同じ情報を受取ることを望むとき、上述の例示的スクリプトを使用して、サーバ12からクライアント１および２への応答の例を記載する。この例では、第１のクライアントがペイロール情報を要求すると仮定し、whoAmlファイルが“payroll/mike smith”を含むようにする。“remote addr”という変数（要求を出しているコンピュータのＩＰアドレスを含む）は現在、第１のクライアントに対するＩＰアドレスを含む。第１のクライアントがこのページを要求するとき、5つの可能な応答条件が存在する。

【００７８】
第１の可能性。サーバ12は、ファイル“payroll/mike smith”が存在し、それが第１のクライアントのＩＰアドレスを含み、そのＤＴＩスタンプの時刻が約1分前であることを知る。これは普通の場合であり、サーバ12に第1のクライアントがファイル“payroll/mike smith”を使用したこと、およびビート機能において第１のクライアントがページに対する第１のクライアントの制御を再び主張していることを知らせる。

【００７９】
第２の可能性。サーバ12が、第１のクライアントからページに対する要求を受取った後で、ファイル“payroll/mike smith”が存在し、第１のクライアントに対するＩＰアドレスを含むが、このファイルに対するＤＴＩスタンプが約1時間前であることを知るとき、サーバ12はこれを例外的な状況であると認識する。この場合、第１のクライアントはページを調べて、このページから離れ（したがってより近いＤＴＩスタンプ時刻を失う）、再びこのページに戻る。このページはキャッシュから取戻される（各ページはこのような状況になる前に有効期限が切れるので、通常は行われない）。キャッシュからファイルを取り戻すと、ページの前の所有者である第１のクライアントはここでページの制御を再び主張することができる。

【００８０】
第３の可能性。ファイル“payroll/mike smith”が存在しないことをサーバが知ると、エラー状況になる。ページを送ったＣＧＩがファイルを生成しているので、ファイルは必ず存在する。しかしながら仮定が設定されるときは、スクリプトをこのエラーから容易に取り戻すことができる。ファイルが幾分削除されると、再び生成され、第１のクライアントはページの所有者になる。別のクライアントがページを所有しているときは、他の所有者はページの制御を失うように再び割り当てられる。これは例外的な場合であり、頻繁に発生することではないが、ファイルはともかくも失われ、さらにビート期間中に別の所有者によって奪われることの両方が求められることに注意すべきである。それにも関わらず、上述の再生成および再割り当ては、このエラー状況を処理することができる。

【００８１】
第４の可能性。第1のクライアントからのページ要求に応答して、サーバ12が、ファイル“payroll/mike smith”が存在し、第２のクライアントのＩＰアドレスを含み、ＤＴＩスタンプの時刻が約1分前であることを知るとすると、サーバ12は、第１のクライアントが最早ページの所有者ではなく、アクセスできるようになるには、第２のクライアントがそのページを完了するまで待たなくてはならないことを第１のクライアントへ知らせる。第４の可能性は、第１のクライアントがこのページにアクセスし、このページを完了し、キャッシュからページを取り戻すことを試行し、一方でこの間に第２のクライアントがページを使用しているときに発生する。第１のクライアントのビートは終了し、第２のクライアントがこのページへのアクセスを与えられているので、第２のクライアントが終了するまで、第１のクライアントはキャッシュからこのページを取り戻すことはできない。

【００８２】
第５の可能性。最後の可能性は、サーバ12が、ファイル“payroll/mike smith”が存在し、第２のクライアントのＩＰアドレスを含み、時刻が約1時間前であることを認識するときに発生する。キャッシュからページを取り出して、1時間の時間間隔前に有効期限が切れるので、このケースも例外的である。第２のクライアントがこのページをあきらめて、第１のクライアントが現在このページに対する権利を再び主張できることが最も可能性が高い。処理しなければならない１つの問題を記載する。第１のクライアントがこのページ上に古いデータをもっていて、第１のクライアントが最後に第２のクライアントのより新しいデータに上書きをすることがある。これがページをキャッシュすべきでない第１の理由である。真に悲観的なロッキングについては、サーバ12は第１のユーザ１に、第１のユーザが古いデータを使用した可能性があること、および新しいデータで作業を再び行うべきことを警告する。

【００８３】
このやり方では、クライアントがウエブ上で新しいデータを使用しているときに、データベース内のデータをロックすることができる。サーバ12上のＣＧＩ　ＤＢＵＰＤＡＴＥスクリプトを使用して、データがページをロックしたクライアントから到来するときのみ、データをデータベースへ書込む。次にＣＧＩスクリプトは次の記録を同じクライアントへ送る。スクリプトはさらに、実際には生じない理想的世界において、ページを提出しているクライアントがページをロックしたクライアントと同一でない場合を取扱う。ページがキャッシュされず、ロックされたページのみがサーバ12によって送られるとすると、ロックされたページを検索する際に第２のクライアントは第１のクライアントにオーバーラップできない。しかしながら、上述のスクリプト例は、頑強性(robustness)のために（キャッシングを行うことができると仮定している）５つの可能性の各々を含むので、サーバ12はハングアップしない。

【００８４】
上述で開示したように、ウエブプロトコルがセッションを１つのページ要求および１つのページ応答のみであると定めても、本発明はＨＴＴＰ通信に対してセッション制御を行なうことが分かる。本発明において、サーバは、クライアントがページを調べている時間を判断でき、他のクライアントが同じデータベースエントリに既にアクセスしているときに

、データベース情報を含むページからクライアントをロックすることができる。

【００８５】

本発明は、現在最も実用的で好ましい実施形態であると考えられることに関して記載したが、本発明は記載した実施形態に限定されないと考えられ、逆に本発明の技術的範囲から逸脱しない種々の変更、および同等の構成を含むことを意図される。

【図面の簡単な説明】

【図１】　従来技術におけるインターネット上でのクライアントとサーバとの間のインターネットセッションの簡単な模式図。

【図２】　本発明のいくつかの例示的な実施形態におけるクライアントとサーバとの間で行われる通信を識別する模式的時間図。

【図３】　本発明のいくつかの例示的な実施形態におけるクライアントとサーバとの間で行われる通信を識別する模式的時間図。

【図４】　本発明のいくつかの例示的な実施形態におけるクライアントとサーバとの間で行われる通信を識別する模式的時間図。

【図５】　本発明の例示的な実施形態にしたがって通信するサーバ12の例示的な実施形態の模式的なブロック図。

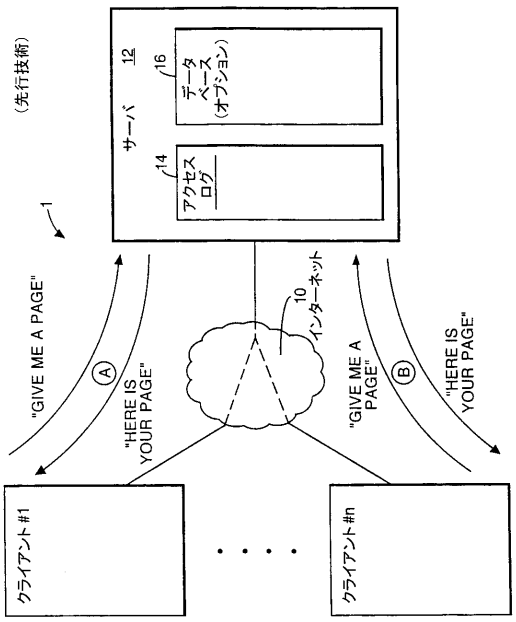
【図６】　本発明の例示的な実施形態にしたがう図５のサーバ12のアクセスログにおける例示的エントリを示す図。

【図７】　本発明のさらに別の例示的実施形態にしたがう図５の例示的サーバ12の模式図。

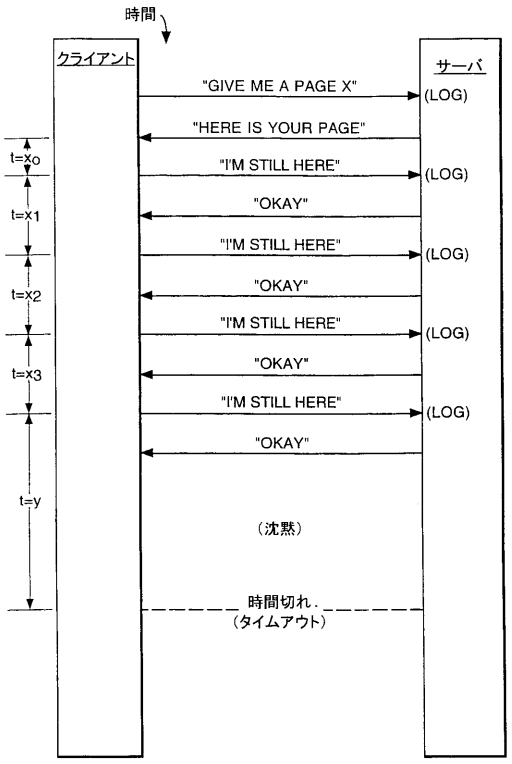
【図８】　クライアントとの例示的な通信セッション中のサーバ12の内部および外部の通信の模式的時間図。

【図９】　本発明の例示的な実施形態にしたがうサーバと2つの異なるクライアントとの通信セッションを示す模式的時間図。

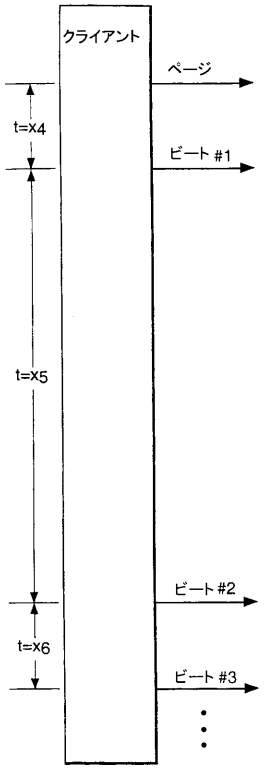
【図１】



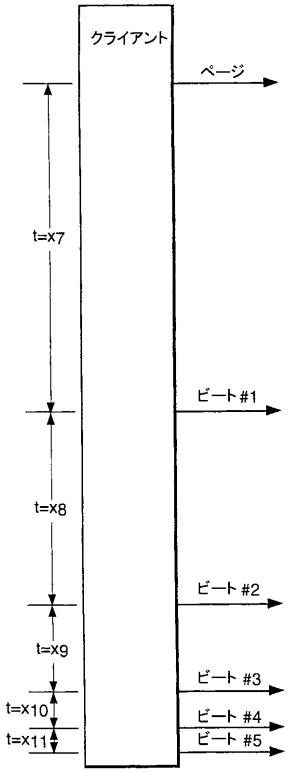
【図２】



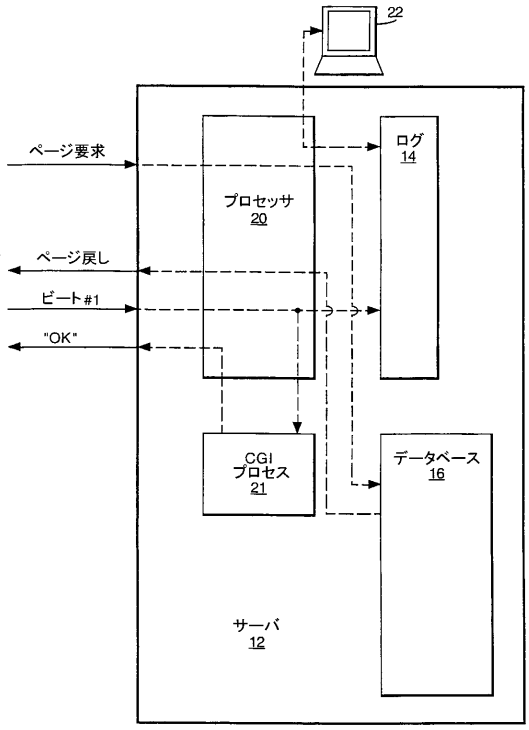
【図３】


クライアント
ページ
t=x4
ビート #1
t=x5
ビート #2
t=x6
ビート #3

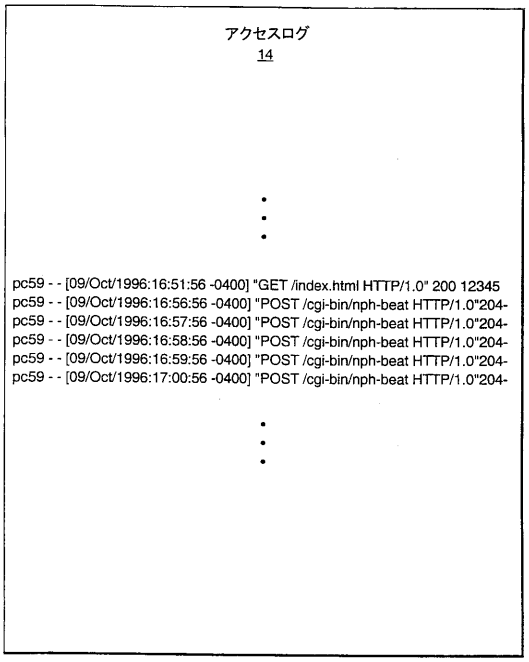
【図４】


クライアント
ページ
t=x7
ビート #1
t=x8
ビート #2
t=x9
ビート #3
t=x10
ビート #4
t=x11
ビート #5

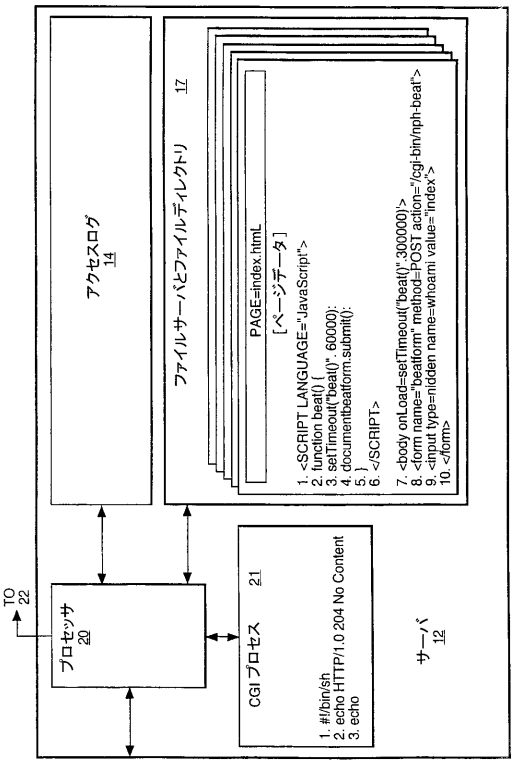
【図５】


22
ページ要求
ログ
14
プロセッサ
20
ページ戻し
ビート#1
"OK"
CGI
プロセス
21
データベース
16
サーバ
12

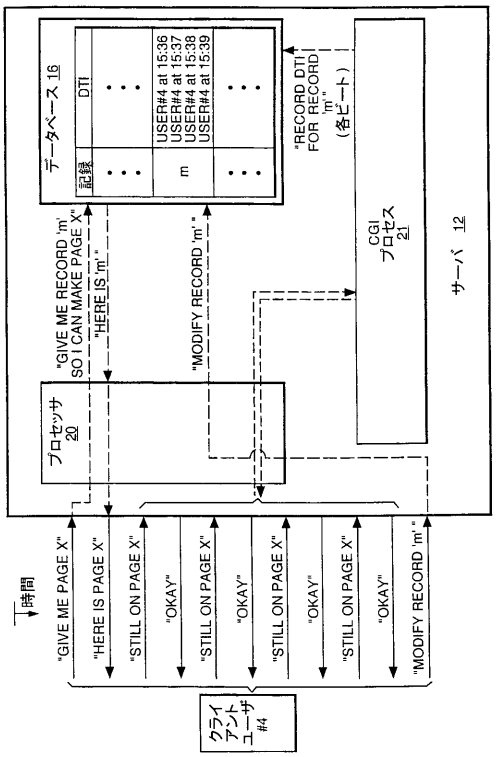
【図６】


アクセスログ
14
pc59 - - [09/Oct/1996:16:51:56 -0400] "GET /index.html HTTP/1.0" 200 12345
pc59 - - [09/Oct/1996:16:56:56 -0400] "POST /cgi-bin/nph-beat HTTP/1.0"204-
pc59 - - [09/Oct/1996:16:57:56 -0400] "POST /cgi-bin/nph-beat HTTP/1.0"204-
pc59 - - [09/Oct/1996:16:58:56 -0400] "POST /cgi-bin/nph-beat HTTP/1.0"204-
pc59 - - [09/Oct/1996:16:59:56 -0400] "POST /cgi-bin/nph-beat HTTP/1.0"204-
pc59 - - [09/Oct/1996:17:00:56 -0400] "POST /cgi-bin/nph-beat HTTP/1.0"204-

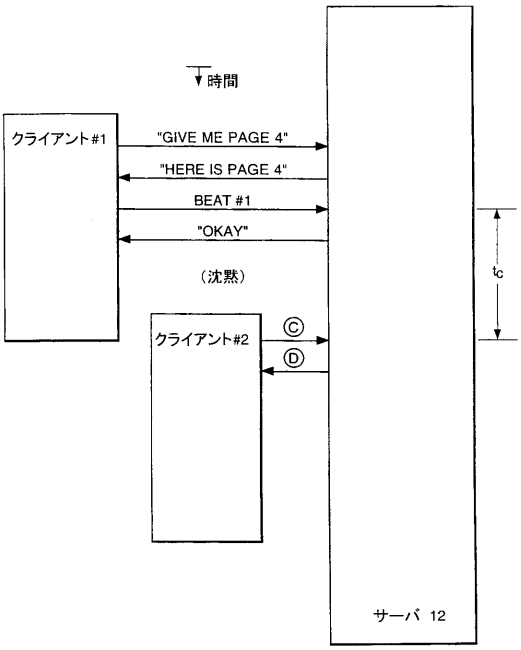
【図７】


アクセスログ
14
ファイルサーバとファイルディレクトリ
17
PAGE=index.html
[ページデータ]
1. <SCRIPT LANGUAGE="JavaScript">
2. function beat() {
3. setTimeout("beat()", 60000);
4. document.beatform.submit();
5. }
6. </SCRIPT>
7. <body onLoad=setTimeout("beat()", 300000)>
8. <form name="beatform" method=POST action="/cgi-bin/nph-beat">
9. <input type=hidden name=whoami value="index">
10. </form>
TO
22
プロセッサ
20
CGI プロセス
21
1. #!/bin/sh
2. echo HTTP/1.0 204 No Content
3. echo
サーバ
12

【図８】


データベース 16
記録
DTI
USER#4 at 15:36
USER#4 at 15:37
USER#4 at 15:38
USER#4 at 15:39
m
"RECORD DTI FOR RECORD 'm'"
(各ビート)
CGI プロセス 21
サーバ 12
プロセッサ 20
"GIVE ME RECORD 'm'
SO I CAN MAKE PAGE X"
"HERE IS 'm'."
"MODIFY RECORD 'm'."
時間
"GIVE ME PAGE X"
"HERE IS PAGE X"
"STILL ON PAGE X"
"OKAY"
"STILL ON PAGE X"
"OKAY"
"STILL ON PAGE X"
"OKAY"
"STILL ON PAGE X"
"OKAY"
"MODIFY RECORD 'm'"
クライアントユーザ #4

【図９】


時間
クライアント#1
"GIVE ME PAGE 4"
"HERE IS PAGE 4"
BEAT #1
"OKAY"
(沈黙)
クライアント#2
Ⓒ
Ⓓ
c
サーバ 12

フロントページの続き

(74)代理人 100095441
弁理士 白根 俊郎
(72)発明者 フェイト、フィル
アメリカ合衆国、バージニア州 ２２００３ アナンデイル、キャメロット・ドライブ ３６３７

審査官 波内 みさ

(56)参考文献 特開平０９－２６５４５６（ＪＰ，Ａ）
市嶋 洋平，WWW開発ツール － 更新系アプリケーションに対応する製品が増，日経バイト 特別増刊号 第１７１号，日本，日経ＢＰ社，１９９７年１１月 ８日，第171号，138～157
一歩先行くイントラネット － 第II部 イントラネットの構築に立ちふさがる壁を越える，日経オープンシステム，日本，日経ＢＰ社，１９９７年 ８月１５日，第53号，182～193
東道 零一，ＶＢＳｃｒｉｐｔ完全攻略ガイド 第二話 初めてのＶＢＳｃｒｉｐｔ，Ｖｉｓｕａｌ Ｂａｓｉｃ ｍａｇａｚｉｎｅ Ｖｏｌ．１１ 第６巻 第４号，日本，株式会社翔泳社，１９９７年 ３月１５日，第6巻 第4号，120～124
森下 哲次，仮想環境社会の展望 － ２．仮想環境サービスの展望，情報処理 第３８巻 第４号，日本，社団法人情報処理学会，１９９７年 ４月１５日，第38巻 第4号，274～279

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 17/30
G06F 12/00
G06F 13/00
G06F 15/00