

CACHING IN A CONTENT DELIVERY FRAMEWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This Patent Cooperation Treaty (PCT) patent application claims priority to United States Nonprovisional Application No. 14/570,512 entitled "CACHING IN A CONTENT DELIVERY FRAMEWORK," filed on December 15, 2015 which is hereby incorporated by reference in its entirety.

COPYRIGHT STATEMENT

[0002] This patent document contains material subject to copyright protection. The copyright owner has no objection to the reproduction of this patent document or any related materials in the files of the United States Patent and Trademark Office, but otherwise reserves all copyrights whatsoever.

INCORPORATION BY REFERENCE

[0003] The following U.S. Patent and published U.S. patent applications are hereby fully incorporated herein by reference for all purposes:

1. U.S. Published Patent Application No. US 2013-0159472, titled "Content Delivery Network," filed December 12, 2012;
2. U.S. Published Patent Application No. US 2013-0159473, titled "Content Delivery Network," filed December 12, 2012;
3. U.S. Published Patent Application No. US 2014-0344399 AI, titled "Origin Server-Side Channel In A Content Delivery Framework," filed June 17, 2014; and
4. U.S. Patent No. 6,185,598, titled "Optimized Network Resource Location," filed February 10, 1998.

FIELD OF THE INVENTION

[0004] This invention relates to content delivery and content delivery networks, and, more specifically, to caching in content delivery networks.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Other objects, features, and characteristics of the present invention as well as the methods of operation and functions of the related elements of structure, and the combination of parts and economies of manufacture, will become more apparent upon consideration of the following description and the appended claims with reference to the accompanying drawings, all of which form a part of this specification.

[0006] FIG. 1 shows aspects of an exemplary content delivery framework in accordance with an exemplary embodiment hereof;

[0007] FIG. 2 shows aspects of an exemplary content delivery network (CDN) according to exemplary embodiments hereof;

[0008] FIG. 3 is a flowchart of aspects of processing according to exemplary embodiments hereof; and

[0009] FIG. 4 depicts aspects of computing according to exemplary embodiments hereof.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EXEMPLARY EMBODIMENTS**GLOSSARY**

[0010] As used herein, unless used otherwise, the following terms or abbreviations have the following meanings:

[0011] CD means Content Delivery;

[0012] CDN means Content Delivery Network;

[0013] HTTP means Hyper Text Transfer Protocol;

[0014] HTTPS means HTTP Secure;

- [0015] TTL means time to live;
- [0016] URI means Uniform Resource Identifier;
- [0017] URL means Uniform Resource Locator.

BACKGROUND AND OVERVIEW

[0018] A content delivery network (CDN) distributes content (e.g., resources) efficiently to clients on behalf of one or more content providers, preferably via a public Internet. Content providers provide their content (e.g., resources) via origin sources (origin servers or origins), and a CDN can also provide an over-the-top transport mechanism for efficiently sending content in the reverse direction — from a client to an origin server. Both end-users (clients) and content providers benefit from using a CDN. Using a CDN, a content provider is able to take pressure off (and thereby reduce the load on) its own servers (e.g., its origin servers). Clients benefit by being able to obtain content with fewer delays.

[0019] With reference to FIG. 1 which shows an exemplary content delivery framework 100 according to exemplary embodiments hereof, content 102 associated with one or more subscribers 104 may be distributed to clients 106 via one or more CDNs 108. The CDN(s) 108 may be any kind of CDN and may include sub-CDNs, delegated CDNs, partial CDNs, etc. Exemplary CDNs are described in U.S. Published Patent Applications Nos. US 2013-0159472 and US 2013-0159473, both filed December 12, 2012; and No. US 2014-0344399 A1, filed June 17, 2014; and U.S. Patent No. 6,185,598, filed February 10, 1998, the entire contents of each of which has been fully incorporated herein by reference for all purposes. FIG. 2 shows aspects of an exemplary CDN 108 according to exemplary embodiments hereof.

[0020] As used herein, clients are agents (e.g., browsers, set-top boxes, or other applications) used, e.g., by end users to issue requests (e.g., DNS and HTTP requests) within the system. When no CDN or other intermediaries are in use, such requests may go directly to the subscriber's own servers (e.g., their origin servers) or to other components in the Internet. When a content provider subscribes to CD services (e.g., as described U.S. Published Patent Applications Nos. US 2013-0159472 and US 2013-0159473), various requests may go to intermediate CD services that may map the end-user requests to origin requests, possibly transforming and caching content along the way.

[0021] Each distinct origin (e.g., origin server) is typically associated with one subscriber, but a subscriber may be associated with any number of origins, including subscriber-owned and CDN provided origins.

[0022] The physical origins with which the CDN interacts may actually be intermediaries that acquire content from a chain of intermediaries, perhaps, e.g., elements of a separate content acquisition system that ultimately terminates at a subscriber's actual origin servers. As far as the internals of the CDN are concerned, however, the origin is that service outside the system boundary from which content is directly acquired.

[0023] As used herein, an end user is an entity (e.g., person or organization) that ultimately consumes some Internet service (e.g., a web site, streaming service, etc.) provided by a service provider entity. This provider entity is sometimes referred to as a subscriber in this description because they subscribe to CDN services in order to efficiently deliver their content, e.g., from their origins to their consumers. A CDN may provide value-added mediation (e.g., caching, transformation, etc.) between its subscribers and their end-users.

CACHING

[0024] Content (e.g., a resource) may be cached both within a CDN and by other entities, including by a client to which that content has been served. As should be appreciated, a client may also be an entity within a CDN. In some cases content may be cached by intermediaries both within a CDN and external to the CDN. With reference again to the content delivery framework 100 in FIG. 1, at least some content 102 may be cached within the CDN(s) 108, e.g., CDN caches or cache services 110.

[0025] Cached and cacheable content typically has a caching policy associated therewith, and it is generally expected that entities that process such content will obey those caching policies. Caching policies of particular interest here include those policies relating to the expiration of content, it being appreciated that it is generally undesirable to serve stale or expired content or to serve content in a manner that is inconsistent with the caching policy associated with that content.

[0026] Caching policies may be specified as cache directives associated with content, and cache directives may depend, e.g., on the type of resource. For example, HTTP resources (including HTTPS resources) preferably use conventional HTTP caching directives

such as described in Internet Engineering Task Force (IETF) Request for Comments (RFC) 7234, "Hypertext Transfer Protocol (HTTP/1.1): Caching," the entire contents of which are hereby fully incorporated herein by reference for all purposes. As is known, in the case of HTTP resources the cache directives may be provided via headers associated with the resources. If no cache directives are provided with a resource then an entity (e.g., a cache node) may use default caching policies for resources of that type or from that provider. The entity may determine the default caching policies from another source. In general, a cache directive (or caching policy) associated with content (e.g., a resource or set of resources) may be provided with that content (e.g., as a header in the case of an HTTP resource) and/or be available from a location or entity (e.g., database or the like) distinct from the content.

[0027] A CDN may cache content on behalf of its subscribers, and preferably a CDN allows for specification, e.g., within a CDN node, of a cache expiration policy of content the CDN caches on behalf of its subscribers. A CDN may have or support an internal caching policy for content (e.g., a resource or set of resources), where the internal caching policy is a policy honored by the CDN cache that is separate and possibly distinct from an external caching policy for the same content (e.g., the caching policy that is conveyed with the content when being delivered to a client).

[0028] In some embodiments a CDN may support a subscriber or other user or entity setting or overriding an internal caching policy for resources. For example, in such a system, a CDN customer/subscriber may access a self-service portal or the like (e.g., via an Application Program(ing) Interface — API) and override a caching policy from the origin server for some or all of their content. For example, a particular CDN subscriber may use a portal to set a caching policy for a particular kind of their content (e.g., for their PDF files or the like), e.g., so that those files are cached for a particular time period (e.g., a week, an hour, etc.).

[0029] Rather than try to invalidate cached content (e.g., resources) every time a caching policy for that content is set, as described herein, the CDN's internal request processing process may determine the caching policy of requested content at request time rather than at fill time. In effect, the request processing uses a late-binding of the caching policy, thereby assuring that the latest internal caching policy is used without having to force invalidations (a potentially expensive process).

[0030] Cache directives similar to those used for external caching may be used for internal cache policies. It should be appreciated, however, that in order to conveniently distinguish internal from external cache policy directives, a CDN will use different names for the internal cache policies and/or provide these policies (e.g., via an internally used HTTP header). In this manner the external cache policies may be provided to external entities while the internal cache policies may be acted upon by the CDN entities (e.g., caches).

[0031] For example, with reference again to the content delivery framework 100 in FIG. 1, content 102 from the subscriber(s) 104 may have internal and/or external cache policies (e.g., cache directives) associated therewith. Content 102' provided by the CDN 108 to the client(s) 106 preferably includes external cache policies (e.g., cache directives). The internal cache policies may be provided with the content 102' to the client(s) 106, since these policies preferably use directives that the client will not recognize or act on. The cache policies associated with particular content may be included with the content (e.g., in the case of HTTP resources, as HTTP headers), or they may be located elsewhere. A CDN may apply a default caching policy to content in the absence of specific directives associated with that content.

[0032] In some embodiments hereof the caching policies (internal and external) associated with content (e.g., a resource or group of resources) may be stored in a database (sometimes referred to herein as a RuleBase). When a client request for a resource is being processed the RuleBase may be accessed in order to lookup/obtain the required caching policies (internal and external). With reference again to FIG. 1, cache(s) 110 may access one or more databases 112 in order to obtain information about internal and/or external caching policies associated with content.

[0033] As noted above, in some embodiments a CDN may support a subscriber or other user or entity setting or overriding an internal caching policy for resources, e.g., via a self-service portal or the like (e.g., via an API). An exemplary portal 114 is shown in FIG. 1 whereby subscribers may access and/or modify caching policies associated with their content (i.e., with content to be served by the CDN on their behalf).

[0034] Thus, according to exemplary embodiments hereof, as a request for content is received at a CDN node from a client, the node will determine the caching expiration policy for that content (e.g., is there an override policy and if so what is it; otherwise a policy

provided by the origin server should be honored). The caching policy for content may be determined, e.g., by access to the RuleBase.

[0035] This policy is then conveyed to the caching node (also referred to as a cache), e.g., as one or more cache constraints. The cache may then use those constraint(s) to determine if the content it has on hand (if any) can be returned to the requesting client, or if a refresh/reload etc. is required. As should be appreciated, if the node does not have the resource available then it needs to obtain the resource regardless of the internal caching policy, although it may honor the then-current caching policy.

[0036] For example, if the configuration specifies an internal caching policy of "one hour" then that policy is conveyed to the cache when the application requests the resource. If the cache has the resource, then it compares the original fill time of the resource it has to the specified policy (and the current time) to determine if the resource it has can be served from the cached resource or if it needs to be refreshed/reloaded first (or possibly asynchronously).

[0037] If the requested resource is not in cache, then a fill is made and stored in cache if and only if the specified policy allows or requires it (e.g., if the response is marked as no-store and there is no internal policy provided that specifies a caching policy, then do not store it, etc.).

[0038] Absent any internal caching policy override, the caching rules of the responses from the origin would still apply (i.e., the cache would return the resource if it saw that the resource was fresh and had no constraints).

[0039] Any external policy would be applied as the resource is delivered to the client. This allows for different classes of client to specify different constraints. For instance, in some cases an internal request for a resource could request the resource in such a way as to always prevent a refresh from occurring; or could similarly require a refresh regardless of what the cache considers the expiration of the resource to be.

[0040] FIG. 3 is a flowchart of aspects of processing according to exemplary embodiments hereof. As shown in the flowchart in FIG. 3, a cache (caching node) in the CDN receives a request for content (C) (at 302). The request may have reached the caching node in any manner (e.g., via a rendezvous system of the CDN). The content C may be or comprise one or more resources and the request may take the form of a URI or URL specifying those resources. The cache determines (at 304) whether the requested content C is cached locally. If

it is determined (at 304) that no locally cached copy of C is available then the node gets a copy of the content C, preferably along with C's caching policy (at 306). The node may obtain the copy of C from an appropriate fill-authoritative node in the CDN, it being understood that the fill-authoritative node will also check that its cached version, if any, is not stale. In other words, the late binding cache policy checking is preferably performed by all intermediate nodes that provide a copy of requested content, whether they are providing that content to an external or internal client.

[0041] On the other hand, if it is determined (at 304) that there is a locally cached copy of the requested content C, then the node obtains (at 310) the current cache policy associated with content C (e.g., from the database 112). The current cache policy is then used (at 312) to determine whether the node can serve the currently cached version of the requested content (C). If, based on the current cache policy obtained/determined in 310, it is not okay to serve the locally cached version of the requested content C, then the node gets the current version of the content C (at 306), as described above and serves the current version of C (at 308). On the other hand, if it is determined (at 312) that it is acceptable for the node to serve the currently-cached version of the content C (based, at least in part, on the current cache policy for C obtained at 310), then the currently-cached version of the content C is served (at 308).

[0042] Caching policies specified in the RuleBase, e.g., policies that are specified based on mime type and/or status code (negative caching being a particular case of this) may be dealt with by extending the constraints passed to the cache. For example, "if cached for no more than an hour, unless it is text/html in which case consider it non-cacheable or text/* in which case cacheable for one day." It should be appreciated that a suitably terse syntax would apply to and be used for such constraints.

[0043] In some cases the RuleBase constraint set may be compilable at sequence generation time since it typically applies property wide — there would need to be provision for additional filters however (the set should be limited and readily selectable amongst during request preparation); negative caching could similarly be handled by including a negative TTL value and a list of status codes to the constraint set.

[0044] As will be appreciated, a downside of this approach is that the caching policy determination must be made for every request. This is unlikely to be free, and could be relatively costly. On the other hand, the benefit of having relatively instant policy updates may not be big

since policy updates may be rare. To mitigate that cost, an implementation may store a configuration version number with each resource and only recalculate it if the configuration has changed. For example, such an approach would call into the caching engine with the current configuration version number and have the cache compare that to the configuration number associated with the resource. If the configuration numbers are the same, then use the policy information attached to the resource, otherwise call back out to get the current set of constraints. Effectively this approach will only process the RuleBase to get the constraints if it looks like they have changed. In some implementations the version number may be or comprise a checksum or hash or the like of the caching policy (as well as or instead of a sequence number). The hash may be a cryptographic hash such as an MD5 or SHA or the like. In this manner a change that is subsequently undone may be seen as matching a prior version.

[0045] Such a system would deal with odd constraints (e.g., the type of internal request that never wants to cause a refresh) by entering some sentinel sequence number that causes the current constraint to be requested. To support such an approach, the interface to the cache may be such that the request would be allowed to convey a sequence number and/or a set of constraints. If the RuleBase needs to be processed for every request anyway, then the constraints can be gathered during that and conveyed on the request to the cache, which could just ignore it if the sequence number hasn't changed rather than calling out from the cache.

COMPUTING

[0046] FIG. 4 is example schematic diagram of a computing system 400 implementing a proxy, a parent server, or other computing device that may be used to perform the processes discussed herein. The computing system includes a bus 402 (i.e., interconnect), at least one processor 404, at least one communication port 414, a main memory 406, a removable storage media 410, a read-only memory 408, and a mass storage device 412. Processor(s) 404 can be any known processor, such as, but not limited to, an Intel® Itanium® or Itanium 2® processor(s), AMD® Opteron® or Athlon MP® processor(s), or Motorola® lines of processors. Communication port 414 can be any of an RS-232 port for use with a modem based dial-up connection, a 10/100 Ethernet port, a Gigabit port using copper or fiber, or a USB port. Communication port(s) 414 may be chosen depending on a network such as a Local Area Network (LAN), a Wide Area Network (WAN), or any network to which the computer system connects. The server may be in communication with peripheral devices (e.g., display screen 416, input device 418 via Input/Output (I/O) port 420.

[0047] Main memory 406 can be Random Access Memory (RAM) or any other dynamic storage device(s) commonly known in the art. Read-only memory 408 can be any static storage device(s) such as Programmable Read-Only Memory (PROM) chips for storing static information such as instructions for processor 404. Mass storage device 412 can be used to store information and instructions. For example, hard disks such as the Adaptec® family of Small Computer Serial Interface (SCSI) drives, an optical disc, an array of disks such as Redundant Array of Independent Disks (RAID), such as the Adaptec® family of RAID drives, or any other mass storage devices, may be used.

[0048] Bus 402 communicatively couples processor(s) 404 with the other memory, storage and communications blocks. Bus 402 can be a PCI / PCI-X, SCSI, or Universal Serial Bus (USB) based system bus (or other) depending on the storage devices used. Removable storage media 410 can be any kind of external hard drives, floppy drives, IOMEGA® Zip Drives, Compact Disc — Read Only Memory (CD-ROM), Compact Disc — Re-Writable (CD-RW), Digital Video Disk — Read Only Memory (DVD-ROM), etc.

[0049] Embodiments herein may be provided as a computer program product, which may include a machine-readable medium having stored thereon instructions which may be used to program a computer (or other electronic devices) to perform a process. The machine-readable medium may include, but is not limited to, floppy diskettes, optical discs, CD-ROMs, magneto-optical disks, ROMs, RAMs, erasable programmable read-only memories (EPROMs), electrically erasable programmable read-only memories (EEPROMs), magnetic or optical cards, flash memory, or other type of media/machine-readable medium suitable for storing electronic instructions.

[0050] As shown, main memory may be encoded with one or more applications/services 422 that support functionality as discussed above with respect to the various figures and elsewhere. For example, in one embodiment, the application 422 may include or otherwise implement the various processes and/or instructions described herein. The application 422 (and/or other resources as described herein) can be embodied as software code such as data and/or logic instructions (e.g., code stored in the memory or on another computer readable medium such as a disk) that supports processing functionality according to different embodiments described herein. During operation of one embodiment, processor(s) 404 accesses main memory 406 via the use of bus 402 in order to launch, run, execute, interpret or otherwise perform the logic instructions of the application 422. Execution of the application 422

produces processing functionality in application process 424. In other words, the process 424 represents one or more portions of the application 422 performing within or upon the processor(s) 404 in the computer system 400.

[0051] The description above includes example systems, methods, techniques, instruction sequences, and/or computer program products that embody techniques of the present disclosure. However, it is understood that the described disclosure may be practiced without these specific details. In the present disclosure, the methods disclosed may be implemented as sets of instructions or software readable by a device. Further, it is understood that the specific order or hierarchy of steps in the methods disclosed are instances of example approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the method can be rearranged while remaining within the disclosed subject matter. The accompanying method claims present elements of the various steps in a sample order, and are not necessarily meant to be limited to the specific order or hierarchy presented.

CONCLUSION

[0052] As used herein, including in the claims, the phrase "at least some" means "one or more," and includes the case of only one. Thus, e.g., the phrase "at least some services" means "one or more services", and includes the case of one service.

[0053] As used herein, including in the claims, the phrase "based on" means "based in part on" or "based, at least in part, on," and is not exclusive. Thus, e.g., the phrase "based on factor X" means "based in part on factor X" or "based, at least in part, on factor X." Unless specifically stated by use of the word "only", the phrase "based on X" does not mean "based only on X."

[0054] As used herein, including in the claims, the phrase "using" means "using at least," and is not exclusive. Thus, e.g., the phrase "using X" means "using at least X." Unless specifically stated by use of the word "only", the phrase "using X" does not mean "using only X."

[0055] In general, as used herein, including in the claims, unless the word "only" is specifically used in a phrase, it should not be read into that phrase.

[0056] As used herein, including in the claims, the phrase "distinct" means "at least partially distinct." Unless specifically stated, distinct does not mean fully distinct. Thus, e.g., the phrase, "X is distinct from Y" means that "X is at least partially distinct from Y," and does not

mean that "X is fully distinct from Y." Thus, as used herein, including in the claims, the phrase "X is distinct from Y" means that X differs from Y in at least some way.

[0057] As used herein, including in the claims, a list may include only one item, and, unless otherwise stated, a list of multiple items need not be ordered in any particular manner. A list may include duplicate items. For example, as used herein, the phrase "a list of CDN services" may include one or more CDN services.

[0058] It should be appreciated that the words "first" and "second" in the description and claims are used to distinguish or identify, and not to show a serial or numerical limitation. Similarly, the use of letter or numerical labels (such as "(a)", "(b)", and the like) are used to help distinguish and / or identify, and not to show any serial or numerical limitation or ordering.

[0059] No ordering is implied by any of the labeled boxes in any of the flow diagrams unless specifically shown and stated. When disconnected boxes are shown in a diagram, the activities associated with those boxes may be performed in any order, including fully or partially in parallel.

[0060] While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiments, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

CLAIMS

1. A computer-implemented method, operable with a content delivery network (CDN), the method comprising, by a node in the CDN:

(A) in response to a request for content, determining if the content is cached locally;

(B) when it is determined that said content is cached locally, then:

(B)(1) determining a current cache policy associated with the content; and

then

(B)(2) determining, based on said current cache policy associated with the content, whether it is acceptable to serve the content that is cached locally;

(B)(3) based on said determining in (B)(2),

(i) when it is not acceptable to serve the content that is cached locally, obtaining a new version of the content and then serving the new version of the content, and

(ii) when it is acceptable to serve the content that is cached locally, serving the content that is cached locally;

wherein version identifiers are associated with caching policies, and wherein said determining said current cache policy in (B)(1) comprises: (i) comparing a first version identifier associated with the current cache policy with a second version identifier associated with a cache policy associated with the content at the node in the CDN; and (ii) using the cache policy associated with the content at the node as the current cache policy when the first version identifier is identical to the second version identifier, otherwise obtaining the current cache policy from a location distinct from the node.

2. The method of claim 1 further comprising:

(C) when it is determined that said content is not cached locally, then (C)(1) obtaining the content and then (C)(2) serving the content obtained in (C)(1).

3. The method of claim 1 or 2 wherein the request comprises an HTTP request and wherein the content comprises one or more resources.

4. The method of any preceding claim wherein the current cache policy is determined from a location distinct from the node.

5. The method of claim 4 wherein the location is a database.

6. The method of claim 1 wherein the cache policy relates to caching content within the CDN.

7. The method of claim 6 wherein the cache policy also relates to caching content outside the CDN.

8. A computer program product having computer readable instructions stored on non-transitory computer readable media, the computer readable instructions including instructions for implementing a computer-implemented method, said method operable on a device comprising hardware including memory and at least one processor and running a service on said hardware, said method operable in a content delivery service (CDN), said method comprising:

(A) in response to a request for content, determining if the content is cached locally;

(B) when it is determined that said content is cached locally, then:

(B)(1) determining a current cache policy associated with the content; and then

(B)(2) determining, based on said current cache policy associated with the content, whether it is acceptable to serve the content that is cached locally;

(B)(3) based on said determining in (B)(2),

- (i) when it is not acceptable to serve the content that is cached locally, obtaining a new version of the content and then serving the new version of the content, and
- (ii) when it is acceptable to serve the content that is cached locally, serving the content that is cached locally;

wherein version identifiers are associated with caching policies, and wherein said determining said current cache policy in (B)(1) comprises: (i) comparing a first version identifier associated with the current cache policy with a second version identifier associated with a cache policy associated with the content at the node in the CDN; and (ii) using the cache policy associated with the content at the node as the current cache policy when the first version identifier is identical to the second version identifier, otherwise obtaining the current cache policy from a location distinct from the node.

9. The computer program product of claim 8 wherein the method further comprises:

(C) when it is determined that said content is not cached locally, then (C)(1) obtaining the content and then (C)(2) serving the content obtained in (C)(1).

10. The computer program product of claim 8 wherein the request comprises an HTTP request and wherein the content comprises one or more resources.

11. The computer program product of claims 8 wherein the current cache policy is determined from a location distinct from the node.

12. The computer program product of claim 11 wherein the location is a database.

13. The computer program product of claim 8 wherein the cache policy relates to caching content within the CDN.

14. The computer program product of claim 13 wherein the cache policy also relates to caching content outside the CDN.