

CACHE OPTIMIZATION

BACKGROUND

[0001] Generally described, computing devices and communication networks can be utilized to exchange information. In a common application, a computing device can request content from another computing device via the communication network. For example, a user at a personal computing device can utilize a software browser application to request a Web page from a server computing device via the Internet. In such embodiments, the user computing device can be referred to as a client computing device and the server computing device can be referred to as a content provider.

[0002] Content providers are generally motivated to provide requested content to client computing devices often with consideration of efficient transmission of the requested content to the client computing device and/or consideration of a cost associated with the transmission of the content. For larger scale implementations, a content provider may receive content requests from a high volume of client computing devices which can place a strain on the content provider's computing resources. Additionally, the content requested by the client computing devices may have a number of components, which can further place additional strain on the content provider's computing resources.

[0003] With reference to an illustrative example, a requested Web page, or original content, may be associated with a number of additional resources, such as images or videos, that are to be displayed with the Web page. In one specific embodiment, the additional resources of the Web page are identified by a number of embedded resource identifiers, such as uniform resource locators ("URLs"). In turn, software on the client computing devices typically processes embedded resource identifiers to generate requests for the content. Often, the resource identifiers associated with the embedded resources reference a computing device associated with the content provider such that the client computing device would transmit the request for the additional resources to the referenced content provider computing device. Accordingly, in order to satisfy a content request, the content provider would provide client computing devices data associated with the Web page as well as the data associated with the embedded resources.

[0004] Some content providers attempt to facilitate the delivery of requested content, such as Web pages and/or resources identified in Web pages, through the utilization of a content delivery network ("CDN") service provider. A CDN server

provider typically maintains a number of computing devices in a communication network that can maintain content from various content providers. In turn, content providers can instruct, or otherwise suggest to, client computing devices to request some, or all, of the content provider's content from the CDN service provider's computing devices.

[0005] As with content providers, CDN service providers are also generally motivated to provide requested content to client computing devices often with consideration of efficient transmission of the requested content to the client computing device and/or consideration of a cost associated with the transmission of the content. Accordingly, CDN service providers often consider factors such as latency of delivery of requested content in order to meet service level agreements or to generally improve the quality of delivery service.

DESCRIPTION OF THE DRAWINGS

[0006] The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same become better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings, wherein:

[0007] Figure 1 is a block diagram illustrative of one embodiment of a content delivery network (CDN) based system including one or more cache servers;

[0008] Figure 2 is a block diagram of the CDN-based system of Figure 1 illustrating the processing of a content request by a content provider;

[0009] Figure 3 is a block diagram of the CDN-based system of Figure 1 illustrating the processing of DNS query and assignment of a resource cache component;

[0010] Figures 4A-4F are simplified block diagrams of the CDN-based system of Figure 1 illustrating embodiments associated with the processing of the resource request by a cache server;

[0011] Figure 5 is a flow diagram of an illustrative method for processing a resource request at a cache server; and

[0012] Figure 6 is a flow diagram of an illustrative method for managing storage of a resource at a cache server.

DETAILED DESCRIPTION

[0013] Generally described, the present disclosure is directed to the management of cache resources utilized when a client computing device requests content

from a network resource, such as content delivery network (“CDN”) service providers. Specifically, aspects of the disclosure will be described with regard to the processing, by a resource cache component, of content, and segmentation of the content with respect to both the storage and retrieval thereof. Although various aspects of the disclosure will be described with regard to illustrative examples and embodiments, one skilled in the art will appreciate that the disclosed embodiments and examples should not be construed as limiting.

[0014] Figure 1 is a block diagram illustrative of a content delivery environment 100 for the management of content storage and delivery. As illustrated in Figure 1, the content delivery environment 100 includes a number of client computing devices 102 (generally referred to as clients) for requesting content from a content provider and/or a CDN service provider. In an illustrative embodiment, the client computing devices 102 can correspond to a wide variety of computing devices including personal computing devices, laptop computing devices, hand-held computing devices, terminal computing devices, mobile devices, wireless devices, various electronic devices and appliances and the like. In an illustrative embodiment, the client computing devices 102 include necessary hardware and software components for establishing communications over a communication network 108, such as a wide area network or local area network. For example, the client computing devices 102 may be equipped with networking equipment and browser software applications that facilitate communications via the Internet or an intranet.

[0015] Additionally, the client computing devices 102 may also include necessary hardware and software components to execute, or otherwise process, translation information as will be described in greater detail below. One skilled in the relevant art will appreciate the additional hardware/software components for processing the translation information may be included with the execution of a multi-purpose software application, such as a browser software application. Alternatively, some or all of the additional hardware/software components may be embodied in stand alone or specialized components configured for processing the translation information. Although not illustrated in Figure 1, each client computing device 102 utilizes some type of local DNS resolver component, such as a DNS Name server, that generates the DNS queries attributed to the client computer. In one embodiment, the local DNS resolver component

may belong to an enterprise network to which the client computer belongs. In another embodiment, the local DNS resolver component may belong to an Internet Service Provider (ISP) that provides the network connection to the client computer.

[0016] The content delivery environment 100 can also include a content provider 104 in communication with the one or more client computing devices 102 via the communication network 108. The content provider 104 illustrated in Figure 1 corresponds to a logical association of one or more computing devices associated with a content provider. Specifically, the content provider 104 can include a web server component 110 corresponding to one or more server computing devices for obtaining and processing requests for content (such as Web pages) from the client computing devices 102. The content provider 104 can further include an origin server component 112 and associated storage component 114 corresponding to one or more computing devices for obtaining and processing requests for network resources from the CDN service provider. One skilled in the relevant art will appreciate that the content provider 104 can be associated with various additional computing resources, such as additional computing devices for administration of content and resources, DNS name servers, and the like.

[0017] With continued reference to Figure 1, the content delivery environment 100 can further include a CDN service provider 106 in communication with the one or more client computing devices 102 and the content providers 104 via the communication network 108. The CDN service provider 106 illustrated in Figure 1 corresponds to a logical association of one or more computing devices associated with a CDN service provider. Specifically, the CDN service provider 106 can include a number of Point of Presence (POP) locations 116, 122 that correspond to nodes on the communication network 108. Each POP 116, 122 includes a DNS component 118, 124 made up of a number of DNS server computing devices for resolving DNS queries from the client computers 102.

[0018] Each POP 116, 122 also includes a resource cache component 120, 126 for storing objects from content providers and transmitting various requested objects to various client computers. Each resource cache component 120, 126 is made up of a number of cache server computing devices 130, 132, 134 for obtaining and processing requests for network resources. Each cache server computing device 130, 132, 134

includes a memory 140, 142, 144 having the lowest data access latency, generally referred to as latency, for the corresponding cache server computing device. One skilled in the relevant art will appreciate that for purposes of the present disclosure data access latency, can include among other things, a minimal time period in which stored data can be retrieved from a memory location and available for transmission. In addition, each cache server computing device 130, 132, 134 can be associated with, either directly or shared via a bus or otherwise, a storage media 150, 152, 154 having a higher latency than the attached memory 140, 142, 144. Storage media 150, 152, 154 can include, for example, non-volatile memory such as a disk memory, flash memory, optical memory, and the like. Even further, the content delivery environment 100 can include a network-based memory 160 which can be utilized by cache server computing devices 130, 132, 134, as well as by other computing devices, for the management of content. Similar to storage media 150, 152, 154, the network-based memory 160 is associated with a higher latency than the attached memory 140, 142, 144.

[0019] In an illustrative embodiment, the DNS component 118, 124 and resource cache component 120, 126 are considered to be logically grouped, regardless of whether the components, or portions of the components, are physically separate. Additionally, although the POPs 116, 122 are illustrated in Figure 1 as logically associated with the CDN service provider 106, the POPs will be geographically distributed throughout the communication network 108 in a manner to best serve various demographics of client computing devices 102. Additionally, one skilled in the relevant art will appreciate that the CDN service provider 106 can be associated with various additional computing resources, such additional computing devices for administration of content and resources, and the like.

[0020] One skilled in the relevant art will appreciate that the components and configurations provided in Figure 1 are illustrative in nature. Accordingly, additional or alternative components and/or configurations, especially regarding the additional components, systems and subsystems for facilitating communications may be utilized.

[0021] With reference now to Figures 2-4, the interaction between various components of the content delivery environment 100 of Figure 1 will be illustrated. For purposes of the examples, however, the illustrations have been simplified such that many of the components utilized to facilitate communications are not shown. One skilled in the

relevant art will appreciate that such components can be utilized and that additional interactions would accordingly occur without departing from the spirit and scope of the present disclosure. Additionally, although communications may be illustrated as direct communications between components, one skilled in the relevant art will appreciate that all the illustrative communications may occur directly between components or facilitated via the communication network 108. Prior to discussing the management of objects by the cache server computing devices 140, 142, 144, a brief overview of the general processing of resource requests from a client computing device 102 in a CDN-based system will be described.

[0022] With reference to Figure 2, a client computing device 102 generates a content request that is received and processed by the content provider 104, such as through the Web server 110. In accordance with an illustrative embodiment, the request for content can be in accordance with common network protocols, such as the hypertext transfer protocol (“HTTP”). Upon receipt of the content request, the content provider 104 identifies the appropriate responsive content. In an illustrative embodiment, the requested content can correspond to a Web page that is displayed on the client computing device 102 via the processing of information, such as hypertext markup language (“HTML”), extensible markup language (“XML”), and the like. The requested content can also include a number of embedded resource identifiers, described above, that corresponds to resource objects that should be obtained by the client computing device 102 as part of the processing of the requested content. The embedded resource identifiers can be generally referred to as original resource identifiers or original URLs.

[0023] In one embodiment, the original URLs identify the domain of the CDN service provider 106 (e.g., “cdnprovider.com”), the same name of the resource to be requested (e.g., “resource.xxx”) and the same path where the resource will be found (e.g., “path”). Additionally, the original URL can include additional processing information (e.g., “additional information”). The original URL would have the form of:

http://additional_information.cdnprovider.com/path/resources.xxx

[0024] In another embodiment, the information associated with the CDN service provider 106 is included the original URL, such as through prepending or other

techniques, such that the original URL can maintain all of the information associated with a URL previously associated with the content provider 104. In this embodiment, the original URL would have the form of:

http://additional_information.cdnprovider.com/www.contentprovider.com/path/resource.xxx

[0025] Upon receipt of the requested content, the client computing device 102, such as through a browser software application, begins processing any of the markup code included in the content and attempts to acquire the resources identified by the embedded resource identifiers. Accordingly, the first step in acquiring the content correspond to the issuance, by the client computing device 102 (through its local DNS resolver), a DNS query for the Original URL resource identifier that results in the identification of a DNS server authoritative to the “.” and the “com” portions of the translated URL. After resolving the “.” and “com” portions of the embedded URL, the client computing device 102 then issues a DNS query for the resource URL that results in the identification of a DNS server authoritative to the “cdnprovider” portion of the embedded URL. The issuance of DNS queries corresponding to the “.” and the “com” portions of a URL are well known and have not been illustrated.

[0026] With reference now to Figure 3, in an illustrative embodiment, the successful resolution of the “cdnprovider” portion of the original URL identifies a network address, such as an IP address, of a DNS server component 118 associated with the CDN service provider 106. In one embodiment, the IP address is a specific network address unique to a DNS server component 118 of POP 116. In another embodiment, the IP address can be shared by one or more POPs 116, 122. In this embodiment, a DNS query to the shared IP address utilizes a one-to-many network routing schema, such as anycast, such a specific POP, POP 118, will receive the request as a function of network topology. For example, in an anycast implementation, a DNS query issued by a client computing device 102 to a shared IP address will arrive at a DNS server component logically having the shortest network topology distance, often referred to as network hops, from the client computing device. The network topology distance does not necessarily correspond to geographic distance. However, in some embodiments, the network

topology distance can be inferred to be the shortest network distance between a client computing device 102 and a POP.

[0027] With continued reference to Figure 3, in either of the above identified embodiments (or any other embodiment), a specific DNS server in the DNS component 118 of a POP 116 receives the DNS query corresponding to the original URL from the client computing device 102. Once one of the DNS servers in the DNS component 118 receives the request, the specific DNS server attempts to resolve the request. In an illustrative embodiment, a specific DNS server can resolve the DNS query by identifying an IP address of a cache server component that will process the request for the requested resource. As described above, a selected resource cache component 120, 126 can process the request by either providing the requested resource if it is available or attempt to obtain the requested resource from another source, such as a peer cache server computing device or the origin server 112 of the content provider 104.

[0028] Upon selection of a cache server computing device 140, 142, 144 (or a resource cache component 120, 126), the DNS server component 118 provides an IP address of the cache server computing device, resource cache component or load balancer/load share device associated with a resource cache component. The client computing device 102 can then utilize Internet communication protocols to request the resource from a cache server computing device 140, 142, 144 identified by the IP address. The cache server computing device 140, 142, 144 then processes the request, as will be described in greater detail below, to provide the resource to the client computing device 102. Specifically, the cache server computing device can begin transmitting an initialization portion of the requested content from a local memory while receiving the remaining portions of the requested the content from other storage locations. Upon receipt, the requested resource is then processed by the browser application on the client computing device 102 as appropriate.

[0029] Referring now to Figures 4A-4F, multiple embodiments of the interaction between various components of the content delivery environment 100 will be described with respect to the processing of a resource request by a cache server computing device 140, 142, 144. For purposes of the examples in Figures 4A-4F, however, the illustrations have been simplified such that many of the systems, subsystems, and components utilized to facilitate communications are not shown. In general, and as will

be described in greater detail below, objects stored in cache server computing devices 130, 132, 134 are each segmented into an initialization fragment for storage in memory 140, 142, 144 and one or more remaining fragments for storage in a media having a higher latency than the memory in which the initialization fragment is stored. In one embodiment, the one or more remaining fragments comprise a majority fragment.

[0030] With reference now to Figure 4A, a cache server computing device 132 receives a request for an object from a client computing device 102. Upon receipt of the request for the object, the cache server computing device 132 begins retrieving a majority fragment of the object from a storage media associated with the cache server computing device, such as a local disk memory 152 in this example. In one example, the storage location of the majority may be previously known to the cache server computing device 132 such that the request for the majority fragment portion may be sent immediately. Alternatively, in another example, the cache server computing device 132 may query a directory or a service to identify an appropriate storage location for the majority fragment portion. The service may be utilized to identify a best or available storage locations if the majority fragment portion is stored in multiple storage locations (such as for purposes of geographic distribution).

[0031] Substantially at the same time as the request for the majority fragment portion, the cache server computing device 132 begins transmitting the initialization fragment over a network to the requesting client computing device 102 from a memory component 142. In this embodiment, the local disk memory 152 likely has a higher latency associated with recall of data therefrom than the memory 142 (e.g. RAM) of the cache server computing device 132. Accordingly, retrieving the initialization fragment from the memory 132 allows the cache server computing device to begin transmitting at least a portion of the requested object as soon as the request is processed. At the same time, the size of the initialization fragment is sufficiently large such that the majority fragment, or fragments, can be retrieved prior to completion of the transmission of the initialization fragment portion. The cache server computing device 132 can the begin transmitting the majority fragment over the communication network to the requesting client computing device 102 upon completion of the transmission of the initialization fragment portion from memory 142. Alternatively, the cache server computing device 132 can the begin transmitting the majority fragment over the communication

network to the requesting client computing device 102 as soon as it begins receiving the majority fragment. Although the initialization fragment portion is shown as being provided from the local cache server computing device memory 142, one skilled in the relevant art will appreciate that the initialization fragment portion may be stored and transmitted from other storage locations and/or by other cache server computing devices.

[0032] In one embodiment, the majority fragment portion is not retained by the cache server computing device 132 for processing a subsequent request for the same object. One skilled in the relevant art will appreciate that in accordance with the operation of a cache server computing device, such a cache server computing device 132, the majority fragment portion may be stored in memory 142 in order to be transmitted to the client computing device 102. In this embodiment, however, such storage would be considered to be generally transient as the majority fragment portion may be deleted (or at least prioritized for overwriting).

[0033] It will be appreciated by one skilled in the relevant art that the precise timing of locating, retrieving and transmitting the initialization and majority fragments can vary. It will also be appreciated by one skilled in the relevant art that as a network includes a number of local and non-local storage media, the majority fragments can be stored on any storage media within a POP 116, 122 or in the network 108 having a higher latency than the memory on which the initialization fragment is stored. For example, one or more majority fragments of may be stored in the local disks of peer cache server computing devices of a POP, on a network-based storage 160, on the content provider's origin server 112, and the like. As described above, when an object is requested by a client computing device 102, the receiving cache server computing device can begin providing the initialization fragment immediately from a local memory, while the majority fragment segment is retrieved from another location such as a cache peer's disk.

[0034] In another illustrative embodiment, the requested object can be segmented into three or more fragments for retrieval. In this embodiment, the initialization fragment can be stored in memory of a cache server computing device in a manner as previously described. Additionally, an intermediate fragment stored on another storage location, such as a cache server computing device disk, while the majority fragment is stored on yet another storage location, such as on the origin server 112. When the initialization fragment begins to be served, the intermediate fragment is retrieved and

served, and the majority fragment is retrieved from the origin server. In one embodiment, the storage location of the intermediate fragment may have a higher latency than the memory (e.g., storage location of the initialization fragment portion) but a lower latency than the storage location of the majority fragment portion. Accordingly, the size of the intermediate fragment portion would be a function to the time required to retrieve the majority fragment portion. Alternatively, the latency of the storage locations of the intermediate and majority fragment portions may not be substantially different. In this embodiment, the intermediate and majority fragment portions may be allocated according to financial or service criteria, such as cost of storage, cost of bandwidth, guaranteed service availability, redundant storage, and the like. As also described above with regard to Figure 4A, in one embodiment, the initialization and majority fragment portions would not be retained by the cache server computing device for processing a subsequent request for the same object.

[0035] With reference now to Figures 4B-4D, the processing performed by the various components in these illustrative embodiments is similar to that described in reference to Figure 4A with the exception that the storage location from which the majority fragment is provided. With reference to Figure 4B, the cache server computing device 132 obtains the request for an object and then begins retrieving the majority fragment of the requested object from disk memory 150 of another cache server computing device 130. In this example, the cache server computing device 130 is in the same POP 116. In the same manner described above, the receiving cache server computing device 132 then transmits the initialization fragment portion and the majority fragment portion to the client computing device 102. As previously described, the majority fragment portion may be stored in memory 142 to facilitate the transmission to the client computing device 102. The majority fragment portion may not be maintained in the memory 142 however for subsequent requests for the object.

[0036] With reference now to Figure 4C, the cache server computing device 132 again receives the request for an object from the client computing device 102. In this illustrative example, however, the cache server computing device 132 requests the majority fragment from disk memory 150 of another cache server computing device 130 from a different POP 122. The cache server computing device 132 would then begin transmitting the initialization fragment portion while retrieving the majority fragment

portion. Subsequently, the majority fragment portion would be transmitted. In yet another alternative embodiment, the majority fragment may be retrieved from a separate network-based memory 160 as shown in Figure 4D, or from a storage media 114 associated with the origin server component 112 as shown in Figure 4E. In both of these embodiments, the size of the initialization fragment portion may be larger than the size of the initialization fragment portion discussed with regard to the examples in Figures 4B-4D in the event of additional latencies associated with the network communications. As will be described below, the size of the initialization fragment portion may be dynamically adjusted.

[0037] Turning now to Figure 4F, in yet a further embodiment, the requested object transmitted by the cache server computing device 132 can be segmented into an initialization fragment, an intermediate fragment, and a majority fragment. In particular, upon request of the object, the cache server computing device 132 retrieving the intermediate fragment associated with the object from a first storage media, such as hard disk 152. As previously described, the hard disk 152 has a higher latency than the memory 142 of the cache server computing device 132. At the same time, the cache server computing device 40, 142, 144 begins retrieving the majority fragment associated with the object from a second storage media, also having higher latency than the memory 142 of the cache server computing device 132. In this embodiment, the second storage media is a storage media associated with the origin server component 112. Substantially at the same time, the cache server computing device 132 begins transmitting the initialization fragment of the object from memory 142 over the network to the requesting client computing device 102. As soon as the intermediate fragment is retrieved, the cache server computing device starts transmitting the intermediate fragment over the network to the requesting client computing device, and likewise for the majority fragment. Again, it will be appreciated by one skilled in the relevant art that the precise timing of locating, retrieving and transmitting the initialization, intermediate, and majority fragments can vary. Additionally, as previously described, although the intermediary and majority fragment portions may be stored in memory 142 to facilitate the transmission to the client computing device 102, these portions may not be maintained in the memory 142 however for subsequent requests for the object.

[0038] With reference now to Figure 5, one embodiment of a routine 500 for processing a resource request at a cache server computing device 130, 132, 134 will be described. At block 502, the routine 500 begins with the receipt of a request for an object at the cache server computing device 130, 132, 134. As previously described, a specific IP address of a cache server may be assigned by a DNS server associated with the CDN service provider 106. Alternatively, a specific cache server computing device 130, 132, 134 may be selected by software/hardware components at a resource cache component.

[0039] At block 504, the cache server computing device 130, 132, 134 finds and begins transmission of an initialization fragment for the object from memory 140, 142, 144 of the cache server computing device. In an illustrative embodiment, the memory 140, 142, 144 corresponds to a local memory associated with the receiving cache server computing device. Alternatively, the memory may correspond to a memory of a peer cache server computing device, such as within the same POP or across POPs. Still further, although the initialization fragment portion has always been illustrated as stored on a memory, one skilled in the relevant art will appreciate that the initialization fragment portion may also be provided from other storage locations. Such variations are considered to be within the scope and spirit of the present disclosure.

[0040] At block 506, the cache server computing device begins retrieving all remaining portions of the object, including all intermediary and majority fragments. In an illustrative embodiment, the intermediary and majority fragment portion are stored on a media having a higher latency than the memory. As previously described, the cache server computing device can utilize a service or transmit a request in the event that the storage location for the majority fragment portion is not known or if multiple storage locations exist. At decision block 508, a test is conducted to determine whether the transmission of the initialization fragment portion is complete. If the transmission is not complete, the cache server computing device continues to transmit the initialization fragment portion from memory. Additionally, if the majority fragment portion is not downloaded, the cache server computing device continues to download the majority fragment portion (or intermediate fragment portions).

[0041] Upon completion of the transmission of the initialization fragment portions, at block 510, the cache server computing device then begins transmission of the remaining fragment portions (including all intermediary and majority fragments) over the

network to the requesting client computing device 102. At block 514, the cache server computing device can then delete any intermediary or majority fragment portions that were stored in memory as part of the transmission process. In an illustrative embodiment, the cache server computing device may explicitly delete the intermediary or majority fragment portions from memory. In another embodiment, the cache server computing device may mark the memory used to store the intermediary or majority fragment portions as available for overwriting. In still a further embodiment, the cache server computing device may lower the priority in a memory management algorithm for the memory used to store the intermediary or majority fragment portions. Additionally, block 514 may be optionally omitted. The routine ends at block 516.

[0042] In other embodiments, the commencement of transmission of the initialization fragment occurs immediately upon receipt of the request for the associated object, with the request for retrieval of the majority fragment occurring simultaneously with or immediately subsequent to the commencement of transmission of the initialization fragment.

[0043] With reference now to Figure 6, one embodiment of a routine 600 for managing storage of a resource at a cache server computing device will be described. At a block 602, the cache server computing device receives a request for storage of an object. At block 604, the cache server computing device determines an initialization fragment and one or more remaining fragments associated with the object. Part of the process of making such determination includes determining the size of the initialization fragment. In one embodiment, the size of the initialization fragment is based on the latency associated with retrieving the one or more remaining fragments from other storage location as compared to the latency associated with transmitting (at the same time) the initialization fragment portion. As previously described, the size of the initialization chunk can be selected such that the retrieval of the intermediary and/or majority fragment portions is complete prior to completion of the transmission of the initialization fragment portion to the client computing device 102. Specifically, the size of the initialization fragment can be based on the throughput of the number of network packets that can be sent during the average or maximum latency of the storage media as compared to the fastest possible throughput of packets to a client computing device 102. It will be appreciated by one skilled in the relevant art that the size of the initialization fragment can be determined in a

number of ways, such as statically or dynamically or otherwise, and based on a variety of factors, including those described above, as well as others such as the encoding rate of the object.

[0044] In another embodiment, additional cache management methodologies may be integrated into the consideration size of the initialization fragment portion. In one example, the size of the initialization fragment can be based on a frequency of a request for the object. For example, an object that is frequently requested could have a larger initialization fragment size so as to reduce the number of IO requests required to serve the one or more remaining fragments from the higher latency storage media. In another example, the size of the initialization fragment is based on a frequency of a request for another object related to the requested object. .

[0045] With reference again to Figure 6, at block 606, the cache server computing device then stores the initialization fragment in memory of the cache server computing device. Alternatively, the cache server computing device may simply associate the initialization fragment with a storage location and store the association, especially for example where another computing device controls the storage of the initialization fragment. At block 608, the cache server computing device stores the one or more remaining fragments in one or more storage locations. Again, alternatively, the cache server computing device may simply associate the one or more remaining fragments with the selected one or more storage locations and store the association.

[0046] In another illustrative embodiment, the size of the majority fragment is the whole file size of the object. In accordance with this embodiment, the cache server component can receive the entire object as the intermediary and/or majority fragment portion and filter the fragments that have not yet been transmitted. Alternatively, the cache server computing device may request only portions of the majority file from its storage location. However, by storing the entire object as the majority and/or intermediary fragments, the size of the initialization fragment portion may be dynamically modified without requiring a corresponding modification to the other fragment portions. Alternatively, in another embodiment, the size of the intermediate and majority fragments each correspond such that all the fragments sum up to the whole file size of the object.

At decision block 610, the cache server computing device determines whether the initialization fragment needs to be updated. There are a number of ways in which such

determination may be made. In a few illustrative examples, the decision block 610 can be based on a determination of a latency associated with retrieval of the majority fragment, a frequency of a request for the object, and/or a frequency of a request for another object related to the requested object. In one embodiment, the decision block 610 includes a specific determination as to a new file size for the initialization fragment. The new file size can be determined in similar ways and based on similar factors presented above for determining the initial file size of the initialization fragment.

At block 612, if a determination is made that an update is needed, the cache server computing device stores an incremental fragment in the memory of the cache server computing device to supplement the initialization fragment. In one embodiment, the size of the incremental fragment and the original initialization fragment together correspond to the newly determined file size for the initialization fragment. Alternatively, in another embodiment, if an update is needed, a new initialization fragment of the appropriate size, which is determined through the update process, replaces the original initialization fragment in the memory. The routine ends at block 614. One skilled in the relevant art will appreciate that portions of routine 600 may be continuously (such as blocks 610 and 612) may be continuously implemented to update the fragment portions.

[0047] It will be appreciated by those skilled in the art and others that all of the functions described in this disclosure may be embodied in software executed by one or more processors of the disclosed components and mobile communication devices. The software may be persistently stored in any type of non-volatile storage.

[0048] Conditional language, such as, among others, "can," "could," "might," or "may," unless specifically stated otherwise, or otherwise understood within the context as used, is generally intended to convey that certain embodiments include, while other embodiments do not include, certain features, elements and/or steps. Thus, such conditional language is not generally intended to imply that features, elements and/or steps are in any way required for one or more embodiments or that one or more embodiments necessarily include logic for deciding, with or without user input or prompting, whether these features, elements and/or steps are included or are to be performed in any particular embodiment.

[0049] Any process descriptions, elements, or blocks in the flow diagrams described herein and/or depicted in the attached figures should be understood as

potentially representing modules, segments, or portions of code which include one or more executable instructions for implementing specific logical functions or steps in the process. Alternate implementations are included within the scope of the embodiments described herein in which elements or functions may be deleted, executed out of order from that shown or discussed, including substantially concurrently or in reverse order, depending on the functionality involved, as would be understood by those skilled in the art. It will further be appreciated that the data and/or components described above may be stored on a computer-readable medium and loaded into memory of the computing device using a drive mechanism associated with a computer readable storing the computer executable components such as a CD-ROM, DVD-ROM, or network interface further, the component and/or data can be included in a single device or distributed in any manner.

[0050] Various aspects of the described embodiments can be further understood with respect to the following clauses:

[0051] Clause 1. A computer-implemented method comprising:
receiving, at a cache component, an object for storage;
segmenting the object into an initialization fragment for storage in memory and a majority fragment for storage in a media having higher latency than the memory, wherein the size of the initialization fragment is based on a latency associated with retrieval of the majority fragment;
receiving a request for the object at a cache component;
transmitting the initialization fragment of the object from the memory;
retrieving the majority fragment of the object; and
transmitting the majority fragment without retaining the majority fragment in the memory for subsequent processing.

[0058] Clause 2. A computer-implemented method comprising:
receiving a request for an object at a cache component;
transmitting an initialization fragment of the object from memory;
retrieving a majority fragment of the object; and
transmitting the majority fragment without retaining the majority fragment in memory for subsequent processing.

[0063] Clause 3. The computer-implemented method of Clause 2, wherein retrieving the majority fragment comprises retrieving the majority fragment from local disk memory of the cache component.

[0064] Clause 4. The computer-implemented method of Clause 2, wherein retrieving the majority fragment comprises retrieving the majority fragment from disk memory of another cache component.

[0065] Clause 5. The computer-implemented method of Clause 2, wherein retrieving the majority fragment comprises retrieving the majority fragment from network-based memory.

[0066] Clause 6. The computer-implemented method of Clause 2, wherein retrieving the majority fragment comprises retrieving the majority fragment from an origin server.

[0067] Clause 7. The computer-implemented method of Clause 2 further comprising:

retrieving an intermediate fragment of the object; and

providing the intermediate fragment without retaining the intermediate fragment in memory for subsequent processing.

[0070] Clause 8. The computer-implemented method of Clause 2 further comprising:

determining if the initialization fragment should be updated;

if so, storing an incremental fragment in the memory to supplement the initialization fragment.

[0073] Clause 9. The computer-implemented method of Clause 4, wherein determining if the initialization fragment should be updated is based on a determination of a latency associated with retrieval of the majority fragment.

[0074] Clause 10. The computer-implemented method of Clause 4, wherein determining if the initialization fragment should be updated is based on a frequency of a request for the object, such that a change in the size of an incremental fragment is based on a change in request frequency.

[0075] Clause 11. The computer-implemented method of Clause 4, wherein determining if the initialization fragment should be updated is based on a frequency of a

request for another object related to the requested object, such that a change in the size of an incremental fragment is based on a change in request frequency for the related object.

[0076] Clause 12. The computer-implemented method of Clause 2 further comprising:

determining a new file size for the initialization fragment; and
storing an incremental fragment in the memory, wherein the initialization fragment and the increment fragment corresponds to the new file size.

[0079] Clause 13. The computer-implemented method of Clause 2, wherein the majority fragment corresponds to a whole file size of the object.

[0080] Clause 14. A computer system having a memory, wherein the memory stores an initialization fragment, the computer system for implementing the method of Clause 2.

[0081] Clause 15. A computer-implemented method comprising:
receiving, at a cache component, an object for storage; and
segmenting the object into an initialization fragment for storage in memory and one or more remaining fragments for storage in a media having higher latency than the memory, wherein the size of the initialization fragment is based on a latency associated with retrieval of the one or more remaining fragments.

[0084] Clause 16. The computer-implemented method of Clause 8, wherein the one or more remaining fragments corresponds to a majority fragment, wherein the majority fragment corresponds to a whole file size of the object.

[0085] Clause 17. The computer-implemented method of Clause 8, wherein the one or more remaining fragments correspond to an intermediate fragment and a majority fragment, wherein the intermediate fragment and majority fragment each correspond to a whole file size of the object.

[0086] Clause 18. The computer-implemented method of Clause 8, wherein the one or more remaining fragments are not retained in the memory for subsequent processing.

[0087] Clause 19. A computer system having a memory, wherein the memory stores an initialization fragment, the computer system for implementing the method of Clause 8.

[0088] Clause 20. A computer-implemented method comprising:
receiving, at a cache component, an object for storage; and
segmenting the object into an initialization fragment for storage in memory
and one or more remaining fragments for storage in a media having higher latency than
the memory, wherein the size of the initialization fragment is based on a frequency of a
request for the object.

[0091] Clause 21. The computer-implemented method of Clause 20, wherein
the one or more remaining fragments corresponds to a majority fragment, wherein the
majority fragment corresponds to a whole file size of the object.

[0092] Clause 22. The computer-implemented method of Clause 20, wherein
the one or more remaining fragments correspond to an intermediate fragment and a
majority fragment, wherein the intermediate fragment and majority fragment each
correspond to a whole file size of the object.

[0093] Clause 23. The computer-implemented method of Clause 20,
wherein the one or more remaining fragments are not retained in the memory for
subsequent processing.

[0094] Clause 24. A computer system having a memory, wherein the memory
stores an initialization fragment, the computer system for implementing the method of
Clause 20.

[0095] Clause 25. A computer-implemented method comprising:
receiving, at a cache component, an object for storage; and
segmenting the object into an initialization fragment for storage in memory
and a majority fragment for storage in a media having higher latency than the memory,
wherein the size of the initialization fragment is based on a frequency of a request for
another object related to the requested object.

[0098] Clause 26. The computer-implemented method of Clause 10, wherein
the majority fragment corresponds to a whole file size of the object.

[0099] Clause 27. The computer-implemented method of Clause 10, wherein
the one or more remaining fragments are not retained in the memory for subsequent
processing.

[0100] Clause 28. A computer system having a memory, wherein the memory stores an initialization fragment, the computer system for implementing the method of Clause 10.

[0101] Clause 29. A computer-implemented method comprising:
receiving, at a cache component, an object for storage;
determining an initialization fragment and a majority fragment for the object;
associating a storage location in memory with the initialization fragment;
associating a storage location with the majority fragment, wherein the associated storage location of the majority fragment corresponds to a media having higher latency than the memory;
determining if the initialization fragment should be updated;
if so, associating a storage location in the memory with an incremental fragment, wherein the incremental fragment supplements the initialization fragment.

[0108] Clause 30. The computer-implemented method of Clause 13, wherein determining if the initialization fragment should be updated is based on a determination of a latency associated with retrieval of the majority fragment.

[0109] Clause 31. The computer-implemented method of Clause 13, wherein determining if the initialization fragment should be updated is based on a frequency of a request for the object, such that a change in the size of an incremental fragment is based on a change in request frequency.

[0110] Clause 32. The computer-implemented method of Clause 13, wherein determining if the initialization fragment should be updated is based on a frequency of a request for another object related to the requested object, such that a change in the size of an incremental fragment is based on a change in request frequency for the related object.

[0111] Clause 33. The computer-implemented method of Clause 13, wherein determining if the initialization fragment should be updated comprises determining a new file size for the initialization fragment, wherein the initialization fragment and the increment fragment corresponds to the new file size.

[0112] Clause 34. The computer-implemented method of Clause 13, wherein the majority fragment corresponds to a whole file size of the object.

[0113] Clause 35. A computer system having a memory, wherein the memory stores an initialization fragment, the computer system for implementing the method of Clause 13.

[0114] Accordingly, general purpose computing devices may be configured to implement the processes, algorithms and methodology of the present disclosure with the processing and/or execution of the various data and/or components described above.

[0115] It should be emphasized that many variations and modifications may be made to the above-described embodiments, the elements of which are to be understood as being among other acceptable examples. All such modifications and variations are intended to be included herein within the scope of this disclosure and protected by the following claims.

WHAT IS CLAIMED IS:

1. A computer-implemented method comprising:
receiving, at a cache component, an object for storage;
segmenting the object into an initialization fragment for storage in memory and a majority fragment for storage in a media having higher latency than the memory, wherein the size of the initialization fragment is based on a latency associated with retrieval of the majority fragment;
receiving a request for the object at a cache component;
transmitting the initialization fragment of the object from the memory;
retrieving the majority fragment of the object; and
transmitting the majority fragment without retaining the majority fragment in the memory for subsequent processing.
2. A computer-implemented method comprising:
receiving a request for an object at a cache component;
transmitting an initialization fragment of the object from memory;
retrieving a majority fragment of the object; and
transmitting the majority fragment without retaining the majority fragment in memory for subsequent processing.
3. The computer-implemented method of Claim 2 further comprising:
retrieving an intermediate fragment of the object; and
providing the intermediate fragment without retaining the intermediate fragment in memory for subsequent processing.
4. The computer-implemented method of Claim 2 further comprising:
determining if the initialization fragment should be updated;
if so, storing an incremental fragment in the memory to supplement the initialization fragment.
5. The computer-implemented method of Claim 4, wherein determining if the initialization fragment should be updated is based on a frequency of a request for the object, such that a change in the size of an incremental fragment is based on a change in request frequency.
6. The computer-implemented method of Claim 4, wherein determining if the initialization fragment should be updated is based on a frequency of a request for another

object related to the requested object, such that a change in the size of an incremental fragment is based on a change in request frequency for the related object.

7. A computer system having a memory, wherein the memory stores an initialization fragment, the computer system for implementing the method of Claim 2.

8. A computer-implemented method comprising:

receiving, at a cache component, an object for storage; and

segmenting the object into an initialization fragment for storage in memory and one or more remaining fragments for storage in a media having higher latency than the memory, wherein the size of the initialization fragment is based on a latency associated with retrieval of the one or more remaining fragments.

9. The computer-implemented method of Claim 8, wherein the one or more remaining fragments correspond to an intermediate fragment and a majority fragment, wherein the intermediate fragment and majority fragment each correspond to a whole file size of the object.

10. A computer-implemented method comprising:

receiving, at a cache component, an object for storage; and

segmenting the object into an initialization fragment for storage in memory and a majority fragment for storage in a media having higher latency than the memory, wherein the size of the initialization fragment is based on a frequency of a request for another object related to the requested object.

11. The computer-implemented method of Claim 10, wherein the majority fragment corresponds to a whole file size of the object.

12. The computer-implemented method of Claim 10, wherein the one or more remaining fragments are not retained in the memory for subsequent processing.

13. A computer-implemented method comprising:

receiving, at a cache component, an object for storage;

determining an initialization fragment and a majority fragment for the object;

associating a storage location in memory with the initialization fragment;

associating a storage location with the majority fragment, wherein the associated storage location of the majority fragment corresponds to a media having higher latency than the memory;

determining if the initialization fragment should be updated;

if so, associating a storage location in the memory with an incremental fragment, wherein the incremental fragment supplements the initialization fragment.

14. The computer-implemented method of Claim 13, wherein determining if the initialization fragment should be updated is based on a frequency of a request for the object, such that a change in the size of an incremental fragment is based on a change in request frequency.

15. The computer-implemented method of Claim 13, wherein determining if the initialization fragment should be updated comprises determining a new file size for the initialization fragment, wherein the initialization fragment and the increment fragment corresponds to the new file size.