(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
23 July 2015 (23.07.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/106692 A1

(51) International Patent Classification:
G06F 15/16 (2006.01)

(21) International Application Number:
PCT/CN2015/070746

(22) International Filing Date:
15 January 2015 (15.01.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
201410021149.8 16 January 2014 (16.01.2014) CN

(71) Applicant: **TENCENT TECHNOLOGY (SHENZHEN) COMPANY LIMITED** [CN/CN]; Room 403, East Block 2, SEG Park, Zhenxing Road, Futian District, Shenzhen City, Guangdong 518000 (CN).

(72) Inventor: **TANG, Wen**; Room 403, East Block 2, SEG Park, Zhenxing Road, Futian District, Shenzhen City, Guangdong 518000 (CN).

(74) Agent: **SHENPAT INTELLECTUAL PROPERTY AGENCY**; Room 1521, West Block, Guomao Building, Shenzhen, Guangdong 518014 (CN).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

(54) Title: WEBPAGE PUSHING METHOD, CLIENT, SERVER, AND SYSTEM

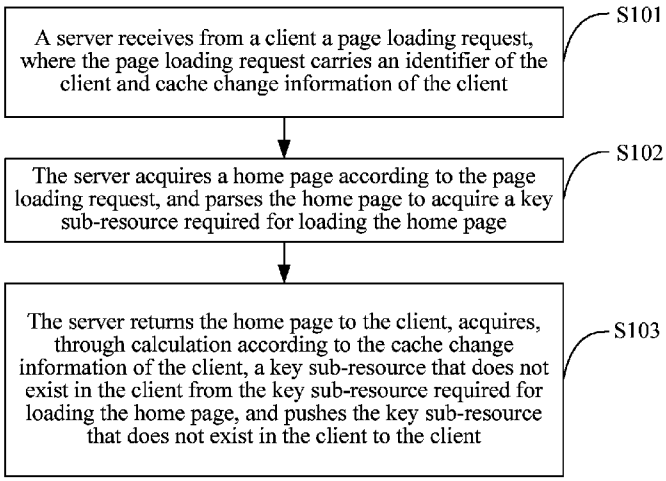

FIG. 1

(57) Abstract: The present disclosure relates to a webpage pushing method, client, server, and system. The method includes: sending, by a client, a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client; acquiring, by the server, a home page according to the page loading request, and parsing the home page to acquire a key sub-resource required for loading the home page; and returning, by the server, the home page to the client, acquiring, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client. The present disclosure can avoid that the server repeatedly pushes resources, which not only accelerates a page loading speed on the client, but also avoids unnecessary traffic overheads.

WO 2015/106692 A1

WEBPAGE PUSHING METHOD, CLIENT, SERVER, AND SYSTEM

RELATED APPLICATIONS

[0001] This application claims the priority benefit of Chinese Patent Application No. 201410021149.8, entitled "WEBPAGE PUSHING METHOD, CLIENT, SERVER, AND SYSTEM", filed on January 16, 2014, the content of which is incorporated by reference herein in its entirety for all purposes.

FIELD

[0002] The present disclosure relates to the field of Internet technologies, and in particular, to a webpage pushing method, client, server, and system.

BACKGROUND

[0003] When loading a webpage, a client generally sends a request to a server to pull page data; before receiving a request of a client, a server pushes some resources to the client in advance.

[0004] The current SPDY protocol supports a Push function, that is, the server can return multiple resources for one request of the client. In this way, the server may push, to the client in advance, a key sub-resource required for displaying a page, so that a page loading speed increases.

[0005] However, in the existing solution, although the server can push the key sub-resources in advance to improve the page loading speed, the server cannot determine in advance whether the client already has cache of the key sub-resources. If the client already has the cache of the key sub-resources, pushing by the server not only can increase overheads of network traffic of the client, but also can occupy a bandwidth, which causes that a speed of acquiring other resources by the client decreases, and lowers a page loading speed on the client.

SUMMARY

[0006] An embodiment provides a webpage pushing method, including:

sending, by a client, a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client;

acquiring, by the server, a home page according to the page loading request, and parsing the home page to acquire a key sub-resource required for loading the home page; and

returning, by the server, the home page to the client, acquiring, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client.

[0007] An embodiment further provides a webpage pushing method, including:

receiving, by a server, a page loading request sent by a client, where the page loading request carries an identifier of the client and cache change information of the client;

acquiring a home page according to the page loading request, and parsing the home page to acquire a key sub-resource required for loading the home page; and

returning the home page to the client, acquiring, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client.

[0008] An embodiment further provides a webpage pushing method, including:

sending, by a client, a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client;

receiving a home page returned by the server and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, where the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client; and

displaying the home page, and loading the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key sub-resource that exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0009] An embodiment further provides a webpage pushing system, including a client and a server, where

the client is configured to send a page loading request to the server, where the page loading request carries an identifier of the client and cache change information of the client; and

the server is configured to acquire a home page according to the page loading request, parse the home page to acquire a key sub-resource required for loading the home page, return the home page to the client, acquire, through calculation according to the cache change information of

the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client.

[0010] An embodiment further provides a webpage pushing server, including:

a request receiving module, configured to receive a page loading request sent by a client, where the page loading request carries an identifier of the client and cache change information of the client;

a parsing module, configured to acquire a home page according to the page loading request, and parse the home page to acquire a key sub-resource required for loading the home page; and

a pushing module, configured to return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client.

[0011] An embodiment further provides a webpage pushing client, including:

a sending module, configured to send a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client;

a receiving module, configured to receive a home page returned by the server and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, where the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client; and

a loading and display module, configured to display the home page, and load the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key sub-resource that exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0012] In the webpage pushing method, client, server, and system provided in the embodiments of the present disclosure, when requesting to load a page, a client sends local cache change information to a server, so that the server acquires an existing cache record of the client; when sending a home page to the client, the server obtains, according to the cache change information of the client, a key sub-resource that does not exist in the client from a key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client

to the client, thereby avoiding repeated pushing of resources. This pushing mechanism not only accelerates a page loading speed on the client, but also avoids unnecessary traffic overheads.

BRIEF DESCRIPTION OF THE DRAWINGS

[0013] FIG. 1 is a schematic flowchart of a first embodiment of a webpage pushing method according to the present disclosure;

[0014] FIG. 2 is a schematic flowchart of a second embodiment of a webpage pushing method according to the present disclosure;

[0015] FIG. 3 is a schematic flowchart of a third embodiment of a webpage pushing method according to the present disclosure;

[0016] FIG. 4 is a schematic flowchart of a fourth embodiment of a webpage pushing method according to the present disclosure;

[0017] FIG. 5 is a schematic flowchart of a fifth embodiment of a webpage pushing method according to the present disclosure;

[0018] FIG. 6 is a schematic flowchart of a sixth embodiment of a webpage pushing method according to the present disclosure;

[0019] FIG. 7 is a schematic flowchart of a seventh embodiment of a webpage pushing method according to the present disclosure;

[0020] FIG. 8 is a schematic flowchart of an eighth embodiment of a webpage pushing method according to the present disclosure;

[0021] FIG. 9 is a schematic flowchart of a ninth embodiment of a webpage pushing method according to the present disclosure;

[0022] FIG. 10 is a schematic flowchart of a tenth embodiment of a webpage pushing method according to the present disclosure;

[0023] FIG. 11a is a schematic structural diagram of a preferred embodiment of a webpage pushing system according to the present disclosure;

[0024] FIG. 11b is a schematic diagram of an interaction process between a client and a server in a webpage pushing system according to an embodiment of the present disclosure;

[0025] FIG. 12 is a schematic diagram of functional modules in a first embodiment of a webpage pushing server according to the present disclosure;

[0026] FIG. 13 is a schematic diagram of functional modules in a second embodiment of a webpage pushing server according to the present disclosure;

[0027] FIG. 14 is a schematic structural diagram of a pushing module in an embodiment of a webpage pushing server according to the present disclosure;

[0028] FIG. 15 is a schematic diagram of functional modules in a third embodiment of a webpage pushing server according to the present disclosure;

[0029] FIG. 16 is a schematic diagram of functional modules in a first embodiment of a webpage pushing client according to the present disclosure;

[0030] FIG. 17 is a schematic diagram of functional modules in a second embodiment of a webpage pushing client according to the present disclosure;

[0031] FIG. 18 is a schematic diagram of functional modules in a third embodiment of a webpage pushing client according to the present disclosure.

[0032] FIG. 19 depicts an exemplary environment incorporating certain disclosed embodiments; and

[0033] FIG. 20 depicts an exemplary computing system consistent with the disclosed embodiments.

DESCRIPTION OF EMBODIMENTS

[0034] It should be understood that the specific embodiments described herein are merely used to explain the present disclosure, but are not used to limit the present disclosure.

[0035] FIG. 19 depicts an exemplary environment 600 incorporating exemplary methods and systems for pushing webpage in accordance with various disclosed embodiments. As shown in FIG. 11, the environment 600 can include a server 604, clients 606 (e.g., a first client and a second client), and a communication network 602. The server 604 and the clients 606 may be coupled through the communication network 602 for information exchange. Although only one clients 606 and one server 604 are shown in the environment 600, any number of clients 606 or servers 604 may be included, and other devices may also be included.

[0036] The communication network 602 may include any appropriate type of communication network for providing network connections to the server 604 and clients 606 or among multiple servers 604 or clients 606. For example, the communication network 602 may

include the Internet or other types of computer networks or telecommunication networks, either wired or wireless.

[0037] A client, as used herein, may refer to any appropriate user client with certain computing capabilities, e.g., a personal computer (PC), a work station computer, a hand-held computing device (e.g., a tablet), a mobile client (e.g., a mobile phone or a smart phone), or any other client-side computing device. A server, as used herein, may refer to one or more server computers configured to provide certain server functionalities. A server may also include one or more processors to execute computer programs in parallel.

[0038] The server 604 and the clients 606 may be implemented on any appropriate computing platform. FIG. 20 shows a block diagram of an exemplary computing system 700 (or computer system 700) capable of implementing the server 604 and/or the clients 606. As shown in FIG. 20, the exemplary computer system 700 may include a processor 702, a storage medium 704, a monitor 706, a communication module 708, a database 710, peripherals 712, and one or more bus 714 to couple the devices together. Certain devices may be omitted and other devices may be included.

[0039] The processor 702 can include any appropriate processor or processors. Further, the processor 702 can include multiple cores for multi-thread or parallel processing. The storage medium 704 may include memory modules, e.g., Read-Only Memory (ROM), Random Access Memory (RAM), and flash memory modules, and mass storages, e.g., CD-ROM, U-disk, removable hard disk, etc. The storage medium 704 may store computer programs for implementing various processes, when executed by the processor 702.

[0040] The monitor 706 may include display devices for displaying contents in the computing system 700. The peripherals 712 may include I/O devices such as keyboard and mouse.

Further, the communication module 708 may include network devices for establishing connections through the communication network 602. The database 710 may include one or more databases for storing certain data and for performing certain operations on the stored data. In operation, the clients 606 may cause the server 604 to perform certain actions. The server 604 may be configured to provide structures and functions for such actions and operations. In various embodiments, a client involved in the disclosed methods and systems can include the clients 606, while a server involved in the disclosed methods and systems can include the server 604. The methods and systems disclosed in accordance with various embodiments can be executed by a computer system.

[0041] Various embodiments provide methods and systems for pushing webpage. The methods and systems are illustrated in various examples described herein.

[0042] As shown in FIG. 1, a first embodiment of the present disclosure provides a webpage pushing method, including:

[0043] Step 101: A server receives from a client a page loading request, where the page loading request carries an identifier of the client and cache change information of the client.

[0044] A running environment of the method in this embodiment of the present disclosure involves the client and the server, where the client may be a PC, or a mobile client capable of displaying a webpage, such as a mobile phone or a tablet computer. The client may display different webpages according to a website or webpage clicking operation instruction input by a user, where webpage data is pulled by the client by sending a request to the server, or is actively pushed to the client by the server.

[0045] To improve a webpage loading speed on the client, generally, the server may return multiple resources for one request of the client. In this way, the server may push, to the client in advance, a key sub-resource required for displaying a page, so that the page loading speed increases.

[0046] In the existing pushing solution, the server cannot determine in advance whether the client already has cache of the key sub-resource; therefore, the server pushes a key sub-resource that is already cached in the client, thereby increasing network traffic overheads on the client, occupying a bandwidth, and reducing the page loading speed on the client.

[0047] In this embodiment, through the Q-Push protocol, when pushing resources to the client, the server may ensure, according to the cache change information recorded in the client, that a key sub-resource pushed by the server is the key sub-resource that does not exist in the client, so as to achieve an effect of improving a pushing speed and reducing traffic overheads on the client.

[0048] Specifically, at first, when a user needs to display a webpage, the client sends the page loading request to the server, where the page loading request may be sent to the server in a manner of an http request.

[0049] The page loading request carries information such as a URL address of a webpage that the client requests to display, and meanwhile, the page loading request further carries the identifier of the client and local cache change information recorded in the client.

[0050] The identifier of the client may be a digital identifier that is pre-generated for the client by the server, and is used as a unique identifier of the client to mark the client. As one

implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm.

[0051] Certainly, the identifier of the client may also be a unique identifier, where the unique identifier is generated by the client or another third-party server under agreement between the client and the server.

[0052] The client locally saves the identifier, and each time when the client interacts with the server subsequently, the client adds the identifier in the request.

[0053] The local cache change information recorded in the client may include that: the client adds or decreases cache data corresponding to one URL.

[0054] The client records, by maintaining a cache recorder, change information of each cache by the client, records the added URL or a hash value corresponding to the URL in an ADD array, and records the deleted URL or a hash value corresponding to the URL in a REMOVE array.

[0055] Using an http request as an example, when the client sends the http request, the client adds fields "globally unique identifier (GUID)" and "Q-Push:" in a header of the http request. The GUID (which is a digital identifier that has a binary length being 128 bits, is generated according to an algorithm, and may be used or marking the client) is identifier information delivered by the server. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[0056] Step S102: The server acquires a home page according to the page loading request, and parses the home page to acquire a key sub-resource required for loading the home page.

[0057] After receiving the page loading request sent by the client, the server acquires corresponding page data according to the URL address carried in the page loading request, uses the page data as home page data returned to the client.

[0058] Meanwhile, the server further parses the home page data to obtain the key sub-resource required for loading the home page on the client.

[0059] The key sub-resource is a resource playing a critical role on parsing and rendering one HTML, where the resource generally is a script language (JS)/cascading style sheet (CSS) of the client or the like.

[0060] The sub-resource is a resource opposite to a main resource of a home page. The main resource refers to one entire HTML page, and the sub-resource refers to files, such as pictures, frames, or scripts, externally cited on the HTML page, and is opposite to the main resource.

[0061] Generally, when returning the home page data to the client, the server pushes, to the client according to the request of the client or actively, the key sub-resource required for loading the home page.

[0062] The server in this embodiment may acquire, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the acquired key sub-resource to the client, so as to save traffic overheads and improve the webpage loading speed on the client.

[0063] Step S103: The server returns the home page to the client, acquires, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client.

[0064] In this embodiment, it is considered that in the key sub-resources required for loading the home page on the client, some key sub-resources may be cached on the client; therefore, the server may only push the key sub-resource that does not exist in the client to the client, thereby saving traffic overheads and improving the webpage loading speed on the client.

[0065] During specific operation, the server first acquires a cache change situation of the client from historical records according to the identifier of the client carried in the page loading request, and obtains, through calculation according to cache change information currently sent by the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page; therefore, the key sub-resource that does not exist in the client may be acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client.

[0066] Further, when returning the home page to the client, the server may return the key sub-resource that does not exist in the client to the client in a manner of SPDY PUSH. The SPDY is the enhanced protocol of the HTTP protocol, where the function of the protocol includes multi-path multiplexing of a data stream, a request priority, and HTTP header compression.

[0067] On the side of the client, the client may cache the key sub-resource pushed by the server. When displaying the home page, the client may load the key sub-resource required for displaying the home page, where the key sub-resource required for the home page and loaded by the client includes the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0068] Compared with the prior art, the client does not need to send the request to the server again to acquire the key sub-resource that is not cached locally, and the server may not push the key sub-resource that already exists in the client to the client repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0069] As shown in FIG. 2, a second embodiment of the present disclosure provides a webpage pushing method, after the sending, by a client, a page loading request to a server in step S101, the method further includes:

[0070] Step S104: The server records the cache change information of the client in a cache of the server in correspondence to the identifier of the client.

[0071] A difference between this embodiment and the first embodiment shown in FIG. 1 lies in that, this embodiment further includes a solution in which the server records and caches the acquired cache change information of the client.

[0072] Specifically, considering that the server receives page loading requests of different clients, to improve page pushing efficiency, in this embodiment, after receiving the page loading request sent by the client, the server extracts the cache change information of the client from the page loading request, and records the cache change information of the client in the cache of the server in correspondence to the identifier of the client.

[0073] During subsequent specific operation, the server first searches for the cache change information of the client from the cache of the server according to the identifier of the client carried in the page loading request, and then acquires, through calculation according to the cache change information of the client, the key sub-resource that already exists from the key sub-resource required for loading the home page.

[0074] Then, the key sub-resource that does not exist in the client is acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client is pushed to the client.

[0075] In this embodiment, by using the foregoing embodiment, when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource

pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency.

[0076] As shown in FIG. 3, a third embodiment of the present disclosure provides a webpage pushing method, before the sending, by a client, a page loading request to a server in step S101, the method further includes:

[0077] Step S90: The server generates the identifier for the client, and delivers the identifier to the client.

[0078] Step S100: The client records local cache change information.

[0079] A difference between this embodiment and the second embodiment shown in FIG. 2 lies in that, this embodiment further includes the solution in which the server generates the identifier for the client and the client records the local cache change information.

[0080] Specifically, the server generates a digital identifier for the client, where the digital identifier is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm. After generating the identifier for the client, the server delivers the identifier to the client.

[0081] The local cache change information includes that: the client adds or decreases cache data corresponding to one URL.

[0082] The process of recording, by the client, the local cache change information specifically includes:

recording, in an add array, the URL added by the client or a hash value corresponding to the URL, and recording, in a remove array, the URL deleted by the client or a hash value corresponding to the URL.

[0083] Recording the hash value corresponding to the URL may decrease the traffic overheads. The involved hash algorithm may be defined according to needs. The following provides one reference hash algorithm:

```
uint32 SuperFastHash(const char * data, int len) {
  uint32_t hash = len, tmp;
```

```
int rem;
if (len <= 0 || data == NULL)
  return 0;
rem = len & 3;
len >>= 2;
/* Main loop */
for (; len > 0; len--) {
  hash  += get16bits(data);
  tmp   = (get16bits(data + 2) << 11) ^ hash;
  hash  = (hash << 16) ^ tmp;
  data  += 2 * sizeof(uint16_t);
  hash  += hash >> 11;
}
/* Handle end cases */
switch (rem) {
  case 3:
    hash += get16bits(data);
    hash ^= hash << 16;
    // Treat the final character as signed. This ensures all platforms behave
    // consistently with the original x86 code.
    hash ^= static_cast<signed char>(data[sizeof(uint16_t)]) << 18;
    hash += hash >> 11;
    break;
  case 2:
    hash += get16bits(data);
    hash ^= hash << 11;
```

```
            hash += hash >> 17;
            break;
          case 1:
            hash += static_cast<signed char>(*data);
            hash ^= hash << 10;
            hash += hash >> 1;
        }
        /* Force "avalanching" of final 127 bits */
        hash ^= hash << 3;
        hash += hash >> 5;
        hash ^= hash << 4;
        hash += hash >> 17;
        hash ^= hash << 25;
        hash += hash >> 6;
        return hash;
      }.
```

[0084] In this embodiment, by using the foregoing solution, the server generates the identifier for the client, and delivers the identifier to the client; the client records the local cache change information; when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency.

[0085] As shown in FIG. 4, a fourth embodiment of the present disclosure provides a webpage pushing method, where after the returning, by the server, the home page to the client, acquiring, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client in step S103, the method further includes:

[0086] Step S105: The client updates the local cache change information, specifically including:

removing, by the client from the add array or the remove array, a URL pushed by the server or a hash value corresponding to the URL.

[0087] A difference between this embodiment and the third embodiment shown in FIG. 3 lies in that: this embodiment further includes a solution in which the client updates the local cache change information.

[0088] After receiving the key sub-resource pushed by the server and completing page loading and displaying, the client removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server or a corresponding hash value, which on one hand can save client cache resources, on the other hand ensures accuracy of the cache change information of the client, thereby further accelerating the page loading speed on the client.

[0089] When accessing a new webpage next, the client repeats the foregoing process.

[0090] In this embodiment, by using the foregoing solution, the server generates the identifier for the client, and delivers the identifier to the client; the client records the local cache change information; when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency. After receiving the key sub-resource pushed by the server and completes page loading and displaying, the client

removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server or a corresponding hash value, which on one hands can save cache resources of the client, on the other hand ensures accuracy of the cache change information of the client, and further accelerates the page loading speed on the client.

[0091] As shown in FIG. 5, a fifth embodiment of the present disclosure provides a webpage pushing method, including:

[0092] Step S201: A server receives a page loading request sent by a client, where the page loading request carries an identifier of the client and cache change information of the client.

[0093] Step S202: Acquire a home page according to the page loading request, and parse the home page to acquire a key sub-resource required for loading the home page.

[0094] Step S203: Return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client.

[0095] A running environment of the method in this embodiment of the present disclosure involves the client and the server, where the client may be a PC, or a mobile client capable of displaying a webpage, such as a mobile phone or a tablet computer. The client may display different webpages according to a website or webpage clicking operation instruction input by a user, where webpage data is pulled by the client by sending a request to the server, or is actively pushed to the client by the server.

[0096] To improve a webpage loading speed on the client, generally, the server may return multiple resources for one request of the client. In this way, the server may push, to the client in advance, the key sub-resource required for displaying the page, so that the page loading speed increases. In the existing pushing solution, the server cannot determine in advance whether the client already has cache of the key sub-resource; therefore, the server pushes a key sub-resource that is already cached in the client, thereby increasing network traffic overheads on the client, occupying a bandwidth, and reducing the page loading speed on the client.

[0097] In this embodiment, through the Q-Push protocol, when pushing resources to the client, the server may ensure, according to the cache change information recorded in the client, that a key sub-resource pushed by the server is the key sub-resource that does not exist in the client, so as to achieve an effect of improving a pushing speed and reducing traffic occupation on the client.

[0098] Specifically, at first, when a user needs to display a webpage, the client sends the page loading request to the server, where the page loading request may be sent to the server in a manner of an http request.

[0099] The page loading request carries information such as a URL address of a webpage that the client requests to display, and meanwhile, the page loading request further carries the identifier of the client and local cache change information recorded in the client.

[00100] The identifier of the client may be a digital identifier that is pre-generated for the client by the server, and is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm.

[00101] Certainly, the identifier of the client may also be a unique identifier, where the unique identifier is generated by the client or another third-party server under agreement between the client and the server.

[00102] The client locally saves the identifier, and each time when the client interacts with the server subsequently, the client adds the identifier in the request.

[00103] The local cache change information recorded in the client may include that: the client adds or decreases cache data corresponding to one URL.

[00104] The client may maintain a cache recorder, record change information of each cache by the client, record the added URL or a hash value corresponding to the URL in an ADD array, and record the deleted URL or a hash value corresponding to the URL in a REMOVE array.

[00105] Using an http request as an example, when the client sends the http request, the client adds fields "GUID" and "Q-Push:" in a header of the http request. The GUID (which is a digital identifier that has a binary length being 128 bits, is generated according to an algorithm, and may be used for marking the client) is identifier information delivered by the server. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[00106] After receiving the page loading request sent by the client, the server acquires corresponding page data according to the URL address carried in the page loading request, uses the page data as home page data returned to the client.

[00107] Meanwhile, the server further parses the home page data to obtain the key sub-resource required for loading the home page on the client.

[0100] The key sub-resource is a resource playing a critical role on parsing and rendering one HTML, where the resource generally is a JS/CSS of the client, or the like.

[0101] The sub-resource is a resource opposite to a main resource of a home page. The main resource refers to one entire HTML page, and the sub-resource refers to files, such as pictures, frames, or scripts, externally cited on the HTML page, and is opposite to the main resource.

[0102] Generally, when returning the home page data to the client, the server pushes, to the client according to the request of the client or actively, the key sub-resource required for loading the home page.

[0103] The server in this embodiment may acquire, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, so as to save traffic overheads and improve the webpage loading speed on the client.

[0104] In this embodiment, it is considered that in the key sub-resources required for loading the home page on the client, some key sub-resources may be cached on the client; therefore, the server may only push the key sub-resource that does not exist in the client to the client, thereby saving traffic overheads and improving the webpage loading speed on the client.

[0105] During specific operation, the server first acquires a cache change situation of the client from historical records according to the identifier of the client carried in the page loading request, and obtains, through calculation according to cache change information currently sent by the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page; therefore, the key sub-resource that does not exist in the client may be acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client.

[0106] Further, when returning the home page to the client, the server may return the key sub-resource that does not exist in the client to the client in a manner of SPDY PUSH. The SPDY is the enhanced protocol of the HTTP protocol, where the function of the protocol includes multi-path multiplexing of a data stream, a request priority, and HTTP header compression.

[0107] On the side of the client, the client may cache the key sub-resource pushed by the server. When displaying the home page, the client may load the key sub-resource required for displaying the home page, where the key sub-resource required for the home page and loaded by the client includes the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0108] Therefore, compared with the prior art, the client does not need to send the request to the server again to acquire the key sub-resource that is not cached locally, and the server may not push the key sub-resource that already exists in the client to the client repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0109] As shown in FIG. 6, a sixth embodiment of the present disclosure provides a webpage pushing method. On the basis of the fifth embodiment shown in FIG. 5, after the receiving, by a server, a page loading request sent by a client in step S201, the method further includes:

[0110] Step S204: The server records the cache change information of the client in a cache of the server in correspondence to the identifier of the client.

[0111] A difference between this embodiment and the fifth embodiment shown in FIG. 5 lies in that, this embodiment further includes a solution in which the server records and caches the acquired cache change information of the client.

[0112] Specifically, considering that the server receives page loading requests of different clients, to improve page pushing efficiency, in this embodiment, after receiving the page loading request sent by the client, the server extracts the cache change information of the client from the page loading request, and records the cache change information of the client in the cache of the server in correspondence to the identifier of the client.

[0113] During subsequent specific operation, the server first searches for the cache change information of the client from the cache of the server according to the identifier of the client carried in the page loading request, and then acquires, through calculation according to the cache change information of the client, the key sub-resource that already exists from the key sub-resource required for loading the home page.

[0114] Then, the key sub-resource that does not exist in the client is acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client is pushed to the client.

[0115] In this embodiment, by using the foregoing solution, when requesting to load a page, the client sends local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource

pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency.

[0116] As shown in FIG. 7, a seventh embodiment of the present disclosure provides a webpage pushing method. On the basis of the sixth embodiment shown in FIG. 6, before the receiving, by a server, a page loading request sent by a client in step S201, the method further includes:

[0117] Step S200: The server generates the identifier for the client, and delivers the identifier to the client.

[0118] A difference between this embodiment and the sixth embodiment shown in FIG. 6 lies in that, this embodiment further includes a solution in which the server generates the identifier for the client.

[0119] Specifically, the server generates a digital identifier for the client, where the digital identifier is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm. After generating the identifier for the client, the server delivers the identifier to the client. By using the foregoing solution, flexibility of interaction between the client and the server is increased, so that the server pushes corresponding network resources to the client according to different client identifiers.

[0120] As shown in FIG. 8, an eighth embodiment of the present disclosure provides a webpage pushing method, including:

[0121] Step S301: A client sends a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client.

[0122] Step S302: Receive a home page returned by the server and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, where the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client.

[0123] Step S303: Display the home page, and load the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key

sub-resource that exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0124] A running environment of the method in this embodiment of the present disclosure involves the client and the server, where the client may be a PC, or a mobile client capable of displaying a webpage, such as a mobile phone or a tablet computer. The client may display different webpages according to a website or webpage clicking operation instruction input by a user, where webpage data is pulled by the client by sending a request to the server, or is actively pushed to the client by the server.

[0125] To improve a webpage loading speed on the client, generally, the server may return multiple resources for one request of the client. In this way, the server may push, to the client in advance, the key sub-resource required for displaying the page, so that the page loading speed increases. In the existing pushing solution, the server cannot determine in advance whether the client already has cache of the key sub-resource; therefore, the server pushes a key sub-resource that is already cached in the client, thereby increasing network traffic overheads on the client, occupying a bandwidth, and reducing the page loading speed on the client.

[0126] In this embodiment, through the Q-Push protocol, when pushing resources to the client, the server may ensure, according to the cache change information recorded in the client, that a key sub-resource pushed by the server is the key sub-resource that does not exist in the client, so as to achieve an effect of improving a pushing speed and reducing traffic occupation on the client.

[0127] Specifically, at first, when a user needs to display a webpage, the client sends the page loading request to the server, where the page loading request may be sent to the server in a manner of an http request.

[0128] The page loading request carries information such as a URL address of a webpage that the client requests to display, and meanwhile, the page loading request further carries the identifier of the client and local cache change information recorded in the client.

[0129] The identifier of the client may be a digital identifier that is pre-generated for the client by the server, and is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm.

[0130] Certainly, the identifier of the client may also be a unique identifier, where the unique identifier is generated by the client or another third-party server under agreement between the client and the server.

[0131] The client locally saves the identifier, and each time when the client interacts with the server subsequently, the client adds the identifier in the request.

[0132] The local cache change information recorded in the client may include that: the client adds or decreases cache data corresponding to one URL.

[0133] The client may maintain a cache recorder, record change information of each cache by the client, record the added URL or a hash value corresponding to the URL in an ADD array, and record the deleted URL or a hash value corresponding to the URL in a REMOVE array.

[0134] Using an http request as an example, when the client sends the http request, the client adds fields "GUID" and "Q-Push:" in a header of the http request. The GUID (which is a digital identifier that has a binary length being 128 bits, is generated according to an algorithm, and may be used for marking the client) is identifier information delivered by the server. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[0135] After receiving the page loading request sent by the client, the server acquires corresponding page data according to the URL address carried in the page loading request, uses the page data as home page data returned to the client.

[0136] Meanwhile, the server further parses the home page data to obtain the key sub-resource required for loading the home page on the client.

[0137] The key sub-resource is a resource playing a critical role on parsing and rendering one HTML, where the resource generally is a JS/CSS of the client, or the like.

[0138] The sub-resource is a resource opposite to a main resource of a home page. The main resource refers to one entire HTML page, and the sub-resource refers to files, such as pictures, frames, or scripts, externally cited on the HTML page, and is opposite to the main resource.

[0139] Generally, when returning the home page data to the client, the server pushes, to the client according to the request of the client or actively, the key sub-resource required for loading the home page.

[0140] The server in this embodiment may acquire, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, so as to save traffic overheads and improve the webpage loading speed on the client.

[0141] In this embodiment, it is considered that in the key sub-resources required for loading the home page on the client, some key sub-resources may be cached on the client; therefore,

the server may only push the key sub-resource that does not exist in the client to the client, thereby saving traffic overheads and improving the webpage loading speed on the client.

[0142] During specific operation, the server first acquires a cache change situation of the client from historical records according to the identifier of the client carried in the page loading request, and obtains, through calculation according to cache change information currently sent by the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page; therefore, the key sub-resource that does not exist in the client may be acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client.

[0143] Further, when returning the home page to the client, the server may return the key sub-resource that does not exist in the client to the client in a manner of SPDY PUSH. The SPDY is the enhanced protocol of the HTTP protocol, where the function of the protocol includes multi-path multiplexing of a data stream, a request priority, and HTTP header compression.

[0144] On the side of the client, the client may cache the key sub-resource pushed by the server. When displaying the home page, the client may load the key sub-resource required for displaying the home page, where the key sub-resource required for the home page and loaded by the client includes the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0145] Therefore, compared with the prior art, the client does not need to send the request to the server again to acquire the key sub-resource that is not cached locally, and the server may not push the key sub-resource that already exists in the client to the client repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0146] As shown in FIG. 9, a ninth embodiment of the present disclosure provides a webpage pushing method. On the basis of the eighth embodiment shown in FIG. 8, before the sending, by a client, a page loading request to a server in step S301, the method further includes:

[0147] Step S300: The client records local cache change information, where the local cache change information includes that: the client adds or decreases cache data corresponding to one URL.

[0148] A difference between this embodiment and the eighth embodiment shown in FIG. 8 lies in that: this embodiment further includes a solution in which the client records the local cache change information.

[0149] Specifically, the local cache change information includes that: the client adds or decreases cache data corresponding to one URL.

[0150] The process of recording, by the client, the local cache change information specifically includes:

recording, in an add array, the URL added by the client or a hash value corresponding to the URL, and recording, in a remove array, the URL deleted by the client or a hash value corresponding to the URL.

[0151] Recording the hash value corresponding to the URL may decrease the traffic overheads. The involved hash algorithm may be defined according to needs. The following provides one reference hash algorithm:

```
uint32 SuperFastHash(const char * data, int len) {
  uint32_t hash = len, tmp;
  int rem;
  if (len <= 0 || data == NULL)
    return 0;
  rem = len & 3;
  len >>= 2;
  /* Main loop */
  for (; len > 0; len--) {
    hash  += get16bits(data);
    tmp    = (get16bits(data + 2) << 11) ^ hash;
    hash   = (hash << 16) ^ tmp;
    data  += 2 * sizeof(uint16_t);
     hash  += hash >> 11;
  }
  /* Handle end cases */
  switch (rem) {
    case 3:
```

```
    hash += get16bits(data);
    hash ^= hash << 16;
    // Treat the final character as signed. This ensures all platforms behave
    // consistently with the original x86 code.
    hash ^= static_cast<signed char>(data[sizeof(uint16_t)]) << 18;
    hash += hash >> 11;
    break;
  case 2:
    hash += get16bits(data);
    hash ^= hash << 11;
    hash += hash >> 17;
    break;
  case 1:
    hash += static_cast<signed char>(*data);
    hash ^= hash << 10;
    hash += hash >> 1;
 }
 /* Force "avalanching" of final 127 bits */
 hash ^= hash << 3;
 hash += hash >> 5;
 hash ^= hash << 4;
 hash += hash >> 17;
 hash ^= hash << 25;
 hash += hash >> 6;
 return hash;
}.
```

[0152] In this embodiment, by using the foregoing embodiment, the client records local cache change information; when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency.

[0153] As shown in FIG. 10, a tenth embodiment of the present disclosure provides a webpage pushing method. Based on the ninth embodiment shown in FIG. 9, after the displaying the home page, and loading the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key sub-resource that exists in the client and a key sub-resource that does not exist in the client previously and is pushed by the server in step S303, the method further includes:

[0154] Step S304: The client updates the local cache change information.

[0155] A difference between this embodiment and the ninth embodiment shown in FIG. 9 lies in that: this embodiment further includes a solution in which the client updates the local cache change information.

[0156] After receiving the key sub-resource pushed by the server and completing page loading and displaying, the client removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server or a corresponding hash value, which on one hand can save client cache resources, on the other hand ensures accuracy of the cache change information of the client, thereby further accelerating the page loading speed on the client.

[0157] When accessing a new webpage next, the client repeats the foregoing process.

[0158] In this embodiment, by using the foregoing solution, the server generates the identifier for the client, and delivers the identifier to the client; the client records the local cache change information; when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change

information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency. After receiving the key sub-resource pushed by the server and completes page loading and displaying, the client removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server or a corresponding hash value, which on one hands can save cache resources of the client, on the other hand ensures accuracy of the cache change information of the client, and further accelerates the page loading speed on the client.

[0159] As shown in FIG. 11a, a preferred embodiment of the present disclosure further provides a webpage pushing system, including a client 501 and a server 502, where

the client 501 is configured to send a page loading request to the server 502, where the page loading request carries an identifier of the client 501 and cache change information of the client 501; and

the server 502 is configured to acquire a home page according to the page loading request, parse the home page to acquire a key sub-resource required for loading the home page, return the home page to the client 501, acquire, according to the cache change information of the client 501, a key sub-resource that does not exist in the client 501 from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client 501 to the client 501.

[0160] The solution in this embodiment involves the client 501 and the server 502, where the client 501 may be a PC, or a mobile client capable of displaying a webpage, such as a mobile phone or a tablet computer. The client 501 may display different webpages according to a website or webpage clicking operation instruction input by a user, where webpage data is pulled by the client 501 by sending a request to the server 502, or is actively pushed to the client 501 by the server 502.

[0161] To improve a webpage loading speed on the client 501, generally, the server 502 may return multiple resources for one request of the client 501. In this way, the server 502 may push, to the client 501 in advance, the key sub-resource required for displaying the page, so that the

page loading speed increases. In the existing pushing solution, the server 502 cannot determine in advance whether the client 501 already has cache of the key sub-resource; therefore, the server 502 pushes a key sub-resource that is already cached in the client 501, thereby increasing network traffic overheads on the client 501, occupying a bandwidth, and reducing the page loading speed on the client 501.

[0162] In this embodiment, through the Q-Push protocol, when pushing resources to the client 501, the server 502 may ensure, according to the cache change information recorded in the client 501, that a key sub-resource pushed by the server 502 is the key sub-resource that does not exist in the client 501, so as to achieve an effect of improving a pushing speed and reducing traffic occupation on the client 502.

[0163] Specifically, at first, when a user needs to display a webpage, the client 501 sends the page loading request to the server 502, where the page loading request may be sent to the server 502 in a manner of an http request.

[0164] The page loading request carries information such as a URL address of a webpage that the client 501 requests to display, and meanwhile, the page loading request further carries the identifier of the client 501 and local cache change information recorded in the client 501.

[0165] The identifier of the client 501 may be a digital identifier that is pre-generated for the client 501 by the server 502, and is used as a unique identifier of the client 501 to mark the client 501. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm.

[0166] Certainly, the identifier of the client 501 may also be a unique identifier, where the unique identifier is generated by the client 501 or another third-party server 502 under agreement between the client 501 and the server 502.

[0167] The client 501 locally saves the identifier, and each time when the client 501 interacts with the server 502 subsequently, the client 501 adds the identifier in the request.

[0168] The local cache change information recorded in the client 501 may include that: the client 501 adds or decreases cache data corresponding to one URL.

[0169] The client 501 may maintain a cache recorder, record change information of each cache by the client 501, record the added URL or a hash value corresponding to the URL in an ADD array, and record the deleted URL or a hash value corresponding to the URL in a REMOVE array.

[0170] Recording the hash value corresponding to the URL may decrease the traffic overheads. The involved hash algorithm may be defined according to needs. The following provides one reference hash algorithm:

```
uint32 SuperFastHash(const char * data, int len) {
  uint32_t hash = len, tmp;
  int rem;
  if (len <= 0 || data == NULL)
    return 0;
  rem = len & 3;
  len >>= 2;
  /* Main loop */
  for (; len > 0; len--) {
    hash  += get16bits(data);
    tmp    = (get16bits(data + 2) << 11) ^ hash;
    hash   = (hash << 16) ^ tmp;
    data  += 2 * sizeof(uint16_t);
    hash  += hash >> 11;
  }
  /* Handle end cases */
  switch (rem) {
    case 3:
      hash += get16bits(data);
      hash ^= hash << 16;
      // Treat the final character as signed. This ensures all platforms behave
      // consistently with the original x86 code.
      hash ^= static_cast<signed char>(data[sizeof(uint16_t)]) << 18;
      hash += hash >> 11;
```

```
            break;
        case 2:
            hash += get16bits(data);
            hash ^= hash << 11;
            hash += hash >> 17;
            break;
        case 1:
            hash += static_cast<signed char>(*data);
            hash ^= hash << 10;
            hash += hash >> 1;
    }
    /* Force "avalanching" of final 127 bits */
    hash ^= hash << 3;
    hash += hash >> 5;
    hash ^= hash << 4;
    hash += hash >> 17;
    hash ^= hash << 25;
    hash += hash >> 6;
    return hash;
}.
```

[0171] Using an http request as an example, when the client 501 sends the http request, the client adds fields "GUID" and "Q-Push:" in a header of the http request. The GUID (which is a digital identifier that has a binary length being 128 bits, is generated according to an algorithm, and may be used for marking the client 501) is identifier information delivered by the server 502. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[0172] After receiving the page loading request sent by the client 501, the server 502 acquires corresponding page data according to the URL address carried in the page loading request, uses the page data as home page data returned to the client 501.

[0173] Meanwhile, the server 502 further parses the home page data to obtain the key sub-resource required for loading the home page on the client 501.

[0174] The key sub-resource is a resource playing a critical role on parsing and rendering one HTML, where the resource generally is a JS/CSS of the client 501, or the like.

[0175] The sub-resource is a resource opposite to a main resource of a home page. The main resource refers to one entire HTML page, and the sub-resource refers to files, such as pictures, frames, or scripts, externally cited on the HTML page, and is opposite to the main resource.

[0176] Generally, when returning the home page data to the client, the server 502 pushes, to the client 501 according to the request of the client 501 or actively, the key sub-resource required for loading the home page.

[0177] The server 502 in this embodiment may acquire, through calculation according to the cache change information of the client 501, the key sub-resource that does not exist in the client 501 from the key sub-resource required for loading the home page, and push the acquired key sub-resource to the client 501, so as to save traffic overheads and improve the webpage loading speed on the client 501.

[0178] In this embodiment, it is considered that in the key sub-resources required for loading the home page on the client 501, some key sub-resources may be cached on the client 501; therefore, the server 502 may only push the key sub-resource that does not exist in the client to the client 501, thereby saving traffic overheads and improving the webpage loading speed on the client 501.

[0179] During specific operation, the server 502 first acquires a cache change situation of the client from historical records according to the identifier of the client 501 carried in the page loading request, and obtains, through calculation according to cache change information currently sent by the client 501, a key sub-resource that already exists in the client 501 from the key sub-resource required for loading the home page; therefore, the key sub-resource that does not exist in the client 501 may be acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client 501.

[0180] Further, when returning the home page to the client 501, the server 502 may return the key sub-resource that does not exist in the client 501 to the client 501 in a manner of SPDY

PUSH. The SPDY is the enhanced protocol of the HTTP protocol, where the function of the protocol includes multi-path multiplexing of a data stream, a request priority, and HTTP header compression.

[0181] On the side of the client 501, the client 501 may cache the key sub-resource pushed by the server 502. When displaying the home page, the client 501 may load the key sub-resource required for displaying the home page, where the key sub-resource required for the home page and loaded by the client 501 includes the key sub-resource that already exists in the client 501, and the key sub-resource that does not exist in the client 501 previously and is pushed by the server 502.

[0182] Therefore, compared with the prior art, the client 501 does not need to send the request to the server 502 again to acquire the key sub-resource that is not cached locally, and the server 502 may not push the key sub-resource that already exists in the client 501 to the client 501 repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0183] Further, the client 501 is further configured to update the local cache change information, specifically configured to remove, from the add array or the remove array, a URL pushed by the server or a hash value corresponding to the URL.

[0184] Specifically, after receiving the key sub-resource pushed by the server 502 and completing page loading and displaying, the client 501 removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server 502 or a corresponding hash value, which on one hand can save client cache resources, on the other hand ensures accuracy of the cache change information of the client, thereby further accelerating the page loading speed on the client.

[0185] When accessing a new webpage next, the client 501 repeats the foregoing process.

[0186] The following describes the solution of this embodiment in detail with reference to FIG. 11b.

[0187] FIG. 11b is a schematic diagram of an interaction process between a client 501 and a server 502 in a webpage pushing system according to an embodiment of the present disclosure.

[0188] As shown in FIG. 11b, the interaction process between the client 501 and the server 502 in the webpage pushing system according to the embodiment of the present disclosure is as follows:

[0189] Step 1: The server 502 generates one GUID for the client 501, uses the GUID as a unique identifier of the client 501, and delivers the GUID to the client 501. After the client 501 saves the GUID, each time when the client 501 interacts with the server 502 subsequently, the client 501 adds the field.

[0190] Step 2: The client 501 maintains one cache recorder, and records a change of each cache (CDC, the same below) of the client 501. The change of the CDC mainly refers to adding or decreasing cache data corresponding to one URL. The added URL is recorded in one ADD array, and the deleted URL is recorded in one REMOVE array.

[0191] Step 3: Send a page request to the server 502.The header of the HTTP request carries the fields "GUID" and "Q-PUSH:". The GUID is GUID information delivered by the server 502. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[0192] Step 3 can ensure that the server 502 knows the change of CDC, that is, can acquire a state of a current cache of the client 501.

[0193] Step 4: The server 502 records CDC change of each client 501 in a cache (SDC) of the server 502.

[0194] Step 5: The server 502 returns a home page to the client 501, and searches for CDC of a corresponding client 501 from the SDC through the GUID, to obtain a key sub-resource (JS/CSS) that already exists in the client 501. The key sub-resource that does not exist in the client 501 is returned to the client 501 in a manner of SPDY PUSH.

[0195] Step 6: After receiving the data, the client 501 updates the CDC, records the change of the CDC in the ADD and REMOVE arrays, and removes, from the ADD and EREMOVE arrays, a URL previously sent through Q-Push or a corresponding hash value.

[0196] Steps 5 and 6 can be ensured that the server 502 does not transmit the key sub-resource to the client 501 repeatedly, which not only accelerates the transmission speed of the key sub-resource JS/CSS (can accelerate the page loading), but also does not lead to the traffic problem caused by retransmission.

[0197] When accessing the new page next, the client 501 repeats steps 2 to 6.

[0198] Therefore, by using the foregoing solution, the client 501 does not need to send the request to the server 502 again to acquire the key sub-resource that is not cached locally, and the server 502 may not push the key sub-resource that already exists in the client 501 to the client 501

repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0199] As shown in FIG. 12, a first embodiment of the present disclosure provides a webpage pushing server, including: a request receiving module 401, a parsing module 402, and a pushing module 403, where

the request receiving module 401 is configured to receive a page loading request sent by a client, where the page loading request carries an identifier of the client and cache change information of the client;

the parsing module 402 is configured to acquire a home page according to the page loading request, and parse the home page to acquire a key sub-resource required for loading the home page; and

the pushing module 403 is configured to return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client.

[0200] The solution in this embodiment involves the client and the server, where the client may be a PC, or a mobile client capable of displaying a webpage, such as a mobile phone or a tablet computer. The client may display different webpages according to a website or webpage clicking operation instruction input by a user, where webpage data is pulled by the client by sending a request to the server, or is actively pushed to the client by the server.

[0201] To improve a webpage loading speed on the client, generally, the server may return multiple resources for one request of the client. In this way, the server may push, to the client in advance, the key sub-resource required for displaying the page, so that the page loading speed increases. In the existing pushing solution, the server cannot determine in advance whether the client already has cache of the key sub-resource; therefore, the server pushes a key sub-resource that is already cached in the client, thereby increasing network traffic overheads on the client, occupying a bandwidth, and reducing the page loading speed on the client.

[0202] In this embodiment, through the Q-Push protocol, when pushing resources to the client, the server may ensure, according to the cache change information recorded in the client, that a key sub-resource pushed by the server is the key sub-resource that does not exist in the client, so as to achieve an effect of improving a pushing speed and reducing traffic occupation on the client.

[0203] Specifically, at first, when a user needs to display a webpage, the client sends the page loading request to the server, where the page loading request may be sent to the server in a manner of an http request.

[0204] The page loading request carries information such as a URL address of a webpage that the client requests to display, and meanwhile, the page loading request further carries the identifier of the client and local cache change information recorded in the client.

[0205] The identifier of the client may be a digital identifier that is pre-generated for the client by the server, and is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm.

[0206] Certainly, the identifier of the client may also be a unique identifier, where the unique identifier is generated by the client or another third-party server under agreement between the client and the server.

[0207] The client locally saves the identifier, and each time when the client interacts with the server subsequently, the client adds the identifier in the request.

[0208] The local cache change information recorded in the client may include that: the client adds or decreases cache data corresponding to one URL.

[0209] The client may maintain a cache recorder, record change information of each cache by the client, record the added URL or a hash value corresponding to the URL in an ADD array, and record the deleted URL or a hash value corresponding to the URL in a REMOVE array.

[0210] Using an http request as an example, when the client sends the http request, the client adds fields "GUID" and "Q-Push:" in a header of the http request. The GUID (which is a digital identifier that has a binary length being 128 bits, is generated according to an algorithm, and may be used for marking the client) is identifier information delivered by the server. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[0211] After receiving the page loading request sent by the client, the server acquires corresponding page data according to the URL address carried in the page loading request, uses the page data as home page data returned to the client.

[0212] Meanwhile, the server further parses the home page data to obtain the key sub-resource required for loading the home page on the client.

[0213] The key sub-resource is a resource playing a critical role on parsing and rendering one HTML, where the resource generally is a JS/CSS of the client, or the like.

[0214] The sub-resource is a resource opposite to a main resource of a home page. The main resource refers to one entire HTML page, and the sub-resource refers to files, such as pictures, frames, or scripts, externally cited on the HTML page, and is opposite to the main resource.

[0215] Generally, when returning the home page data to the client, the server pushes, to the client according to the request of the client or actively, the key sub-resource required for loading the home page.

[0216] The server in this embodiment may acquire, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, so as to save traffic overheads and improve the webpage loading speed on the client.

[0217] In this embodiment, it is considered that in the key sub-resources required for loading the home page on the client, some key sub-resources may be cached on the client; therefore, the server may only push the key sub-resource that does not exist in the client to the client, thereby saving traffic overheads and improving the webpage loading speed on the client.

[0218] During specific operation, the server first acquires a cache change situation of the client from historical records according to the identifier of the client carried in the page loading request, and obtains, through calculation according to cache change information currently sent by the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page; therefore, the key sub-resource that does not exist in the client may be acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client.

[0219] Further, when returning the home page to the client, the server may return the key sub-resource that does not exist in the client to the client in a manner of SPDY PUSH. The SPDY is the enhanced protocol of the HTTP protocol, where the function of the protocol includes multi-path multiplexing of a data stream, a request priority, and HTTP header compression.

[0220] On the side of the client, the client may cache the key sub-resource pushed by the server. When displaying the home page, the client may load the key sub-resource required for displaying the home page, where the key sub-resource required for the home page and loaded by the client includes the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0221] Therefore, compared with the prior art, the client does not need to send the request to the server again to acquire the key sub-resource that is not cached locally, and the server may not push the key sub-resource that already exists in the client to the client repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0222] As shown in FIG. 13, a second embodiment of the present disclosure provides a webpage pushing server. Based on the first embodiment shown in FIG. 12, the server further includes:

a storage module 404, configured to record the cache change information of the client in a cache of the server in correspondence to the identifier of the client.

[0223] A difference between this embodiment and the first embodiment shown in FIG. 12 lies in that, this embodiment further includes a solution in which the server records and caches the acquired cache change information of the client.

[0224] Specifically, considering that the server receives page loading requests of different clients, to improve page pushing efficiency, in this embodiment, after receiving the page loading request sent by the client, the server extracts the cache change information of the client from the page loading request, and records the cache change information of the client in the cache of the server in correspondence to the identifier of the client.

[0225] During subsequent specific operation, the server first searches for the cache change information of the client from the cache of the server according to the identifier of the client carried in the page loading request, and then acquires, through calculation according to the cache change information of the client, the key sub-resource that already exists from the key sub-resource required for loading the home page.

[0226] Then, the key sub-resource that does not exist in the client is acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client is pushed to the client.

[0227] More specifically, as shown in FIG. 14, the pushing module 403 includes a searching unit 4031, a calculation and acquisition unit 4032, and a pushing unit 4033, where

the searching unit 4031 is configured to search for the cache change information of the client from the cache of the server according to the identifier of the client;

the calculation and acquisition unit 4032 is configured to acquire, through calculation according to the cache change information of the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page, and acquire, according to the key sub-resource that already exists in the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page; and

the pushing unit 4033 is configured to push the key sub-resource that does not exist in the client to the client.

[0228] In this embodiment, by using the foregoing embodiment, when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency.

[0229] As shown in FIG. 15, a third embodiment of the present disclosure provides a webpage pushing server. Based on the second embodiment shown in FIG. 13, the server further includes:

a generation and sending module 400, configured to generate the identifier for the client, and deliver the identifier to the client.

[0230] A difference between this embodiment and the second embodiment shown in FIG. 13 lies in that, this embodiment further includes a solution in which the server generates the identifier for the client.

[0231] Specifically, the server generates a digital identifier for the client, where the digital identifier is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm. After generating the identifier for the client, the server delivers the identifier to the client. By using the foregoing solution, flexibility of interaction

between the client and the server is increased, so that the server pushes corresponding network resources to the client according to different client identifiers.

[0232] As shown in FIG. 16, a first embodiment of the present disclosure provides a webpage pushing client, including: a sending module 601, a receiving module 602, and a loading and display module 603, where

the sending module 601 is configured to send a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client;

the receiving module 602 is configured to receive a home page returned by the server and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, where the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client; and

the loading and display module 603 is configured to display the home page, and load the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key sub-resource that exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0233] The solution in this embodiment involves the client and the server, where the client may be a PC, or a mobile client capable of displaying a webpage, such as a mobile phone or a tablet computer. The client may display different webpages according to a website or webpage clicking operation instruction input by a user, where webpage data is pulled by the client by sending a request to the server, or is actively pushed to the client by the server.

[0234] To improve a webpage loading speed on the client, generally, the server may return multiple resources for one request of the client. In this way, the server may push, to the client in advance, the key sub-resource required for displaying the page, so that the page loading speed increases. In the existing pushing solution, the server cannot determine in advance whether the client already has cache of the key sub-resource; therefore, the server pushes a key sub-resource that is already cached in the client, thereby increasing network traffic overheads on the client, occupying a bandwidth, and reducing the page loading speed on the client.

[0235] In this embodiment, through the Q-Push protocol, when pushing resources to the client, the server may ensure, according to the cache change information recorded in the client, that

a key sub-resource pushed by the server is the key sub-resource that does not exist in the client, so as to achieve an effect of improving a pushing speed and reducing traffic occupation on the client.

[0236] Specifically, at first, when a user needs to display a webpage, the client sends the page loading request to the server, where the page loading request may be sent to the server in a manner of an http request.

[0237] The page loading request carries information such as a URL address of a webpage that the client requests to display, and meanwhile, the page loading request further carries the identifier of the client and local cache change information recorded in the client.

[0238] The identifier of the client may be a digital identifier that is pre-generated for the client by the server, and is used as a unique identifier of the client to mark the client. As one implementation manner, the digital identifier may be a digital identifier that has a binary length being 128 bits and is generated according to a predefined algorithm.

[0239] Certainly, the identifier of the client may also be a unique identifier, where the unique identifier is generated by the client or another third-party server under agreement between the client and the server.

[0240] The client locally saves the identifier, and each time when the client interacts with the server subsequently, the client adds the identifier in the request.

[0241] The local cache change information recorded in the client may include that: the client adds or decreases cache data corresponding to one URL.

[0242] The client may maintain a cache recorder, record change information of each cache by the client, record the added URL or a hash value corresponding to the URL in an ADD array, and record the deleted URL or a hash value corresponding to the URL in a REMOVE array.

[0243] Using an http request as an example, when the client sends the http request, the client adds fields "GUID" and "Q-Push:" in a header of the http request. The GUID (which is a digital identifier that has a binary length being 128 bits, is generated according to an algorithm, and may be used for marking the client) is identifier information delivered by the server. The Q-Push records URLs corresponding to the ADD and REMOVE arrays, or hash values corresponding to the URLs.

[0244] After receiving the page loading request sent by the client, the server acquires corresponding page data according to the URL address carried in the page loading request, uses the page data as home page data returned to the client.

[0245] Meanwhile, the server further parses the home page data to obtain the key sub-resource required for loading the home page on the client.

[0246] The key sub-resource is a resource playing a critical role on parsing and rendering one HTML, where the resource generally is a JS/CSS of the client, or the like.

[0247] The sub-resource is a resource opposite to a main resource of a home page. The main resource refers to one entire HTML page, and the sub-resource refers to files, such as pictures, frames, or scripts, externally cited on the HTML page, and is opposite to the main resource.

[0248] Generally, when returning the home page data to the client, the server pushes, to the client according to the request of the client or actively, the key sub-resource required for loading the home page.

[0249] The server in this embodiment may acquire, through calculation according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, so as to save traffic overheads and improve the webpage loading speed on the client.

[0250] In this embodiment, it is considered that in the key sub-resources required for loading the home page on the client, some key sub-resources may be cached on the client; therefore, the server may only push the key sub-resource that does not exist in the client to the client, thereby saving traffic overheads and improving the webpage loading speed on the client.

[0251] During specific operation, the server first acquires a cache change situation of the client from historical records according to the identifier of the client carried in the page loading request, and obtains, through calculation according to cache change information currently sent by the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page; therefore, the key sub-resource that does not exist in the client may be acquired from the key sub-resource required for loading the home page and according to the key sub-resource that already exists in the client.

[0252] Further, when returning the home page to the client, the server may return the key sub-resource that does not exist in the client to the client in a manner of SPDY PUSH. The SPDY is the enhanced protocol of the HTTP protocol, where the function of the protocol includes multi-path multiplexing of a data stream, a request priority, and HTTP header compression.

[0253] On the side of the client, the client may cache the key sub-resource pushed by the server. When displaying the home page, the client may load the key sub-resource required for displaying the home page, where the key sub-resource required for the home page and loaded by the

client includes the key sub-resource that already exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

[0254] Therefore, compared with the prior art, the client does not need to send the request to the server again to acquire the key sub-resource that is not cached locally, and the server may not push the key sub-resource that already exists in the client to the client repeatedly, which accelerates a speed of transmitting the key sub-resource JS/CSS, does not cause the traffic problem caused by the retransmission, and improves the webpage loading speed on the client.

[0255] As shown in FIG. 17, a second embodiment of the present disclosure provides a webpage pushing client. Based on the first embodiment shown in FIG. 16, the client further includes:

a recording module 600, configured to record local cache change information, where the local cache change information includes that: the client adds or decreases cache data corresponding to one URL.

[0256] A difference between this embodiment and the first embodiment shown in FIG. 16 lies in that: this embodiment further includes a solution in which the client records the local cache change information.

[0257] Specifically, the local cache change information includes that: the client adds or decreases cache data corresponding to one URL.

[0258] The process of recording, by the client, the local cache change information specifically includes:

recording, in an add array, the URL added by the client or a hash value corresponding to the URL, and recording, in a remove array, the URL deleted by the client or a hash value corresponding to the URL.

[0259] Recording the hash value corresponding to the URL may decrease the traffic overheads. The involved hash algorithm may be defined according to needs. The following provides one reference hash algorithm:

```
uint32 SuperFastHash(const char * data, int len) {
  uint32_t hash = len, tmp;
  int rem;
  if (len <= 0 || data == NULL)
```

```
  return 0;
rem = len & 3;
len >>= 2;
/* Main loop */
for (; len > 0; len--) {
  hash  += get16bits(data);
  tmp    = (get16bits(data + 2) << 11) ^ hash;
  hash   = (hash << 16) ^ tmp;
  data  += 2 * sizeof(uint16_t);
  hash  += hash >> 11;
}
/* Handle end cases */
switch (rem) {
  case 3:
    hash += get16bits(data);
    hash ^= hash << 16;
    // Treat the final character as signed. This ensures all platforms behave
    // consistently with the original x86 code.
    hash ^= static_cast<signed char>(data[sizeof(uint16_t)]) << 18;
    hash += hash >> 11;
    break;
  case 2:
    hash += get16bits(data);
    hash ^= hash << 11;
    hash += hash >> 17;
    break;
```

```
            case 1:
              hash += static_cast<signed char>(*data);
              hash ^= hash << 10;
              hash += hash >> 1;
            }
            /* Force "avalanching" of final 127 bits */
            hash ^= hash << 3;
            hash += hash >> 5;
            hash ^= hash << 4;
            hash += hash >> 17;
            hash ^= hash << 25;
            hash += hash >> 6;
            return hash;
          }.
```

[0260] In this embodiment, by using the foregoing solution, the client records the local cache change information; when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency.

[0261] As shown in FIG. 18, a third embodiment of the present disclosure provides a webpage pushing client. Based on the second embodiment shown in FIG. 17, the client further includes:

an updating module 604, configured to update the local cache change information, specifically configured to remove, from the add array or the remove array, a URL pushed by the server or a hash value corresponding to the URL.

[0262] A difference between this embodiment and the second embodiment shown in FIG. 17 lies in that: this embodiment further includes a solution in which the client updates the local cache change information.

[0263] After receiving the key sub-resource pushed by the server and completing page loading and displaying, the client removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server or a corresponding hash value, which on one hand can save client cache resources, on the other hand ensures accuracy of the cache change information of the client, thereby further accelerating the page loading speed on the client.

[0264] When accessing a new webpage next, the client repeats the foregoing process.

[0265] In this embodiment, by using the foregoing solution, the server generates the identifier for the client, and delivers the identifier to the client; the client records the local cache change information, when requesting to load a page, the client sends the local cache change information to the server, so that the server acquires a cache record that already exists in the client; when returning the home page to the client, the server obtains, according to the cache change information of the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client, thereby avoiding repeated resource pushing, accelerating the page loading speed on the client, and avoiding unnecessary traffic overheads. In addition, after acquiring the cache change information of the client, the server records the cache change information of the client in the cache of the server in correspondence to the identifier of the client, so as to facilitate calculation and improve calculation efficiency and page data pushing efficiency. After receiving the key sub-resource pushed by the server and completes page loading and displaying, the client removes, from the ADD and REMOVE arrays, a URL previously sent through the Q-Push by the server or a corresponding hash value, which on one hands can save cache resources of the client, on the other hand ensures accuracy of the cache change information of the client, and further accelerates the page loading speed on the client.

[0266] It should be noted that, in this specification, the terms "include", "comprise", and any variants thereof are intended to cover a non-exclusive inclusion. Therefore, in the context of a process, method, object, or device that includes a series of elements, the process, method, object, or

device not only includes such elements, but also includes other elements not specified expressly, or may include inherent elements of the process, method, object, or device. Unless otherwise specified, an element limited by "include a/an..." does not exclude other same elements existing in the process, the method, the article, or the device that includes the element.

[0267] The sequence numbers of the foregoing embodiments of the present disclosure are merely for the convenience of description, and do not imply the preference among the embodiments.

[0268] Through the descriptions of the foregoing implementation manners, a person skilled in the art may clearly know that the foregoing method embodiments may be implemented by means of software plus a general hardware platform, and certainly, may also be implemented by means of hardware. However, in many cases, the former is the preferred implementation manner. Based on such an understanding, Based on such an understanding, the technical solutions of the present disclosure essentially, or the part contributing to the prior art may be implemented in a form of a software product. The computer software product is stored in a storage medium (such as a ROM/RAM, a magnetic disk, or an optical disc) and includes several instructions for instructing a computer device (which may be a personal computer, a server, a network device, or the like) to perform the methods described in the embodiments of the present disclosure.

[0269] The foregoing descriptions are merely preferred embodiments of the present disclosure but are not intended to limit the patent scope of the present disclosure. Any equivalent modifications made to the structures or processes based on the content of the specification and the accompanying drawings of the present disclosure for direct or indirect use in other relevant technical fields shall also be encompassed in the patent protection scope of the present disclosure.

CLAIMS

What is claimed is:

1. A webpage pushing method, comprising:

receiving, by a server from a client, a page loading request, wherein the page loading request carries an identifier of the client and cache change information of the client;

acquiring, by the server, a home page according to the page loading request, and parsing the home page to acquire a key sub-resource required for loading the home page; and

returning, by the server, the home page to the client, acquiring, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client.

2. The method according to claim 1, wherein after the step of sending, by a client, a page loading request to a server, the method further comprises:

recording, by the server, the cache change information of the client in a cache of the server in correspondence to the identifier of the client.

3. The method according to claim 2, wherein the step of acquiring, by the server through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client comprises:

searching for, by the server, the cache change information of the client from the cache of the server according to the identifier of the client;

acquiring, through calculation according to the cache change information of the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page;

acquiring, according to the key sub-resource that already exists in the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page; and

pushing the key sub-resource that does not exist in the client to the client.

4. The method according to claim 1, wherein before the step of sending, by a client, a page loading request to a server, the method further comprises:

recording, by the client, local cache change information, wherein the local cache change information comprises that: the client adds or decreases cache data corresponding to one URL.

5. The method according to claim 4, wherein the step of recording, by the client, local cache change information comprises:

recording, in an add array, the URL added by the client or a hash value corresponding to the URL, and recording, in a remove array, the URL deleted by the client or a hash value corresponding to the URL.

6. The method according to claim 5, further comprising:

updating, by the client, the local cache change information, specifically comprising:

removing, by the client from the add array or the remove array, a URL pushed by the server or a hash value corresponding to the URL.

7. The method according to claim 1, wherein before the step of sending, by a client, a page loading request to a server, the method further comprises:

generating, by the server, the identifier for the client, and delivering the identifier to the client.

8. A webpage pushing method, comprising:

receiving, by a server, a page loading request sent by a client, wherein the page loading request carries an identifier of the client and cache change information of the client;

acquiring a home page according to the page loading request, and parsing the home page to acquire a key sub-resource required for loading the home page; and

returning the home page to the client, acquiring, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the key sub-resource that does not exist in the client to the client.

9. The method according to claim 8, wherein after the step of receiving, by a server, a page loading request sent by a client, the method further comprises:

recording, by the server, the cache change information of the client in a cache of the server in correspondence to the identifier of the client.

10. The method according to claim 9, wherein the step of acquiring, by the server through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushing the

key sub-resource that does not exist in the client to the client comprises:

searching for, by the server, the cache change information of the client from the cache of the server according to the identifier of the client;

acquiring, through calculation according to the cache change information of the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page;

acquiring, according to the key sub-resource that already exists in the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page; and

pushing the key sub-resource that does not exist in the client to the client.

11. The method according to claim 8, 9, or 10, wherein before the step of receiving, by a server, a page loading request sent by a client, the method further comprises:

generating, by the server, the identifier for the client, and delivering the identifier to the client.

12. A webpage pushing method, comprising:

sending, by a client, a page loading request to a server, wherein the page loading request carries an identifier of the client and cache change information of the client;

receiving a home page returned by the server and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, wherein the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client; and

displaying the home page, and loading the key sub-resource required by the home page in the home page, wherein the key sub-resource required by the home page comprises a key sub-resource that exists in the client, and the key sub-resource that does not exist in the client previously and is pushed by the server.

13. The method according to claim 12, wherein before the step of sending, by a client, a page loading request to a server, the method further comprises:

recording, by the client, local cache change information, wherein the local cache change information comprises that: the client adds or decreases cache data corresponding to one URL.

14. The method according to claim 13, wherein the step of recording, by the client, local cache change information comprises:

recording, in an add array, the URL added by the client or a hash value corresponding to the URL, and recording, in a remove array, the URL deleted by the client or a hash value corresponding to the URL.

15. The method according to claim 12, further comprising:

updating, by the client, the local cache change information, specifically comprising:

removing, by the client from the add array or the remove array, a URL pushed by the server or a hash value corresponding to the URL.

16. The method according to claim 15, wherein before the step of sending, by a client, a page loading request to a server, the method further comprises:

receiving, by the client, the identifier generated for the client and delivered by the server.

17. A webpage pushing server, comprising:

one or more processors;

memory storing a plurality of modules consisting of instructions configured for executing by one or more processors, the plurality of modules comprising:

a request receiving module, configured to receive a page loading request sent by a client, wherein the page loading request carries an identifier of the client and cache change information of the client;

a parsing module, configured to acquire a home page according to the page loading request, and parse the home page to acquire a key sub-resource required for loading the home page; and

a pushing module, configured to return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client.

18. The server according to claim 17, further comprising:

a storage module, configured to record the cache change information of the client in a cache of the server in correspondence to the identifier of the client; and

a generation and sending module, configured to generate the identifier for the client, and deliver the identifier to the client.

19. The server according to claim 18, wherein the pushing module comprises:

a searching unit, configured to search for the cache change information of the client from the cache of the server according to the identifier of the client;

a calculation and acquisition unit, configured to acquire, through calculation according to the cache change information of the client, a key sub-resource that already exists in the client from the key sub-resource required for loading the home page, and acquire, according to the key sub-resource that already exists in the client, the key sub-resource that does not exist in the client from the key sub-resource required for loading the home page; and

a pushing unit, configured to push the key sub-resource that does not exist in the client to the client.

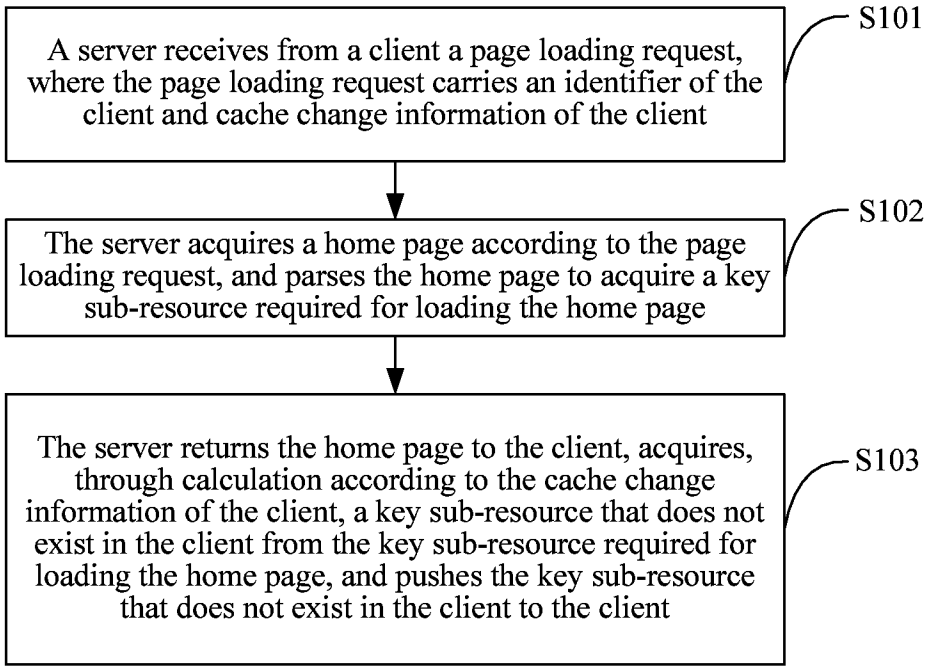
A server receives from a client a page loading request, where the page loading request carries an identifier of the client and cache change information of the client
S101
The server acquires a home page according to the page loading request, and parses the home page to acquire a key sub-resource required for loading the home page
S102
The server returns the home page to the client, acquires, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client
S103

FIG. 1

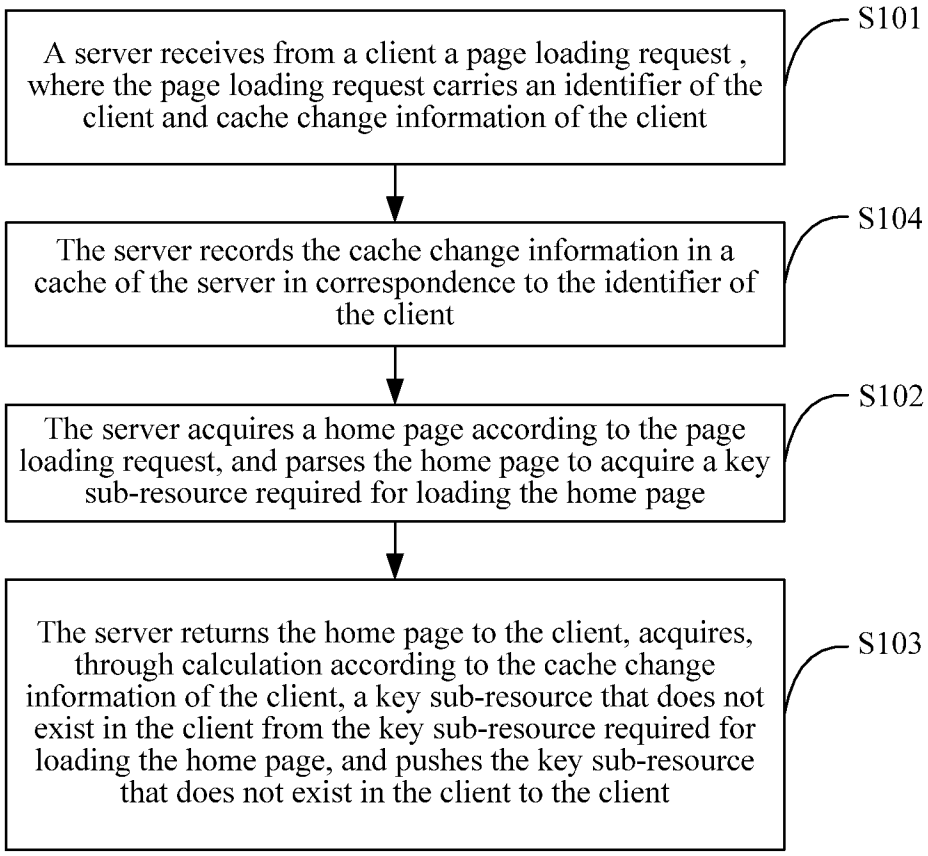
A server receives from a client a page loading request , where the page loading request carries an identifier of the client and cache change information of the client
S101
The server records the cache change information in a cache of the server in correspondence to the identifier of the client
S104
The server acquires a home page according to the page loading request, and parses the home page to acquire a key sub-resource required for loading the home page
S102
The server returns the home page to the client, acquires, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client
S103

FIG. 2

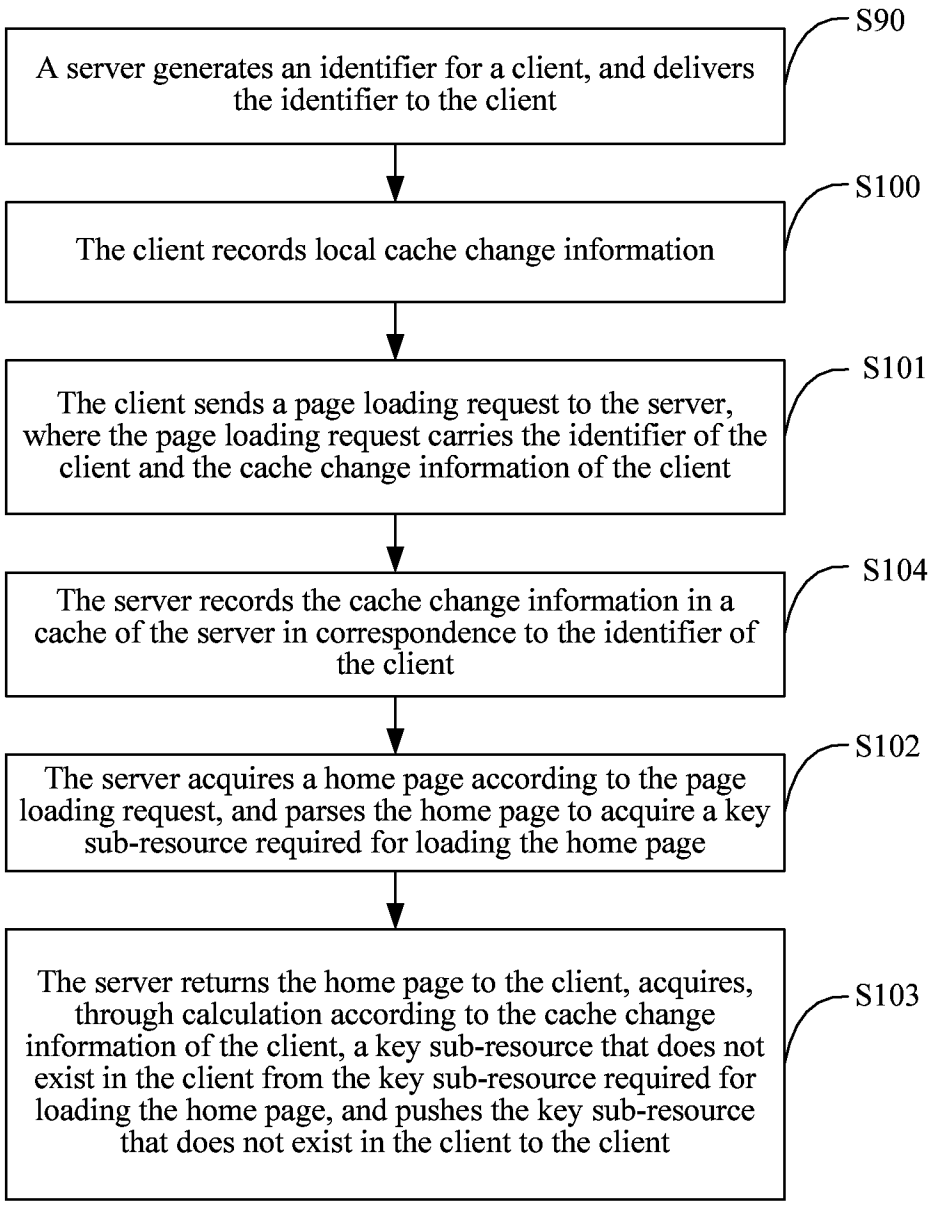
A server generates an identifier for a client, and delivers the identifier to the client
S90
The client records local cache change information
S100
The client sends a page loading request to the server, where the page loading request carries the identifier of the client and the cache change information of the client
S101
The server records the cache change information in a cache of the server in correspondence to the identifier of the client
S104
The server acquires a home page according to the page loading request, and parses the home page to acquire a key sub-resource required for loading the home page
S102
The server returns the home page to the client, acquires, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client
S103

FIG. 3

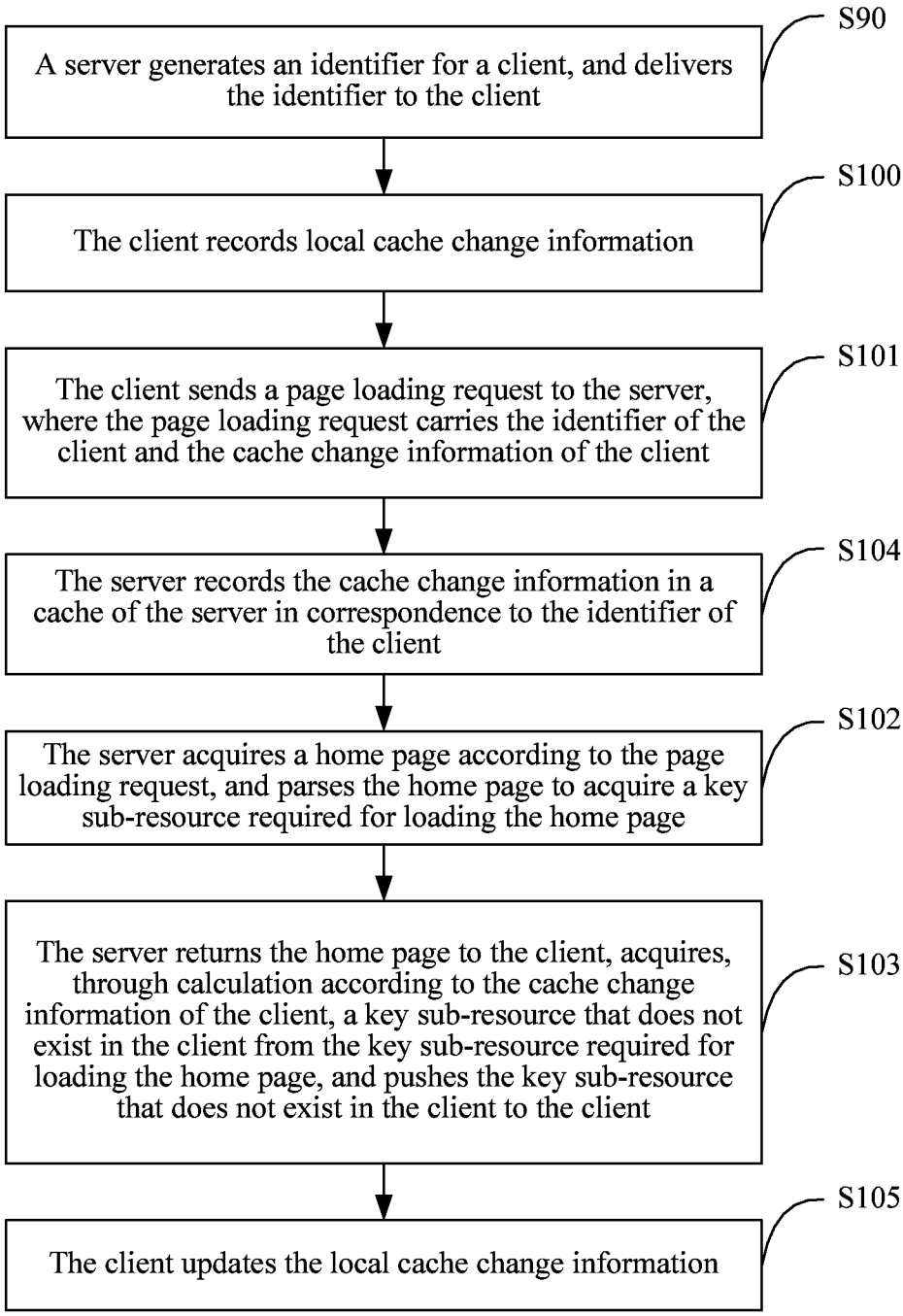
A server generates an identifier for a client, and delivers the identifier to the client
S90
The client records local cache change information
S100
The client sends a page loading request to the server, where the page loading request carries the identifier of the client and the cache change information of the client
S101
The server records the cache change information in a cache of the server in correspondence to the identifier of the client
S104
The server acquires a home page according to the page loading request, and parses the home page to acquire a key sub-resource required for loading the home page
S102
The server returns the home page to the client, acquires, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and pushes the key sub-resource that does not exist in the client to the client
S103
The client updates the local cache change information
S105

FIG. 4

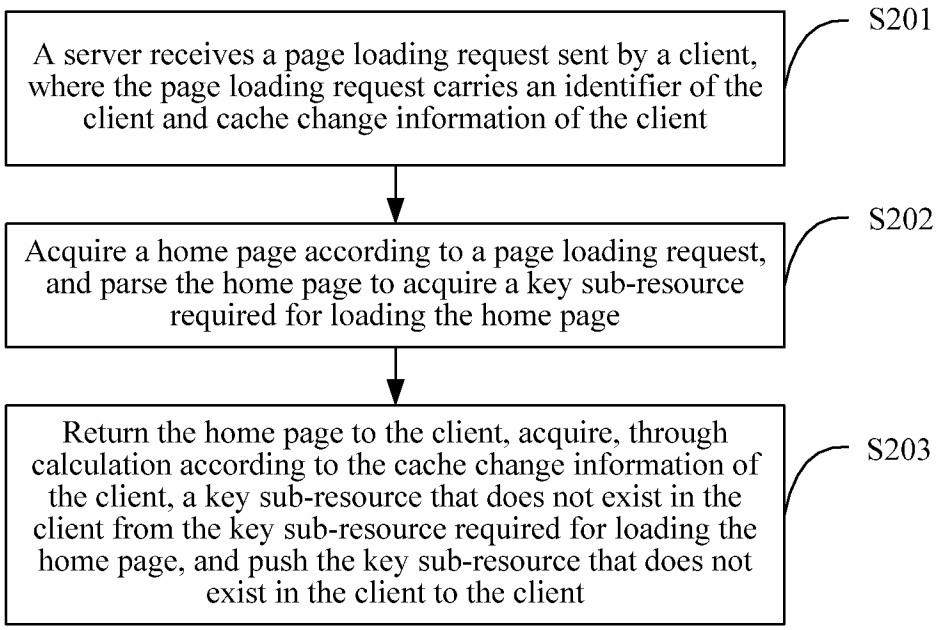
A server receives a page loading request sent by a client, where the page loading request carries an identifier of the client and cache change information of the client
S201
Acquire a home page according to a page loading request, and parse the home page to acquire a key sub-resource required for loading the home page
S202
Return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client
S203

FIG. 5

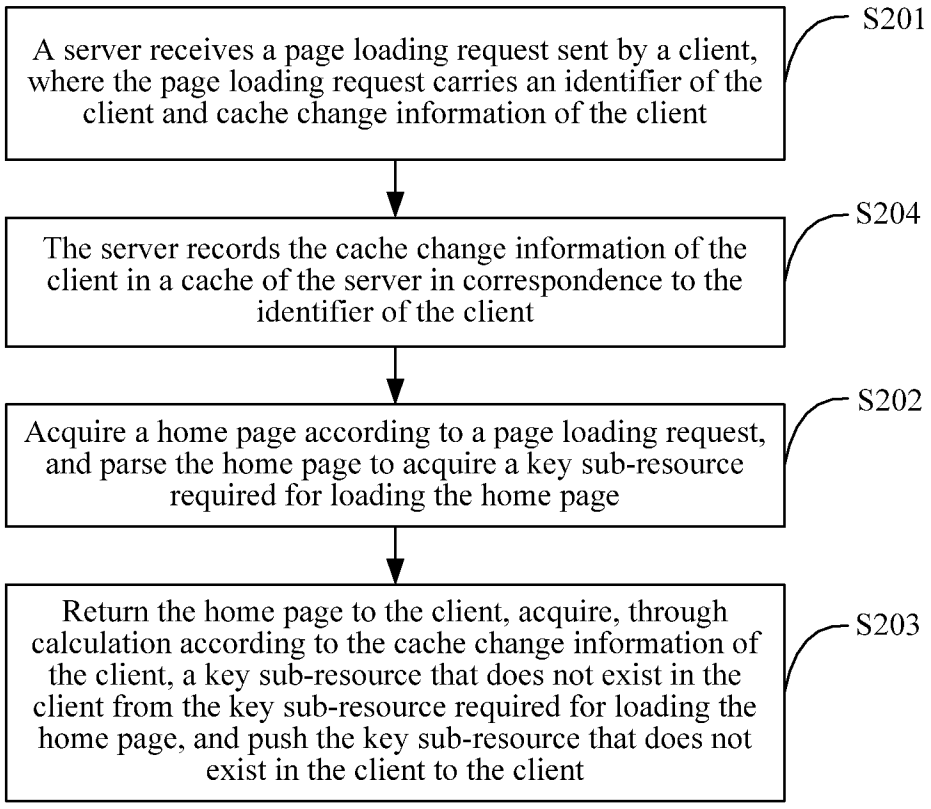
A server receives a page loading request sent by a client, where the page loading request carries an identifier of the client and cache change information of the client
S201
The server records the cache change information of the client in a cache of the server in correspondence to the identifier of the client
S204
Acquire a home page according to a page loading request, and parse the home page to acquire a key sub-resource required for loading the home page
S202
Return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client
S203

FIG. 6

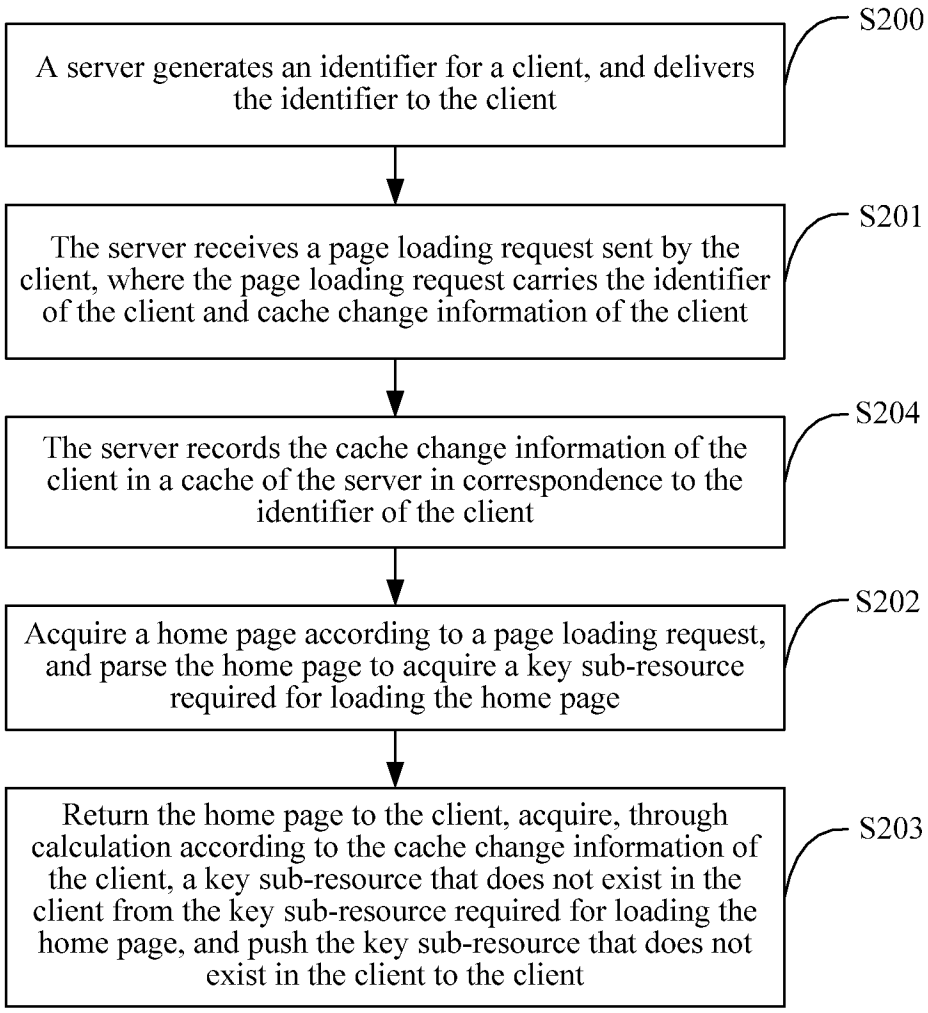
A server generates an identifier for a client, and delivers the identifier to the client
S200
The server receives a page loading request sent by the client, where the page loading request carries the identifier of the client and cache change information of the client
S201
The server records the cache change information of the client in a cache of the server in correspondence to the identifier of the client
S204
Acquire a home page according to a page loading request, and parse the home page to acquire a key sub-resource required for loading the home page
S202
Return the home page to the client, acquire, through calculation according to the cache change information of the client, a key sub-resource that does not exist in the client from the key sub-resource required for loading the home page, and push the key sub-resource that does not exist in the client to the client
S203

FIG. 7

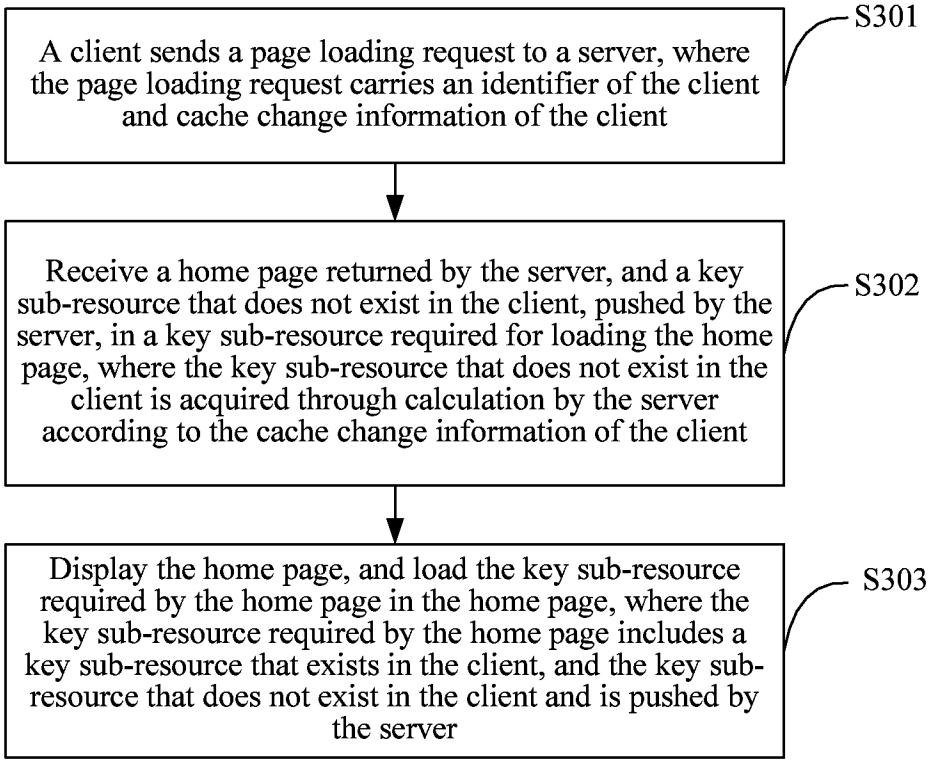
A client sends a page loading request to a server, where the page loading request carries an identifier of the client and cache change information of the client
S301
Receive a home page returned by the server, and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, where the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client
S302
Display the home page, and load the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key sub-resource that exists in the client, and the key sub-resource that does not exist in the client and is pushed by the server
S303

FIG. 8

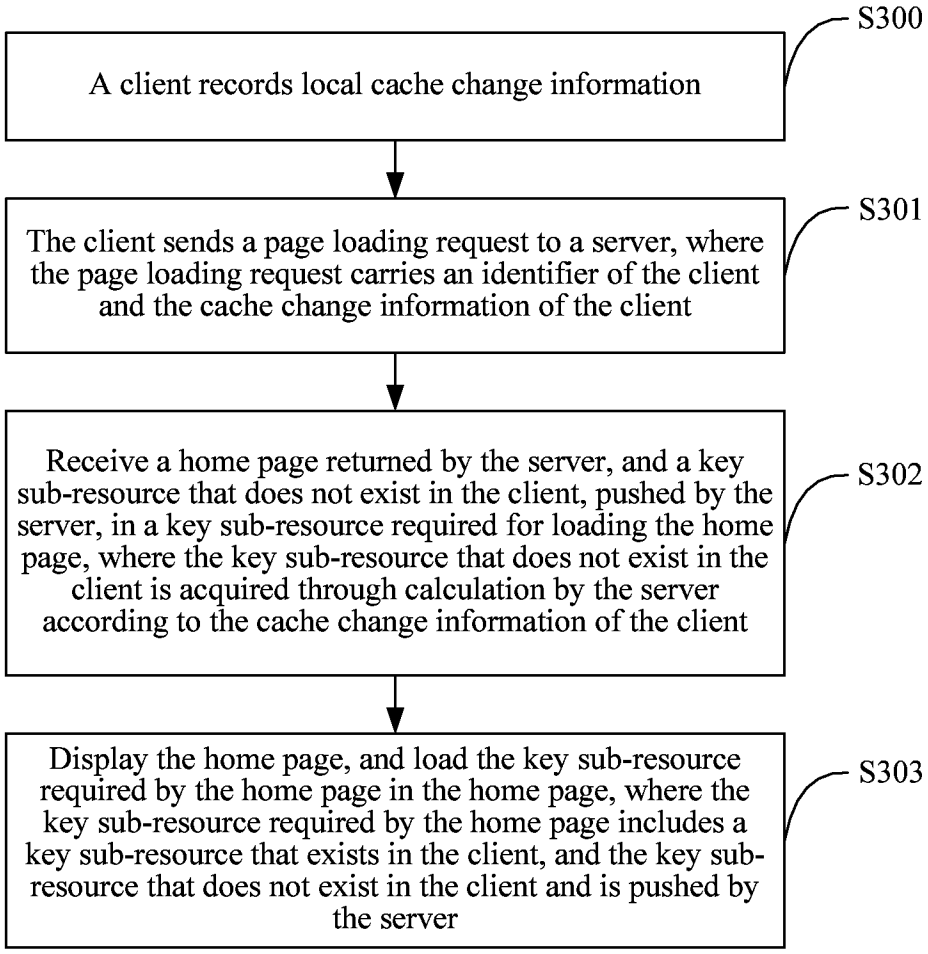
S300
A client records local cache change information
S301
The client sends a page loading request to a server, where the page loading request carries an identifier of the client and the cache change information of the client
S302
Receive a home page returned by the server, and a key sub-resource that does not exist in the client, pushed by the server, in a key sub-resource required for loading the home page, where the key sub-resource that does not exist in the client is acquired through calculation by the server according to the cache change information of the client
S303
Display the home page, and load the key sub-resource required by the home page in the home page, where the key sub-resource required by the home page includes a key sub-resource that exists in the client, and the key sub-resource that does not exist in the client and is pushed by the server

FIG. 9

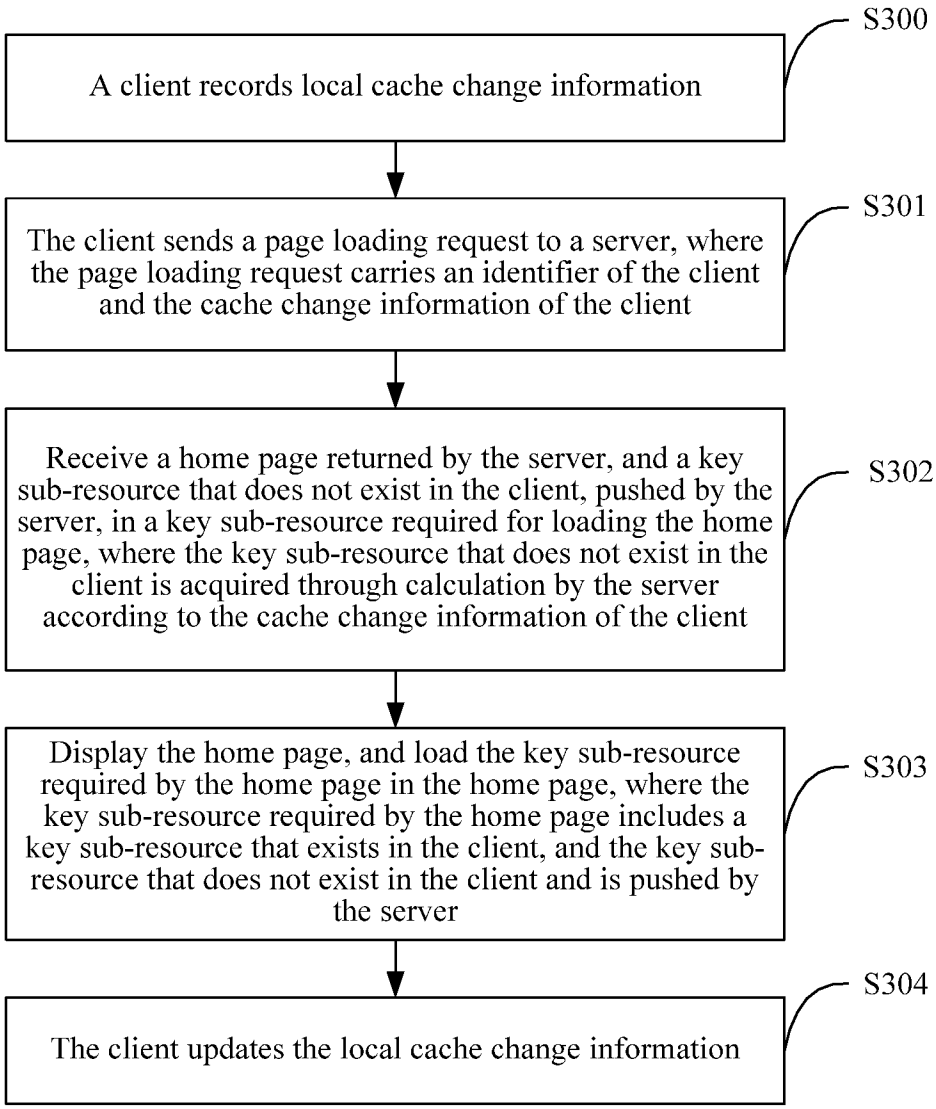

FIG. 10

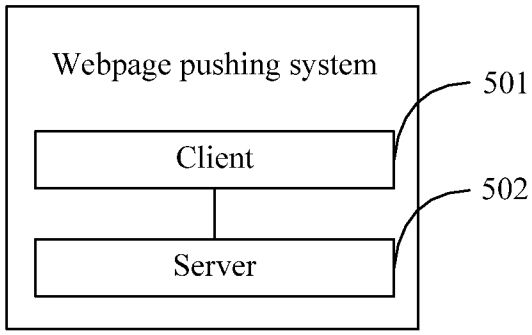

FIG. 11a

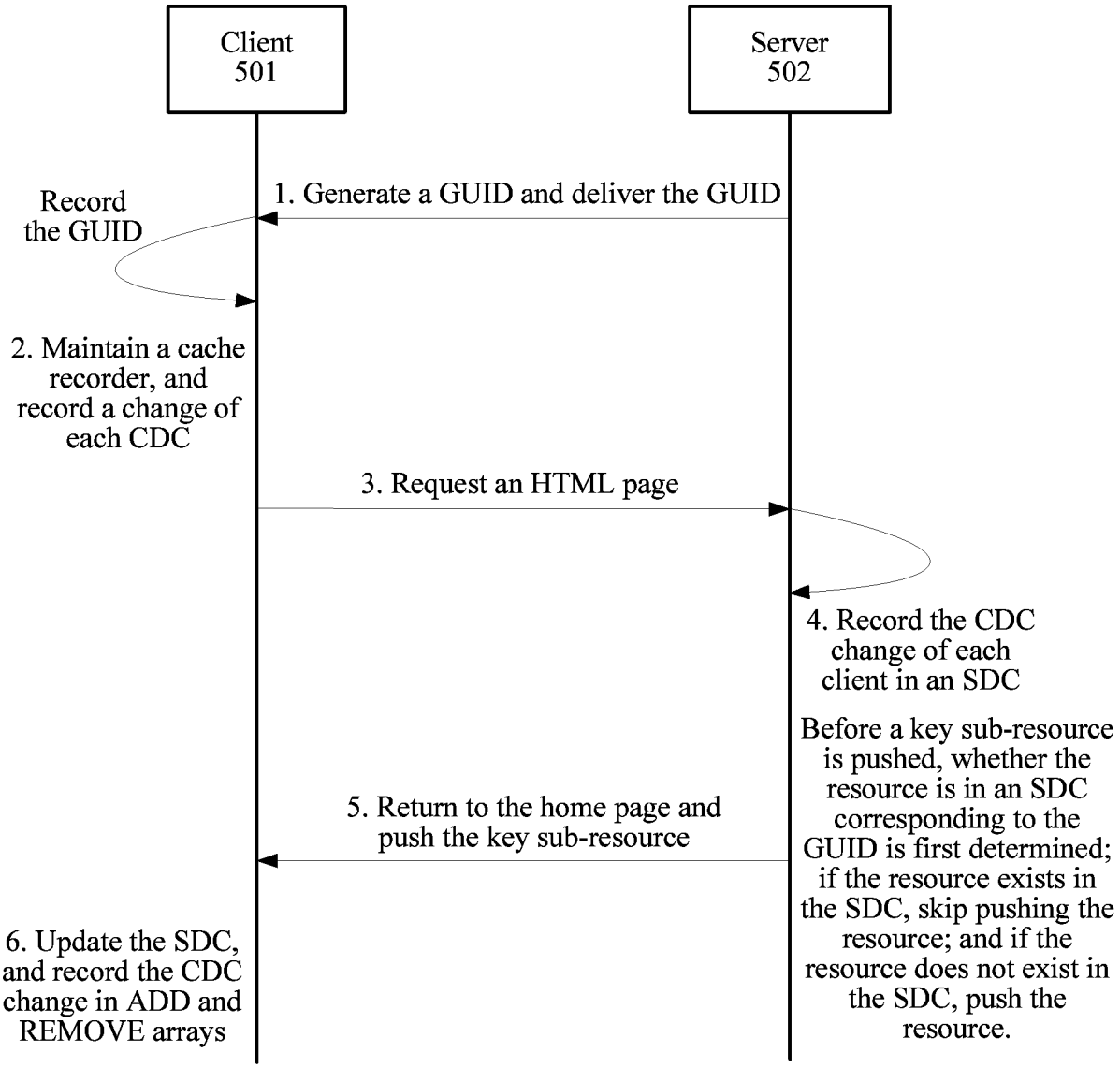

FIG. 11b

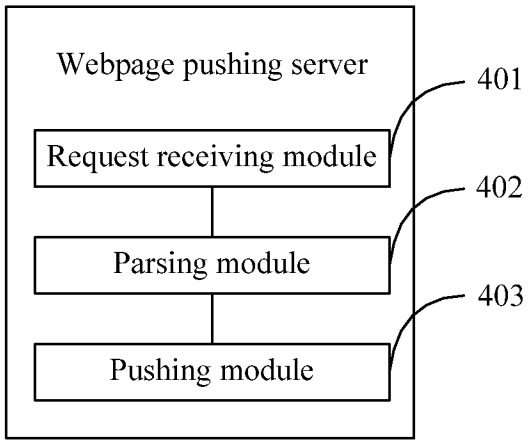

FIG. 12

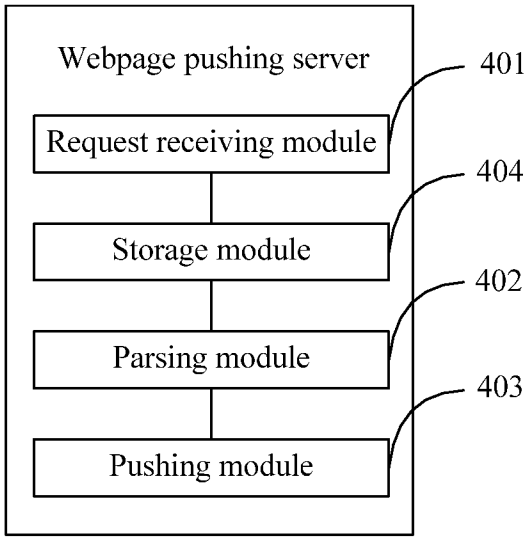
Webpage pushing server
401
Request receiving module
404
Storage module
402
Parsing module
403
Pushing module

FIG. 13

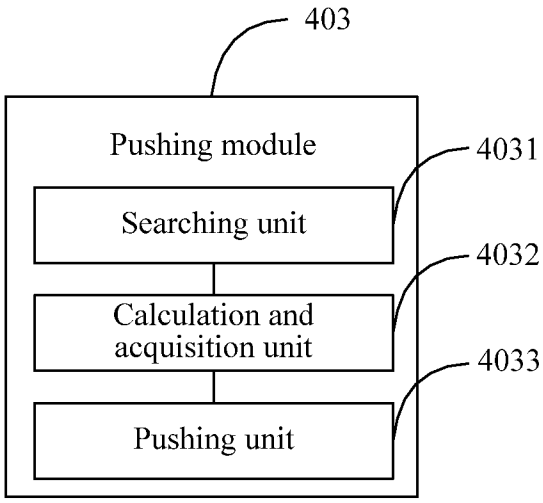
403
Pushing module
4031
Searching unit
4032
Calculation and acquisition unit
4033
Pushing unit

FIG. 14

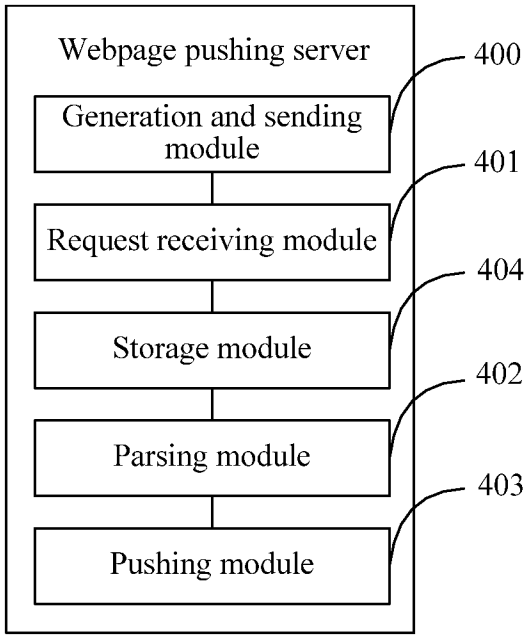
Webpage pushing server
400
Generation and sending module
401
Request receiving module
404
Storage module
402
Parsing module
403
Pushing module

FIG. 15

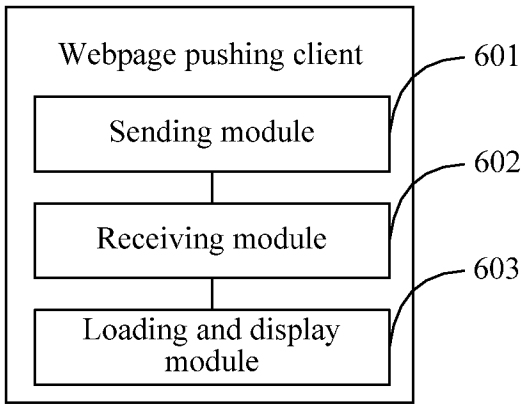
Webpage pushing client
601
Sending module
602
Receiving module
603
Loading and display module

FIG. 16

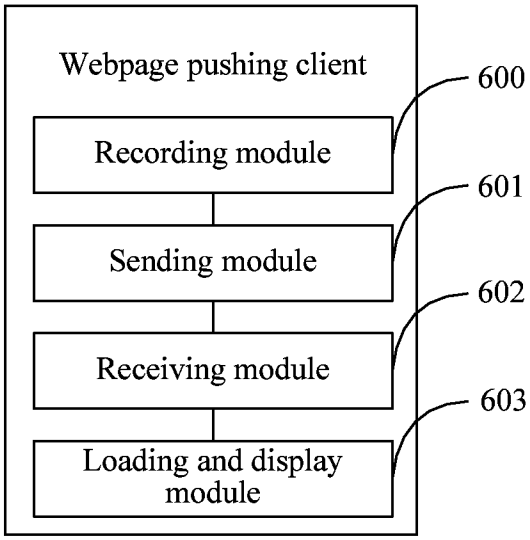
Webpage pushing client
600
Recording module
601
Sending module
602
Receiving module
603
Loading and display module

FIG. 17

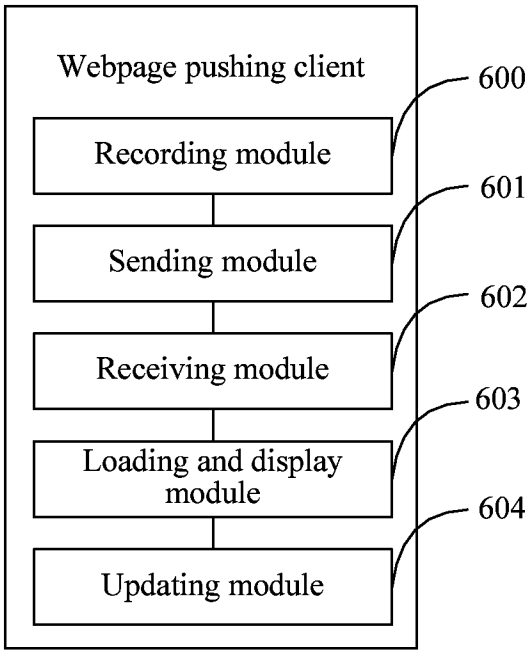
Webpage pushing client
600
Recording module
601
Sending module
602
Receiving module
603
Loading and display module
604
Updating module

FIG. 18

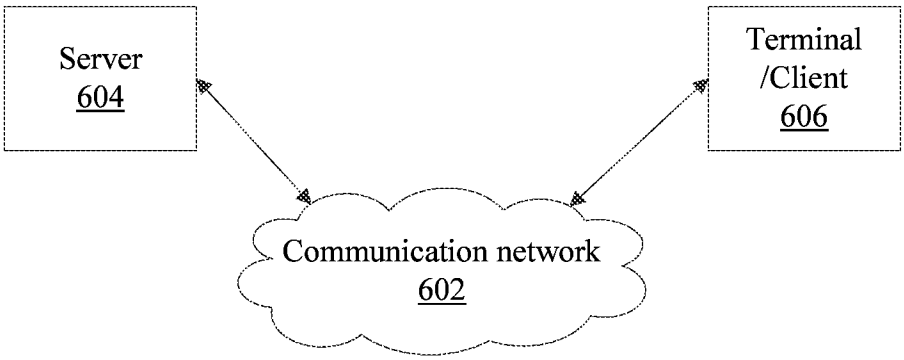
Server
604
Terminal
/Client
606
Communication network
602

FIG. 19

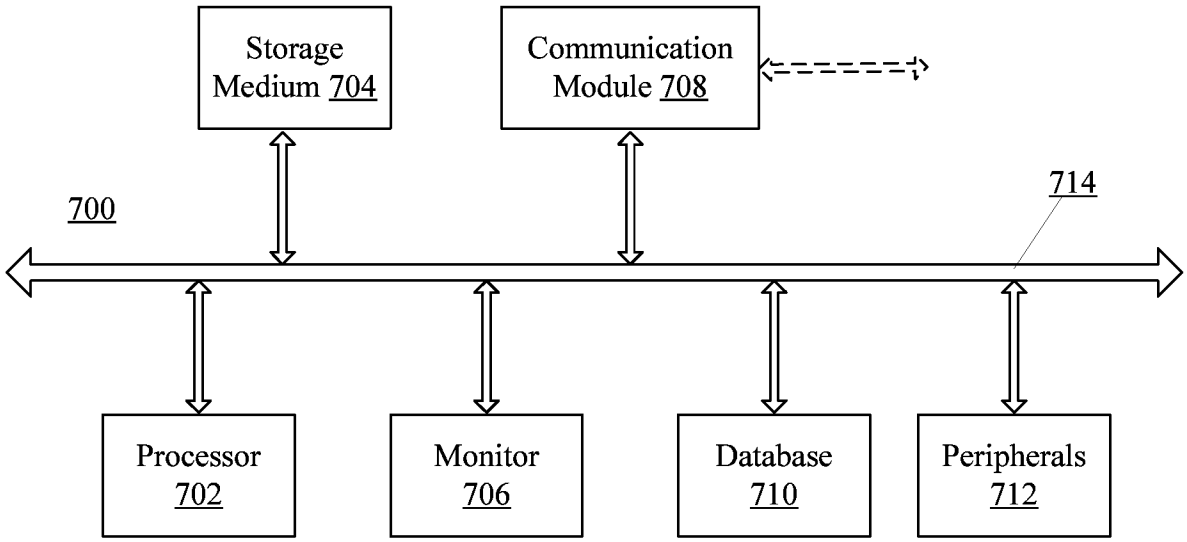
Storage
Medium 704
Communication
Module 708
700
714
Processor
702
Monitor
706
Database
710
Peripherals
712

FIG. 20

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2015/070746

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/16(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F; H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNABS, CNTXT, CNKI, VEN, USTXT: web, page, load, cache, javascript, resource, CSS, picture, HTML, URL, push, client, server

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 20120084346 A (MICROSOFT CORPORATION) 05 April 2012 (2012-04-05) description, paragraphs [0052]-[0068] and figures 1-6	1-19
A	CN 102185923 A (GUANGZHOU UCWEB COMPUTER TECHNILOGY CO., LTD.) 14 September 2011 (2011-09-14) the whole document	1-19
A	CN 103440276 A (XINGYUN RONGCHUANG BEIJING INFORMATION TECHNOLOGY CO., LTD.) 11 December 2013 (2013-12-11) the whole document	1-19

☐ Further documents are listed in the continuation of Box C. ☑ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 April 2015	**20 April 2015**
Name and mailing address of the ISA/CN **STATE INTELLECTUAL PROPERTY OFFICE OF THE P.R.CHINA(ISA/CN)** **6,Xitucheng Rd., Jimen Bridge, Haidian District, Beijing 100088, China** Facsimile No. **(86-10)62019451**	Authorized officer **YU,Hongrui** Telephone No. **(86-10)62411490**

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/CN2015/070746

Patent document cited in search report	Publication date (day/month/year)	Patent family member(s)	Publication date (day/month/year)
US 20120084346 A	05 April 2012	None	
CN 102185923 A	14 September 2011	US 2014201617 A1	17 July 2014
		WO 2012155849 A1	22 November 2012
CN 103440276 A	11 December 2013	None	