



- (51) International Patent Classification:
G06F 12/12 (2006.01) H04L 29/08 (2006.01)
G06F 12/08 (2006.01)
- (21) International Application Number:
PCT/US2016/021421
- (22) International Filing Date:
9 March 2016 (09.03.2016)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
14/669,408 26 March 2015 (26.03.2015) US
- (71) Applicant: ALCATEL LUCENT [FR/FR]; 148/152
Route de la Reine, 92100 Boulogne-Billancourt (FR).
- (72) Inventor: AKHTAR, Shahid; 601 Data Drive, Plano, TX
75075-7839 (US).
- (74) Agent: DESAI, Niraj, A.; Alcatel-Lucent USA Inc., Att:
Docket Administrator - Room 3B-212F, 600-700 Mountain
Avenue, Murray Hill, NJ 07974-0636 (US).

- (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))



WO 2016/153779 A1

(54) Title: HIERARCHICAL COST BASED CACHING FOR ONLINE MEDIA

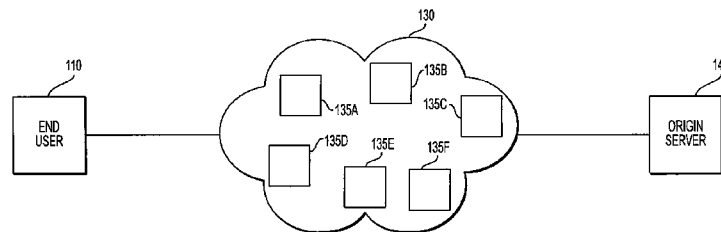


FIG. 1

(57) Abstract: A method of operating a first cache device may include receiving, at the first cache device, a request to send a first asset to a second device; determining whether the first asset is stored at the first cache device; and when the determining determines that the first asset is not stored at the first cache device, obtaining, at the first cache device, the first asset, comparing, at the first cache device, a moving average of a marginal value of the first asset with respect to the first cache device and a characteristic marginal value of the first cache device, calculating a cost associated with the first asset, selectively storing the first asset at the first cache device based on the comparison, and sending the obtained first asset and the calculated cost to the second device.

HIERARCHICAL COST BASED CACHING FOR ONLINE MEDIA

CROSS-REFERENCE TO RELATED APPLICATION

This application claims the benefit of provisional U.S. Application No. 62/064,631 filed on
5 October 16, 2014 and U.S. Application No. 14/577,039 filed on December 19, 2014, the
entire disclosure of both of which is hereby incorporated by reference.

BACKGROUND

1. Field

10 Example embodiments relate generally to caching media, and particularly to providing
hierarchical caching for online media.

2. Related Art

Online video is presently the largest component, or one of the largest components, of internet
15 traffic. Caching such video adds additional requirements on traditional algorithms. For many
conventional algorithms, hit-rate has traditionally been the key performance metric used to
compare such algorithms. Increasingly newer caches use SSD memory for storage due to
SSD memory's access speed and reliability. A lifetime of SSD memory may have a limited
20 number of write cycles available. Replacement rate, or the number of write cycles per request
is another key performance metric that may be of interest with such memory. According to
some online video usage data, a significant part of the online video is used to rebroadcast
recent TV shows which are initially quite popular but rapidly drop in popularity. Measuring
the performance of caching algorithms with such rapid changes is another new performance
metric. Typically online caching stores the video in small chunks of typically between 2 and
25 10 seconds. Further each video may be encoded into multiple video quality levels further
increasing the number of chunks per video. In all, online video can increase the number of
files that can be requested by end users in a caching system by a thousand fold or more.

Many conventional caching algorithms proposed in research over the last 30 years have not
been implemented in commercial caches. For example, prominent examples of implemented
30 schemes include simple schemes like the least recently used (LRU) and a few others like
greedy dual size frequency (GDSF) and least frequently used with dynamic aging (LFU-
DA).

Further, a size of a cache may be finite. Therefore, storage of a new asset may require the

eviction of a stored asset. However, when determining which stored asset to evict, many conventional caching algorithms do not factor in the cost associated with re-obtaining each of the stored assets.

5

SUMMARY

Some example embodiments relate to a method of operating a cache device.

In some example embodiments, the method includes receiving, at the first cache device, a request to send a first asset to a second device; determining whether the first asset is stored at the first cache device; and when the determining determines that the first asset is not stored at the first cache device, obtaining, at the first cache device, the first asset, calculating a cost associated with the first asset, selectively storing the first asset at the first cache device based on a moving average of a marginal value of the first asset with respect to the first cache device and a characteristic marginal value of the first cache device, the characteristic marginal value of the first cache device representing a cost conscious average period of time between receipt of last requests for, and eviction of, assets cached at the first cache, and sending the obtained first asset and the calculated cost to the second device.

In some example embodiments, the first cache is one of a plurality of cache devices of a content distribution network divided into a plurality of hierarchy levels, and the method further includes, for each of the plurality of cache devices, determining a moving average of the marginal value of a requested asset, when a request for the requested asset is received by a respective one of the plurality of cache devices, and determining a characteristic marginal value associated with a respective one of the plurality of cache devices, when the cache device evicts an asset.

In some example embodiments, the obtaining obtains the first asset and an obtained cost associated therewith from a higher level of the plurality of hierarchy levels, and the calculating the cost includes adding a marginal cost to the obtained cost if the selectively storing does not store the first asset.

In some example embodiments, the method further includes determining the characteristic marginal value of the first cache device by, initializing the characteristic marginal value of the first cache device as a value which is higher than the initialized value of the moving average of the marginal value (MU) of the first asset with respect to the first cache device, and updating the characteristic marginal value of the first cache device based on exponentially weighted moving average of periods of time between receipt of last requests

for, and eviction of, assets cached at the first cache device and the cost associated with the assets, in accordance with a least recently used cache eviction policy.

In some example embodiments, the updating the characteristic marginal value of the first cache device further includes, gently increasing the characteristic marginal value of the first cache device, when the first cache evicts an asset.

In some example embodiments, the method includes determining the moving average of the marginal value of the first asset with respect to the first cache device by, initializing the moving average of the marginal value of the first asset with respect to first cache device as a value lower than the initialized value of the characteristic marginal value of the first cache device, and updating the moving average of the marginal value of the first asset with respect to the first cache device based on exponentially weighted moving average of periods of time between consecutive receipts of requests for the first asset and the cost associated with the first asset, at the first cache device, the requests requesting the first cache device to send the first asset to another device.

In some example embodiments, the selectively storing includes, assigning the first asset to a first database, when an initial request for the first asset is received at the first cache device; and when a second request is received consecutively with respect to the initial request, determining an inter-arrival time of the first asset based on times at which the initial and second requests were received at the first cache device, and if a number of assets stored in an inter-arrival database is greater than or equal to a threshold then, determining a second asset to remove from an inter-arrival database based on which of a plurality of assets stored in the inter-arrival database has a largest marginal value associated therewith that decreases as the cost associated with the second asset increases, the inter-arrival time database storing arrival times of requests corresponding to assets, the inter-arrival time database being different than the first database, and removing the second asset from the inter-arrival database; and assigning the first asset to the inter-arrival time database.

In some example embodiments, the method further includes determining the marginal value of the assets by dividing a last requested time of each of the plurality of assets by the cost associated therewith.

According to one or more example embodiments, a first cache device may include a processor and a memory.

In some example embodiments, the memory may contains computer readable code that, when executed by the processor, configures the processor to, receive a request to send a first asset

to a second device, determine whether the first asset is stored at the first cache device, and when the processor determines that the first asset is not stored at the first cache device, the processor is configured to, obtain, at the first cache device, the first asset, calculate a cost associated with the first asset, selectively store the first asset at the first cache device based
5 on a moving average of a marginal value of the first asset with respect to the first cache device and a characteristic marginal value of the first cache device, the characteristic marginal value of the first cache device representing a cost conscious average period of time between receipt of last requests for, and eviction of, assets cached at the first cache, and send the obtained first asset and the calculated cost to the second device.

10 According to one or more example embodiments, a method of operating a content distribution network, the content distribution network including a plurality of first cache devices divided into a plurality of hierarchy levels, each of the plurality of first caches devices including a processor and a memory.

In some example embodiments, the memory may contain computer readable code that, when
15 executed by a processor, configures the processor of a respective first cache device to, receive a request to send a first asset to a second device, determine whether the first asset is stored at the first cache device, and when the processor determines that the first asset is not stored at the first cache device, obtain, at the first cache device, the first asset, calculate a cost associated with the first asset, selectively store the first asset at the first cache device based on
20 a moving average of a marginal value of the first asset with respect to the first cache device and a characteristic marginal value of the first cache device, the characteristic marginal value of the first cache device representing a cost conscious average period of time between receipt of last requests for, and eviction of, assets cached at the first cache, and send the obtained first asset to the second device.

25

BRIEF DESCRIPTION OF THE DRAWINGS

At least some example embodiments will become more fully understood from the detailed description provided below and the accompanying drawings, wherein like elements are represented by like reference numerals, which are given by way of illustration only and thus
30 are not limiting of example embodiments and wherein:

FIG. 1 is a diagram illustrating a portion of a data network according some example embodiments;

FIG. 2A is a diagram illustrating the organization of a hierarchical cache system;

FIG. 2B is a diagram illustrating an example structure of a network element according to example embodiments;

FIG. 3 is a flow chart illustrating an example method of operating a network element to perform hierarchical caching of online media according to some example embodiments; and

5 FIG. 4 is a flow chart illustrating an example method of operating a network element to perform hierarchical caching of online media according to other example embodiments.

DETAILED DESCRIPTION OF EXAMPLE EMBODIMENTS

Various example embodiments will now be described more fully with reference to the
10 accompanying drawings in which some example embodiments are shown.

Detailed illustrative embodiments are disclosed herein. However, specific structural and functional details disclosed herein are merely representative for purposes of describing at least some example embodiments. Example embodiments may, however, be embodied in many alternate forms and should not be construed as limited to only the embodiments set
15 forth herein.

Accordingly, while example embodiments are capable of various modifications and alternative forms, embodiments thereof are shown by way of example in the drawings and will herein be described in detail. It should be understood, however, that there is no intent to limit example embodiments to the particular forms disclosed, but on the contrary, example
20 embodiments are to cover all modifications, equivalents, and alternatives falling within the scope of example embodiments. Like numbers refer to like elements throughout the description of the figures. As used herein, the term "and/or" includes any and all combinations of one or more of the associated listed items.

It will be understood that when an element is referred to as being "connected" or "coupled" to another element, it can be directly connected or coupled to the other element or intervening elements may be present. In contrast, when an element is referred to as being "directly connected" or "directly coupled" to another element, there are no intervening elements present. Other words used to describe the relationship between elements should be interpreted in a like fashion (e.g., "between" versus "directly between", "adjacent" versus
30 "directly adjacent", etc.).

The terminology used herein is for the purpose of describing particular embodiments only and is not intended to be limiting of example embodiments. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context

clearly indicates otherwise. It will be further understood that the terms "comprises", "comprising", "includes" and/or "including", when used herein, specify the presence of stated features, integers, steps, operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps, operations, elements, components, and/or groups thereof.

It should also be noted that in some alternative implementations, the functions/acts noted may occur out of the order noted in the figures. For example, two figures shown in succession may in fact be executed substantially concurrently or may sometimes be executed in the reverse order, depending upon the functionality/acts involved.

Example embodiments are discussed herein as being implemented in a suitable computing environment. Although not required, example embodiments will be described in the general context of computer-executable instructions, such as program modules or functional processes, being executed by one or more computer processors or CPUs. Generally, program modules or functional processes include routines, programs, objects, components, data structures, etc. that performs particular tasks or implement particular abstract data types.

The program modules and functional processes discussed herein may be implemented using existing hardware in existing communication networks. For example, program modules and functional processes discussed herein may be implemented using existing hardware at existing network elements or control nodes (e.g., an eNB shown in FIG. 1). Such existing hardware may include one or more digital signal processors (DSPs), application-specific-integrated-circuits (ASICs), field programmable gate arrays (FPGAs) computers or the like.

In the following description, illustrative embodiments will be described with reference to acts and symbolic representations of operations (e.g., in the form of flowcharts) that are performed by one or more processors, unless indicated otherwise. As such, it will be understood that such acts and operations, which are at times referred to as being computer-executed, include the manipulation by the processor of electrical signals representing data in a structured form. This manipulation transforms the data or maintains it at locations in the memory system of the computer, which reconfigures or otherwise alters the operation of the computer in a manner well understood by those skilled in the art.

Example Network Architecture

FIG. 1 illustrates a portion of a data network 100 according to one or more example embodiments. Communications network 100 includes an end user 110, a content distribution

network (CDN) 130 and an origin server 140.

The end user 110 may be embodied by, for example, an electronic user device. An electronic device may include a mobile device, smart phone, laptop, tablet, or a personal computer. The end user 110 is capable of receiving content stored at the origin server 140 via CDN 130.

5 The end user 110, the CDN 130 and the origin server 140 may be connected to each other through, for example, the internet.

The CDN 130 includes caches 135A~F. Caches 135A~F each include storage for storing media content. Caches 135A~F may be embodied, together, in groups, or individually in, for example, servers, routers, or wireless communications network components including, for
10 example, base stations (BSs), evolved node Bs (eNBs), or a radio network controllers (RNCs). Though CDN 130 is illustrated as only including six caches 135A~F, the CDN 130 may include any number of caches. Further, in the example illustrated in FIG. 1, the origin server 140 is separate from the CDN 130. However, according to one or more example embodiments, the origin server 140 may be considered part of the CDN 130.

15 The origin server 140 is a server that provides content in response to content requests. For example, the origin server 140 may store content corresponding to one or more videos which may be requested by the end user 110 for streaming. In this case, the origin server 140 may receive content requests associated with the particular videos, for example from a cache within the CDN 130, and the origin server 140 may respond to the requests by providing the
20 requested content. Though, for the purpose of simplicity, only one origin server 140 is illustrated, data network 100 may include any number of origin servers.

The caches in the CDN 130 may be organized in a hierarchical cache structure. FIG. 2A is a diagram illustrating an example organization of a hierarchical cache system. As is illustrated in FIG. 2A, caches in a hierarchical cache system may be organized in a tree structure. Each
25 cache in FIG. 2A has an index 'j' and a hierarchy level 'k' denoted by the coordinates (j,k). Accordingly, the lowest hierarchy level, k=0, includes four caches, cache (0,0), cache (1,0), cache (2,0), and cache (3,0). The next highest hierarchy level, k=1, includes 2 caches, cache (0,1), and cache (1,1). The highest hierarchy level, k=2, includes 1 cache, cache (0,2). Cache (0,2) may be the parent of cache (0,1) and cache (1,1). Cache (0,1) may be the parent of
30 cache (0,0) and cache (1,0). Cache (1,1) may be the parent of cache (2,0) and cache (3,0). Every cache that has a parent cache is a child cache with respect to the parent cache. Further, caches in a hierarchical cache system that are not parent caches may be referred to as 'leaf' caches. For example, in the tree structure shown in FIG. 2A, the caches of hierarchy level

k=0 (i.e., cache (0,0), cache (1,0), cache (2,0), and cache (3,0)) are examples of leaf caches. When a child cache in a hierarchical cache system experiences a cache miss with respect to a requested asset, the child cache may communicate with the parent cache of the child cache to determine whether or not the parent cache has the requested asset. If the parent cache does not have the requested asset, the parent cache may then communicate with a higher level cache or an origin server to obtain the requested asset, before providing the requested asset to the child cache.

As used herein, the term “asset” refers to data that may be stored in a cache or provided by an origin server, and may be requested by a user. For example, with respect to online video, an example of an asset is a 2-10 second chunk of video data stored at an origin server, that may be requested by a user and may be stored at one or more caches.

According to one or more example embodiments, the caches in the CDN 130 may be organized in the hierarchical cache structure shown in FIG. 2A. For example the caches 135A-F in FIG. 1 may correspond, respectively, to cache (0,2), cache (0,1), cache (1,1), cache (0,0), cache (1,0), and cache (2,0).

An example structure of the network elements of data network 100 will now be discussed below with reference to FIG. 2B.

Example Network Element Structure

FIG. 2B is a diagram illustrating an example structure of a network element 251. According to example embodiments, any or all network elements in the data network 100, including for example the end user 110, the caches 135A-F, and the origin server 140, may have the same structure and operation described below with respect to network element 251.

Referring to FIG. 2B, the network element 251 may include, for example, a data bus 259, a transmitting unit 252, a receiving unit 254, a memory unit 256, and a processing unit 258.

The transmitting unit 252, receiving unit 254, memory unit 256, and processing unit 258 may send data to and/or receive data from one another using the data bus 259.

The transmitting unit 252 is a device that includes hardware and any necessary software for transmitting signals including, for example, control signals or data signals via one or more wired and/or wireless connections to other network elements in data network 100.

The receiving unit 254 is a device that includes hardware and any necessary software for receiving wireless signals including, for example, control signals or data signals via one or more wired and/or wireless connections to other network elements in the data network 100.

The memory unit 256 may be any device capable of storing data including magnetic storage, flash storage, etc.

The processing unit 258 may be any device capable of processing data including, for example, a processor.

5 According to at least one example embodiment, any operations described herein, for example with reference to any of FIGS. 1-3, as being performed by any one of a user (e.g., the end user 110), a cache (e.g., caches 135A-F), and a server (e.g., the origin server 140) may be performed by an electronic device having the structure of the network element 251 illustrated in FIG. 2B. For example, according to at least one example embodiment, the network element
10 251 may be programmed, in terms of software and/or hardware, to perform any or all of the functions described herein as being performed by a user, a cache, or server. Consequently, each of the end user 110, the caches and the servers described herein may be embodied as special purpose computers.

Examples of the network element 251 being programmed, in terms of software, to perform
15 any or all of the functions described herein as being performed by any of a user, a cache, or server described herein will now be discussed below. For example, the memory unit 256 may store a program including executable instructions corresponding to any or all of the operations described herein with reference to FIGS. 1-3 as being performed by a user, a cache, or server. According to at least one example embodiment, additionally or alternatively
20 to being stored in the memory unit 256, the executable instructions may be stored in a computer-readable medium including, for example, an optical disc, flash drive, SD card, etc., and the network element 251 may include hardware for reading data stored on the computer readable-medium. Further, the processing unit 258 may be a processor configured to perform any or all of the operations described herein with reference to FIGS. 1-3 as being performed
25 by a user, a cache, or server, for example, by reading and executing the executable instructions stored in at least one of the memory unit 256 and a computer readable storage medium loaded into hardware included in the network element 251 for reading computer-readable mediums.

Examples of the network element 251 being programmed, in terms of hardware, to perform
30 any or all of the functions described herein as being performed by a user, a cache, or server will now be discussed below. Additionally or alternatively to executable instructions corresponding to the functions described with reference to FIGS. 1-3 as being performed by a user, a cache, or server being stored in a memory unit or a computer-readable medium as is

discussed above, the processing unit 258 may include a circuit that has a structural design dedicated to performing any or all of the operations described herein with reference to FIGS. 1-3 as being performed by a user, a cache, or server. For example, the above-referenced circuit included in the processing unit 258 may be a FPGA or ASIC physically programmed to perform any or all of the operations described with reference to FIGS. 1-3 as being performed by a user, a cache, or server.

An overview of hierarchical caching according to one or more example embodiments will now be discussed below.

10 Overview of Hierarchical Caching According to One or More Example Embodiments

With respect to many conventional caching algorithms, some of the impediments to practical use may have been a perceived complexity of implementation of the algorithms which can impede throughput performance and an expectation that the value of additional caching performance gains such as increased hit-rate may be outweighed by factors such as ability to adapt to asset popularity changes.

One or more example embodiments use new caching algorithms for online video. The new caching algorithms have low implementation complexity; and have improved hit-rate and replacement rate. The new caching algorithms have been evaluated based on simulations that involve typical popularity of assets and realistic changes using video on-demand (VoD) statistics. Some studies have indicated that many assets are quite popular in the beginning of their lifecycle, but drop in popularity at an exponential rate and are a fraction of their popularity within days after their introduction. Based on these, not only hit-rate, but also byte bit-rate and replacement rate are used to evaluate caching algorithms according to one or more example embodiments. Though one or more example embodiments of caching algorithms are described herein with reference to caching online video, one or more example embodiments described herein for caching online video may also apply to other types of media transferred through communications networks like the internet including, for example, audio, pictures, video games, and 3-D objects and models.

30 Example Method of Providing Hierarchical Caching According to One or More Example Embodiments

A method of providing hierarchical caching according to one or more example embodiments will now be discussed with respect to FIG. 3 and with reference to examples in which online

video data is being cached by the CDN 130 of FIG. 1. According to one or more example embodiments, the method of providing hierarchical caching illustrated in FIG. 3 is based on a hierarchical cache, like that shown in FIG. 2A, rather than a horizontally cooperative cache, because hierarchical caches may be more efficient for operator networks.

5 According to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that works in a local manner in a given cache without the need for global information. Further, according to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that is an $O(1)$ algorithm with respect to the number of assets or nodes.

10 According to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that is relatively quick to respond to changes in popularity when previously very popular assets drop their value quickly.

According to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that provides improved hit-rate performance.

15 According to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that does not use the size of an asset in the decision to evict as this may cause undesirable behaviour for online video.

According to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that runs at each cache in a given hierarchy independent of the

20 algorithm in other caches. Each cache estimates the inter-arrival time of each asset locally.

Each cache also calculates its own characteristic time, which is defined as the average time an asset stays in cache before it is evicted. For example the characteristic time of a cache may be determined by the cache in accordance with known method using LRU, by determining the average of several eviction times corresponding to several assets where, for each asset, the

25 eviction time for that asset may be determined when the cache is about to evict an asset, as the difference between the current time and the time that asset was last requested. One or more example embodiments of the above-referenced caching algorithm will now be discussed in greater detail below.

30 According to one or more example embodiments, a method of providing hierarchical caching uses a caching algorithm that evicts assets in accordance with the known LRU scheme. For example, according to one or more example embodiments, once the caching algorithm determines an asset is to be evicted from a cache, the asset selected for eviction is the least recently used asset.

Table 1 below describes variables that are defined for each asset.

Table 1

i	Asset Number from 0 to N.
j	Cache number in a given level of hierarchy from 0 to P.
k	Cache hierarchy level from 0 to M, higher number is higher in hierarchy.
T_{ijk}	Average inter-arrival time of asset i for cache j in hierarchy k .
P_{ijk}	Previous time asset i was seen in cache j in hierarchy k .

5 Table 2 below describes variables that are defined for each cache.

Table 2

TC_{jk}	Characteristic time of cache j in hierarchy k .
PL_{jk}	Last time this asset was seen in cache j in hierarchy k . This is part of the LRU function which leaves a timestamp for the last time request for an asset was seen in a cache.
TS	Current time.

Table 3 below describes variables that are defined for the whole cache hierarchy.

Table 3

W_{TC}	Weight for the exponential moving average for the characteristic time. Example values include 0.01 – 0.05.
W_{IA}	Weight for the exponential moving average for the inter-arrival times. Example values include 0.01 – 0.05.
GS	Gentle slope parameter on how quickly TC_{jk} is increased over time. Example values include 0.001 – 0.0001.

10

Table 4 below describes initial conditions that, according to one or more example embodiments, are set before the caching algorithm used by the method of providing hierarchical caching illustrated in FIG. 3 is run.

Table 4

P_{ijk}	Set all to the current time.
T_{ijk}	Set all to a large number, but smaller than the number used for initial value for the characteristic times TC_{jk} . This allows all assets to be cached until TC_{jk} is updated to a real value. An example would set this number 100 times higher than the largest expected inter-arrival time.
TC_{jk}	Set all to a large number, larger than the number for T_{ijk} . Typically set this at 10 times the T_{ijk} number.

15

Equation (1) below illustrates the manner in which TC_{jk} , the characteristic time value of a

cache (j,k), is calculated.

$$TC_{jk} = w_{TC} \times (TS - P_{ijk}) + (1 - w_{TC}) \times TC_{jk} \quad (1)$$

Equation (1) calculates the characteristic time TC_{jk} as an exponential moving average of times
 5 assets stay in cache (j,k) without being requested before being evicted from the cache (j,k),
 for example, in accordance with an LRU cache eviction policy. For example, the
 characteristic time TC_{jk} as an exponential moving average of times between receipt last
 requests for assets in cache (j,k) and eviction of the assets from cache (j,k). According to one
 or more example embodiments, in accordance with known LRU methods for operating
 10 caches, the value P_{ijk} is updated to the current time whenever a cache receives a request for
 the i^{th} asset by the LRU function itself.

Equation (2) below illustrates the manner in which the average the inter-arrival time of asset i
 for a cache (j,k), T_{ijk} , is calculated. According to one or more example embodiments, the
 value T_{ijk} may be calculated whenever an asset is requested from any cache, whether it is a
 15 leaf cache or a higher layer cache.

$$T_{ijk} = w_{IA} \times (TS - P_{ijk}) + (1 - w_{IA}) \times T_{ijk} \quad (2)$$

Equation (2) calculates an exponential moving average of the inter-arrival time of asset i
 20 using the weight w_{IA} .

According to one or more example embodiments, after calculating the value TC_{jk} in
 accordance with equation (1), the characteristic time TC_{jk} is gently increased so that the
 characteristic time TC_{jk} does not get stuck at a low number. The characteristic time TC_{jk} is
 gently increased in accordance with equation (3) below.

$$TC_{jk} = TC_{jk} + GS \times (TS - PL_{jk}) \quad (3)$$

According to one or more example embodiments, after gently increasing the value TC_{jk} in
 accordance with equation (3), the current value for PL_{jk} is set to the current time as is
 illustrated by equation (4) below.

$$PL_{jk} = TS \quad (4)$$

An example of a caching algorithm used by the method of providing hierarchical caching according to one or more example embodiments is described below in the form of pseudo code by algorithm 1. According to one or more example embodiments, algorithm 1 may be performed, individually, by each cache in the hierarchical tree structure of CDN 130. For example, every time a cache from among the caches of the CDN 130 receives a request for an asset i , the cache may perform Algorithm 1 to determine if the requested asset needs to be cached by the cache.

10	Algorithm 1
	Update asset counts T_{ijk} ; as defined earlier
	If (asset already in cache) begin
	Deliver asset i to client or lower layer cache
15	Update LRU Database (P_{ijk}) with timestamp for that asset
	end
	else if ($T_{ijk} < TC_{jk}$) begin
	Request asset i from upstream cache or origin
	Deliver asset i to client or lower layer cache
20	Store asset i in the local cache
	end
	else begin
	Request asset i from upstream cache or origin
	Deliver asset i - note here we do not store this asset
25	end

An example use of Algorithm 1 will be discussed in greater detail below with respect to FIG. 3. FIG. 3 is a flow chart illustrating an example method of operating a network element to perform hierarchical caching of online media. FIG. 3 will be explained from the perspective of cache 135B which, as is explained above with reference to FIG. 2A, corresponds to cache (0,1) in FIG 2A (i.e., the 0th cache in hierarchy level $k=1$).

Referring to FIG. 3, in step S305, a request for an asset is received at the cache. For example, in step S305, the cache 135B receives a request for an asset x . The cache 135B may receive the request for the asset x from one of the children caches of cache 135B (i.e., as is illustrated in FIG. 2A, the cache 135B is the parent to caches 135D and 135E, which correspond to cache (0,0) and cache (1,0) illustrated in FIG. 2A, respectively). The method shown in FIG. 3 will be explained with reference to an example where the cache 135B receives a request for

the asset x from the cache 135D. For example, the cache 135D may send a request for the asset x to cache 135B in response to receiving, at the cache 135D, a request for the asset x from the end user 110. Accordingly, when the method shown in FIG. 3 is performed by the cache 135B when receiving a request for an asset x , $i=x$, $j=0$, and $k=1$.

5 As is shown in Algorithm 1, upon receiving the request for the asset x in step S305, the cache 135B may calculate the value T_{ijk} using equation (2). Next the cache may set the value for P_{ijk} as the current time. Next, the cache 135B may proceed to step S310. Whenever the cache 135B evicts an asset, it may calculate the value TC_{jk} using equation (1) and then the cache 135B may gently increase the value TC_{jk} using equation (3).

10 In step S310, the cache determines whether or not the asset for which the request was received in step S305 is stored (i.e., cached) in the cache. For example, in step S310, the cache 135B may determine whether the asset x is already cached in the cache 135B. If the asset the asset x is currently stored in the cache 135B, then the cache 135B proceeds to step S315.

15 In step S315, the cache provides the asset requested in step S305 and updates the LRU database of the cache. For example, in step S315, the cache 135B may provide the asset x to the network element that requested the asset in step S310, the cache 135D. Further, in accordance with known LRU methods of operating a cache, the cache 135D may update an LRU database within the cache 135D (e.g., within the memory 256 of the cache 135D) that
20 stores timestamps indicating, respectively, times of the most recent uses of each the assets presently stored in the cache.

Returning to step S310, if the cache determines in step S310 that the asset requested in step S305 is not included in the cache, the cache proceeds to step S320. For example, if, in step S310, the cache 135B determines that the asset x is not currently stored in the cache 135B,
25 then the cache 135B proceeds to step S320.

In step S320, the cache retrieves the asset requested in step S305. For example, in step S320, the cache 135B may send a request for the asset x to the parent of the cache 135B, cache 135A. In response to the request sent by the cache 135B, the cache 135B may receive the asset x from the cache 135A. For example, the cache 135A may be storing the asset x
30 already, or the cache 135A may retrieve the asset x from the origin server 140 before providing the asset x to the cache 135B.

Next, in step S325, the cache performs a comparison based on the values TC_{jk} and T_{ijk} and determines whether or not to store the asset x based on the comparison. For example, in step

S325, the cache 135B may compare the value TC_{jk} to the value T_{ijk} in accordance with equation (5) below.

$$(T_{ijk} < TC_{jk}) \quad (5)$$

If the result of the comparison operation in step S325 is true, the cache 135B may proceed to step S335 before proceeding to step S330. In step S335, the cache 135B stores the asset x in the cache 135B. In step S330, the cache 135B provides the asset to the network element that requested the asset, cache 135D.

If the result of the comparison operation in step S325 is false, the cache 135B may proceed directly to step S330 without proceeding to step S335. Accordingly, if the result of the comparison operation in step S325 is false, the cache 135B may provide the asset x to the cache 135D without caching the asset x in the cache 135B. Thus, according to one or more example embodiments, caches in the CDN 130 may determine whether or not to cache a retrieved asset in accordance with the comparison defined by equation (5).

After providing the requested asset x in step S330, the cache 135B returns to step S305 to await receipt of the next asset request.

Though FIG. 3 is explained from the perspective of the cache 135B, the operations discussed above with respect to the cache 135B may be performed by any cache in the CDN 130. For example, all caches in the CDN may perform the operations discussed above with respect to the cache 135B concurrently.

A new database, which may be included in caches of the CDN 130 (e.g., within the memories 256 of the caches of the CDN 130) in addition to the LRU databases of the caches, will now be discussed below.

Large Asset Inter-arrival Time Database

As is discussed above, caches of the CDN 130 may include LRU databases in accordance with known LRU methods for operating caches. Further, each of the caches of the CDN 130 may also include a database for storing inter-arrival times of various assets for example, for calculating the average inter-arrival of asset i, T_{ijk} .

Given that the number of assets in a library can be very large, sometimes much larger than the number of assets that can be stored in a cache, it may be desirable to use more than one database to store the inter-arrival time statistics. For example, according to one or more example embodiments, the caches of the CDN 130 may include a main database of inter-arrival times and an additional database of inter-arrival times. In accordance with one or

more example embodiments, the main database of inter-arrival times for a cache does not contain inter-arrival times that are more than a few times the TC_{jk} value for that cache; for example 3 times. According to one or more example embodiments, the exact value above which inter-arrival times are not included in the main inter-arrival time data base may be based on the popularity distribution of assets stored at that cache.

The additional database may be a “one occurrence” database that is used to store assets which have seen only a single request so far and assets that are demoted from the main inter-arrival time database. For example, when a cache sees a second request for an asset, the cache may calculate an inter-arrival time based on the timings of the first and second requests, and the cache may place that asset into the main inter-arrival time database based on the newly calculated inter-arrival time. Further, upon placing asset which has the newly calculated inter-arrival time into the main database, the asset with the largest inter-arrival time in the main database may be removed from the main database. In order to return to the main database, the removed asset may need to enter the “one occurrence” database first, and then be promoted to the main inter-arrival database in the manner discussed above.

The additional database may be a “one occurrence” database that is used to store assets which have seen only a single request so far. When the cache sees a second request for an asset in the “one occurrence” database, the cache may calculate an inter-arrival time based on the timings of the first and second requests, then the cache may place that asset into the main inter-arrival time database based on the newly calculated inter-arrival time while also deleting that entry from the “one occurrence” database. Further, upon placing asset which has the newly calculated inter-arrival time into the main database, the asset that was the least recently used in the main database may be removed from the main database. Least recently used means the asset whose last request was the oldest among all assets. In order to return to the main database, the removed asset may need to enter the “one occurrence” database first, and then be promoted to the main inter-arrival database in the manner discussed above.

Table 5 below describes additional variables for the cost function that are defined for each asset.

Table 5

CV_{jk}	Characteristic marginal value of cache j in hierarchy k .
C_{ijk}	The current cost of asset i in cache j in hierarchy k
MV_{ijk}	The marginal value of asset i in cache j in hierarchy k
MU_{ijk}	The exponential moving averages of marginal value of asset i in cache j in hierarchy k

$C-out_{ijk}$	The cost of asset i in cache j in hierarchy k sent from origin or upstream cache
$C-in_{ijk}$	The cost of asset i in cache j in hierarchy k sent to downstream cache
$C-sto_{ijk}$	The marginal value of asset i in cache j in hierarchy k
TS	Current time.

As discussed below, in other example embodiments, the caches in the CDN 130 may perform Algorithm 2 when receiving a request for an asset.

```

5  Algorithm 2

Update asset counts  $T_{ijk}$ ; as defined earlier

If (asset already in cache) begin
10  Deliver asset  $i$  and cost  $C_{out,ijk}$  of asset  $i$  to client or lower layer cache,
    Update LRU Database ( $P_{ijk}$ ) with timestamp for that asset;
    end
else if ( $MU_{ijk} < CV_{jk}$ ) begin
15  Request asset  $i$  from upstream cache or origin
    Deliver asset  $i$  and cost  $C_{out,ijk}$  of asset  $i$  to client or lower layer cache
    Store asset  $i$  in the cache, where the storage includes evicting a currently stored asset if the
    cache is full.
    end
    else begin
20  Request asset  $i$  from upstream cache or origin
    Deliver asset  $i$  and cost  $C_{out,ijk}$  of asset  $i$  to client or lower layer cache, where the cost
     $C_{out,ijk}$  includes an additional marginal cost  $C_{sto,ijk}$  due to the asset  $i$  not being stored locally
    end

```

25 Algorithm 2 will be explained with reference to FIG. 4, from the perspective of cache 135B which, as explained above with reference to FIG. 2A, corresponds to cache (0,1) in FIG 2A (i.e., the 0th cache in hierarchy level $k=1$).

Referring to FIG. 4, in operation S405, the cache receives a request for an asset. For example, the cache 135B may receive the request for the asset x from one of the children
30 caches of cache 135B (i.e., as is illustrated in FIG. 2A, the cache 135B is the parent to caches 135D and 135E, which correspond to cache (0,0) and cache (1,0) illustrated in FIG. 2A, respectively).

Hereinafter, for the sake of explanation, it is assumed that cache 135B receives a request for the asset x from the cache 135D, for example, in response to cache 135D receiving a request
35 for the asset x from the end user 110. Therefore, from the point of view of cache 135B

receiving a request for asset x , $i=x$, $j=0$, and $k=1$.

As is shown in Algorithm 2, the cache 135B may calculate the exponential moving averages of the marginal value MU_{ijk} using Equation (7). Next the cache may set the value for P_{ijk} as the current time.

- 5 In operation S410, the cache 135B may determine whether or not the asset x is stored (i.e., cached) in the cache 135B. If the cache 135B determines that the asset x is currently stored in the cache 135B, then the cache 135B may proceed to operation S415, otherwise the cache 135B may proceed to operation S420.

10 In operation S415, the cache 135B provides the asset x to cache 135D and updates the LRU database of the cache in accordance with known least recently used (LRU) methods. For example, the cache 135D may update an LRU database within the cache 135D (e.g., within the memory 256 of the cache 135D) that stores timestamps indicating, respectively, times of the most recent uses of each the assets presently stored in the cache. Further the cache 135B provides the cost $C_{out,ijk}$ of the asset $i=x$ to cache 135D where the cost $C_{out,ijk}$ equals the
15 current cost C_{ijk} associated with the asset i .

In some example embodiments, the cost may represent the number of hops (numbers of caches) the asset has passed before being stored in the cache. In other example embodiments the cost may represent the latency the asset has encountered before being stored in the cache. In still other example embodiments, the cost may represent a combination of metrics, for
20 example, both the number of hops and the latency encountered.

If the cache 135B determines in operation S410 that the asset x is not stored therein, the cache 135B proceeds to operation S420.

In operation S420, the cache 135B retrieves the asset x from the origin or another one of the caches. For example, the cache 135B may send a request for the asset x to the parent of the
25 cache 135B, cache 135A. In response to the request sent by the cache 135B, the cache 135B may receive the asset x from the cache 135A. For example, the cache 135A may be storing the asset x already, or the cache 135A may retrieve the asset x from the origin server 140 before providing the asset x to the cache 135B.

In operation S425, the cache 135B determines whether to store the asset x based on a
30 comparison between the moving average of the marginal value MU_{ijk} associated with the asset $i=x$ and the characteristic marginal value CV_{jk} of the cache 135B.

For example, the cache 135B may compare the value MU_{ijk} to the value CV_{jk} in accordance with Equation (6) below.

$$(MU_{ijk} < CV_{jk}) \quad (6)$$

The cache 135B may calculate the characteristic marginal value CV_{jk} using Equation (7). The cache 135B may recalculate the value of the characteristic marginal value CV_{jk} only when the cache 135B is full and one of the previously stored assets needs to be evicted to make room for the asset x . The cache 135B may store the updated value of the characteristic marginal value CV_{jk} .

$$CV_{jk} = \frac{TC_{jk}}{C_{ijk}} = w_{TC} \times \frac{(TS - P_{ijk})}{C_{ijk}} + (1 - w_{TC}) \times CV_{jk} \quad (7)$$

As illustrated in Equation 7, the characteristic marginal value is equal to the characteristic time value TC_{jk} determined in Equation 1 divided by the cost C_{ijk} of asset i to cache (j,k) . Therefore, when evicting an asset i from cache (j,k) , if the cost C_{ijk} associated with obtaining the asset i is high, the characteristic marginal value CV_{jk} , which represents a cost conscious average period of time assets stay in a cache (j,k) without being requested before eviction, will be low.

Further, the cache 135B may calculate the exponential moving averages of the marginal value MU_{ijk} of the asset $i=x$ using Equation (8) below.

$$MU_{ijk} = \frac{T_{ijk}}{C_{ijk}} = w_{IA} \times \frac{(TS - P_{ijk})}{C_{ijk}} + (1 - w_{IA}) \times MU_{ijk} \quad (8)$$

The cache 135B may calculate the marginal value MU_{ijk} for each asset i retrieved by the cache 135B, and may recalculate the marginal value MU_{ijk} of a stored asset i when one of the stored assets i is requested.

As illustrated in Equation 8, the exponential moving averages of the marginal value MU_{ijk} is equal to the average the inter-arrival time of asset T_{ijk} divided by the cost C_{ijk} of asset i to cache (j,k) .

If the cache 135B determines, using Equations 6 to 8, that the moving average of the marginal value MU_{ijk} associated with the asset $i=x$ is not less than the characteristic marginal value CV_{jk} of the cache 135B, the cache 135B may proceed to operation S430.

In operation S430, if the moving average of the marginal value MU_{ijk} associated with the asset $i=x$ is not less than the characteristic marginal value CV_{jk} of the cache 135B, the cache 135B provides the asset to the network element that requested the asset, cache 135D. Further, the cache 135B provides the cost $C_{out,ijk}$ of the asset $i=x$ to cache 135D where, due to the cache 135B not storing the asset x , the cost $C_{out,ijk}$ includes an additional marginal cost C -

sto_{ijk} indicating the cost of cache 135B retrieving the asset x if cache 135D decides to evict the asset at a later time period.

In operation S435, if the cache 135B determines, using Equations 6 to 8, that the moving average of the marginal value MU_{ijk} associated with the asset $i=x$ is less than the characteristic marginal value CV_{jk} of the cache 135B, the cache 135B may store the asset x in the cache 135B and proceed to operation S440.

When the cache 135B stores the asset x therein, since the size of the cache 135B may be definite, the cache 135B may require eviction of a currently stored asset to generate space to store the asset x.

For example, the cache 135B may determine which asset to evict based the largest marginal value MV_{ijk} of the stored assets i.

For example the cache 135B may utilize Equation 9 below to determine the marginal value MV_{ijk} of the assets i, and evict the asset with the largest marginal value MV_{ijk} .

$$MV_{ijk} = \frac{(TS - P_{ijk})}{C_{ijk}} \quad (9)$$

As illustrated in Equation 9, if the cost C_{ijk} associated with the asset i is relatively high, then the marginal value MV_{ijk} of the asset i will be relatively low, and, therefore, that asset i will likely not be evicted due to the high costs associated with re-obtaining that asset i.

Thus, according to one or more example embodiments, caches in the CDN 130 may determine whether or not to cache a retrieved asset in accordance with the comparison defined by equation (6) such that the determination of whether to cache an asset is based on the cost associated with retrieving that asset. Further, the cache 135B may determine which asset to evict to make space for the stored asset in consideration of the cost of obtaining each of the assets i.

In operation S440, the cache 135B may provide the asset x to cache 135D. Further the cache 135B provides the cost $C_{out,ijk}$ of the asset $i=x$ to cache 135D where the cost $C_{out,ijk}$ equals the current cost C_{ijk} associated with the asset i.

After providing the requested asset x in operations S415, S430 and S440, the cache 135B returns to operation S405 to await receipt of the next asset request.

Example embodiments being thus described, it will be obvious that embodiments may be varied in many ways. Such variations are not to be regarded as a departure from example embodiments, and all such modifications are intended to be included within the scope of example embodiments.

WE CLAIM:

1. A method of operating a first cache device, the method comprising:
receiving, at the first cache device, a request to send a first asset to a second device;
5 determining whether the first asset is stored at the first cache device; and
when the determining determines that the first asset is not stored at the first cache
device,

obtaining, at the first cache device, the first asset,

calculating a cost associated with the first asset,

10 selectively storing the first asset at the first cache device based on a moving
average of a marginal value of the first asset with respect to the first cache device and
a characteristic marginal value of the first cache device, the characteristic marginal
value of the first cache device representing a cost conscious average period of time
between receipt of last requests for, and eviction of, assets cached at the first cache,
15 and

sending the first asset and the calculated cost to the second device.

2. The method of claim 1, wherein the first cache is one of a plurality of cache
devices of a content distribution network divided into a plurality of hierarchy levels, and the
20 method further comprises:

for each of the plurality of cache devices,

determining a moving average of the marginal value of a requested asset,
when a request for the requested asset is received by a respective one of the plurality
of cache devices, and

25 determining a characteristic marginal value associated with a respective one of
the plurality of cache devices, when the cache device evicts an asset.

3. The method of claim 2, wherein the obtaining obtains the first asset and an
obtained cost associated therewith from a higher level of the plurality of hierarchy levels, and
30 the calculating the cost includes adding a marginal cost to the obtained cost if the selectively
storing does not store the first asset.

4. The method of claim 1, further comprising:

determining the characteristic marginal value of the first cache device by,

initializing the characteristic marginal value of the first cache device as a value which is higher than the initialized value of the moving average of the marginal value of the first asset with respect to the first cache device, and

updating the characteristic marginal value of the first cache device based on exponentially weighted moving average of periods of time between receipt of last requests for, and eviction of, assets cached at the first cache device and the cost associated with the assets, in accordance with a least recently used cache eviction policy.

5. The method of claim 1, further comprising:

determining the moving average of the marginal value of the first asset with respect to the first cache device by,

initializing the moving average of the marginal value of the first asset with respect to first cache device as a value lower than the initialized value of the characteristic marginal value of the first cache device, and

updating the moving average of the marginal value of the first asset with respect to the first cache device based on exponentially weighted moving average of periods of time between consecutive receipts of requests for the first asset and the cost associated with the first asset, at the first cache device, the requests requesting the first cache device to send the first asset to another device.

6. The method of claim 1, wherein the selectively storing includes,

assigning the first asset to a first database, when an initial request for the first asset is received at the first cache device; and

when a second request is received consecutively with respect to the initial request,

determining an inter-arrival time of the first asset based on times at which the initial and second requests were received at the first cache device, and

if a number of assets stored in an inter-arrival database is greater than or equal to a threshold then,

determining a second asset to remove from an inter-arrival database based on which of a plurality of assets stored in the inter-arrival database has a largest

marginal value associated therewith that decreases as the cost associated with the second asset increases, the inter-arrival time database storing arrival times of requests corresponding to assets, the inter-arrival time database being different than the first database, and

5 removing the second asset from the inter-arrival database; and
 assigning the first asset to the inter-arrival time database.

7. The method of claim 6, further comprising:

 determining the marginal value of the assets by dividing a last requested time of each
10 of the plurality of assets by the cost associated therewith, wherein

 the cost associated with the first asset represents one or more of a number of hops between a source of the first asset and the first cache device and a latency the first asset encountered therebetween.

15 8. A first cache device comprising:

 a processor and a memory, the memory containing computer readable code that, when executed by the processor, configures the processor to,

 receive a request to send a first asset to a second device,

 determine whether the first asset is stored at the first cache device, and

20 when the processor determines that the first asset is not stored at the first cache device, the processor is configured to,

 obtain, at the first cache device, the first asset,

 calculate a cost associated with the first asset,

 selectively store the first asset at the first cache device based on a
25 moving average of a marginal value of the first asset with respect to the first cache device and a characteristic marginal value of the first cache device, the characteristic marginal value of the first cache device representing a cost conscious average period of time between receipt of last requests for, and eviction of, assets cached at the first cache, and

30 send the obtained first asset and the calculated cost to the second device.

9. The first cache device of claim 8, wherein,

the first cache is one of a plurality of cache devices of a content distribution network divided into a plurality of hierarchy levels, each of the plurality of cache devices having a processor associated therewith, and

each one of the plurality of cache devices is configured to,

5 determine a moving average of the marginal value of a requested asset, when a request for the requested asset is received by a respective one of the plurality of cache devices, and

 determine a characteristic marginal value associated with a respective one of the plurality of cache devices, when the cache device evicts an asset.

10

10. The first cache device of claim 9, wherein the processor is configured to, obtain the first asset and an obtained cost associated therewith from a higher level of the plurality of hierarchy levels,

 calculate the cost by adding a marginal cost to the obtained cost if the processor
15 determines not to store the first asset, and

 selectively store the first asset by,

 assigning the first asset to a first database, when an initial request for the first asset is received at the first cache device;

 if a number of assets stored in an inter-arrival database is greater than or equal
20 to a threshold when a second request is received consecutively with respect to the initial request then,

 determining an inter-arrival time of the first asset based on times at which the initial and second requests were received at the first cache device, and

 determining a second asset to remove from an inter-arrival database
25 based on which of a plurality of assets stored in the inter-arrival database has a largest marginal value associated therewith that decreases as the cost associated with the second asset increases, the inter-arrival time database storing arrival times of requests corresponding to assets, the inter-arrival time database being different than the first database, and

 removing the second asset from the inter-arrival database; and

 assigning the first asset to the inter-arrival time database.

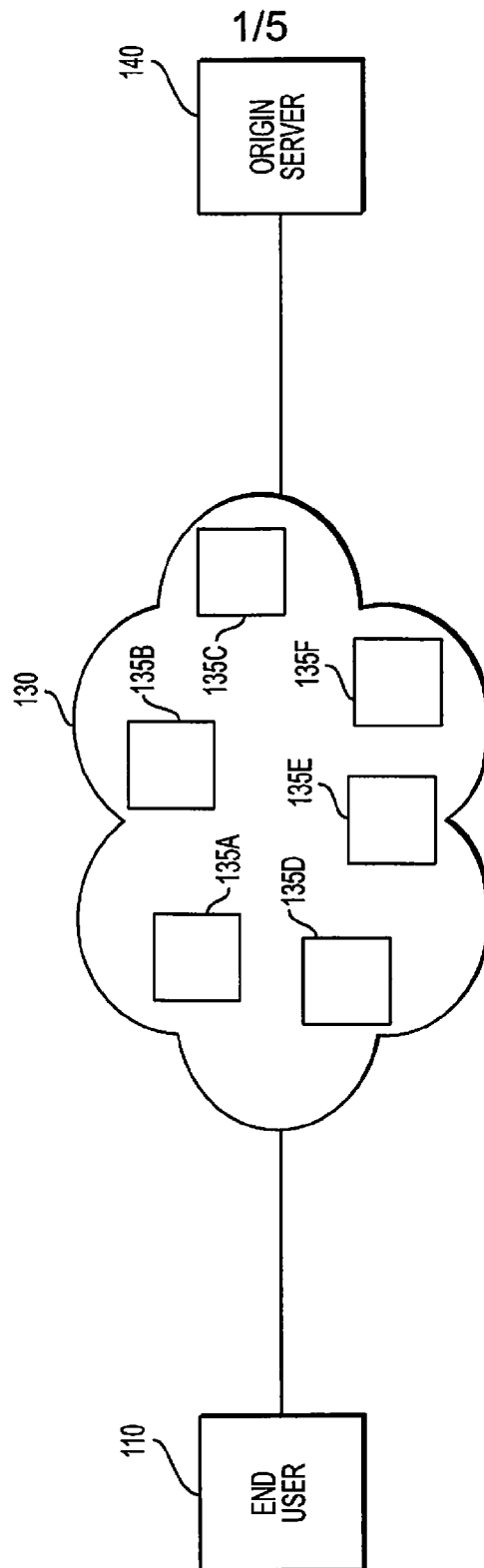


FIG. 1

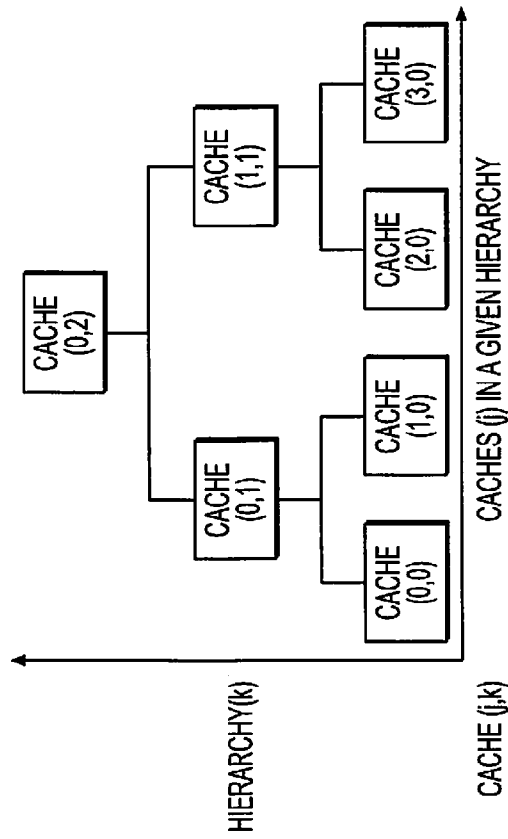


FIG. 2A

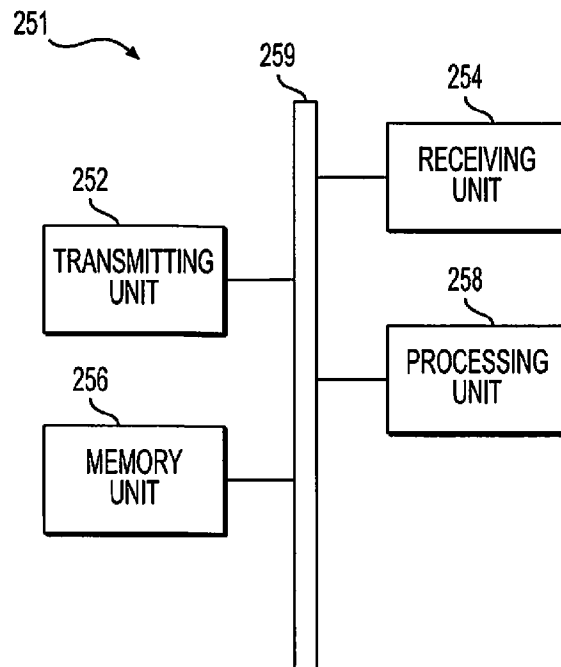


FIG. 2B

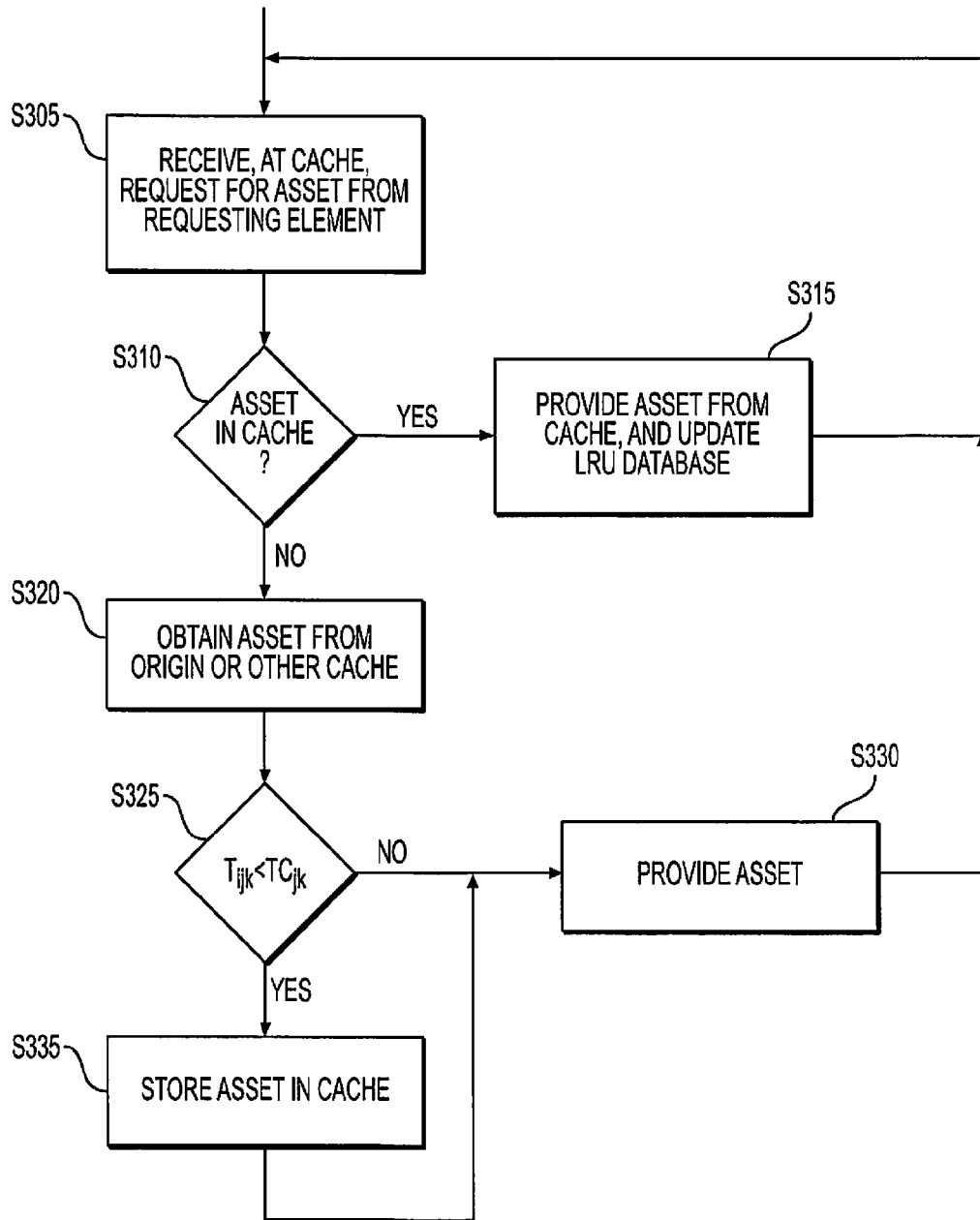


FIG. 3

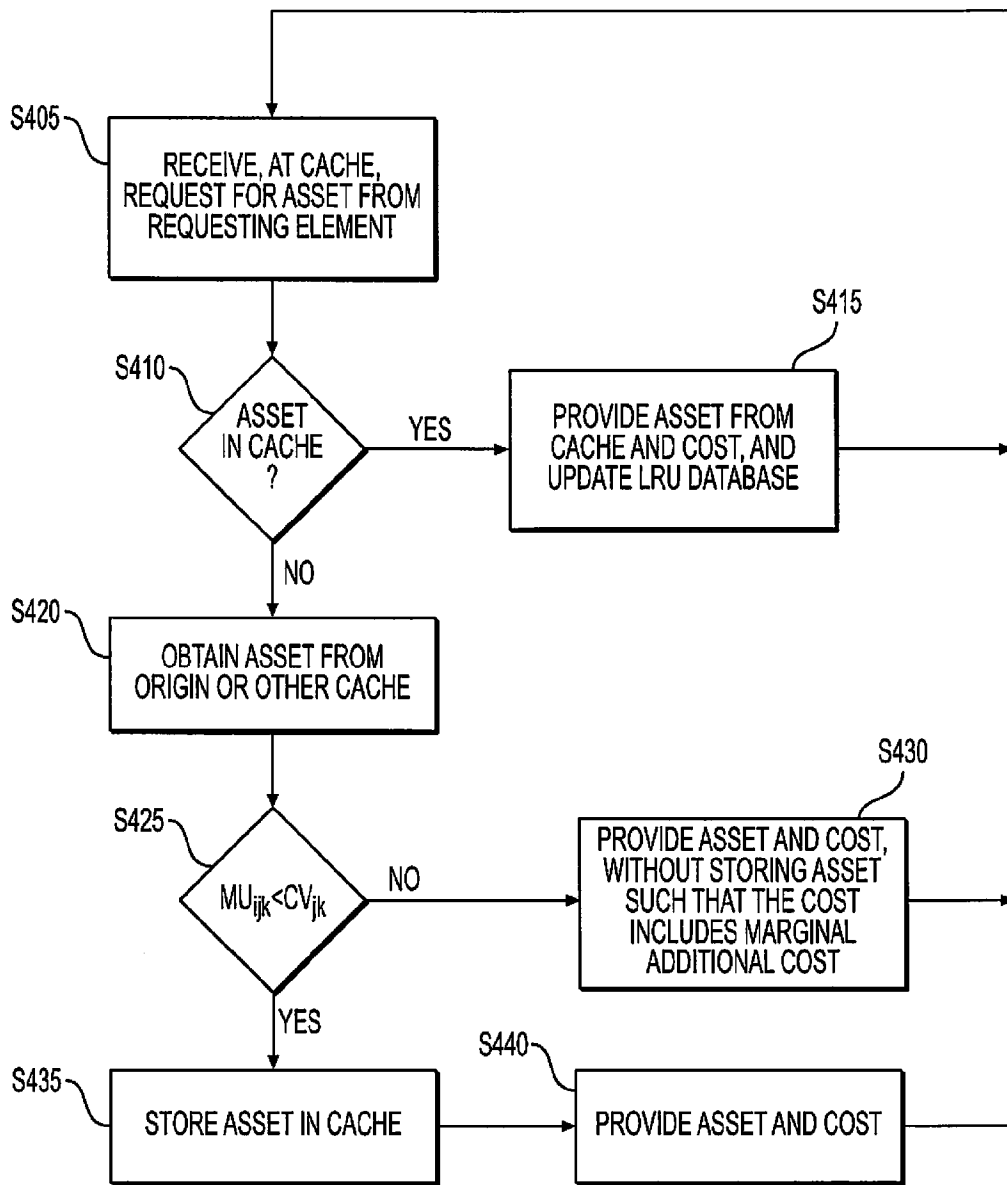


FIG. 4

INTERNATIONAL SEARCH REPORT

International application No PCT/US2016/021421

A. CLASSIFICATION OF SUBJECT MATTER INV. G06F12/12 G06F12/08 H04L29/08 ADD.				
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) EPO-Internal, WPI Data				
C. DOCUMENTS CONSIDERED TO BE RELEVANT				
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.		
X	US 2013/268733 A1 (NARAYANAN ASHOK [US] ET AL) 10 October 2013 (2013-10-10) abstract paragraph [0027] - paragraph [0036] figures 1, 3-4 -----	1-10		
A	US 2011/107030 A1 (BORST SIMON [US] ET AL) 5 May 2011 (2011-05-05) abstract figures 1, 3, 5 -----	1-10		
A	US 6 826 599 B1 (SHAFFER SHMUEL [US] ET AL) 30 November 2004 (2004-11-30) abstract column 7, line 1 - column 8, line 34 figures 1, 3 ----- -/--	1-10		
<input checked="" type="checkbox"/> Further documents are listed in the continuation of Box C. <input checked="" type="checkbox"/> See patent family annex.				
* Special categories of cited documents : <table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> "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 </td> <td style="width: 50%; border: none; vertical-align: top;"> "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 </td> </tr> </table>			"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
"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			
22 April 2016	06/05/2016			
Name and mailing address of the ISA/ European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Authorized officer Mandato, Davide			

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2016/021421

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2014/280679 A1 (DEY SUJIT [US] ET AL) 18 September 2014 (2014-09-18) abstract paragraph [0055] - paragraph [0067] paragraph [0163] - paragraph [0185] figures 2, 3, 12C, 13, 14 -----	1-10
A	US 2012/054445 A1 (SWART GARRET FREDERICK [US] ET AL) 1 March 2012 (2012-03-01) abstract figures 2, 3, 4A, 4B, 10 -----	1-10
A	EP 2 830 285 A1 (THOMSON LICENSING [FR]) 28 January 2015 (2015-01-28) abstract paragraph [0007] - paragraph [0018] figure 3 -----	1-10
A	US 2006/090040 A1 (ESCHMANN MICHAEL K [US]) 27 April 2006 (2006-04-27) abstract paragraph [0010] - paragraph [0017] figure 3 -----	1-10

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No PCT/US2016/021421

Patent document cited in search report	A1	Publication date	Patent family member(s)	Publication date
US 2013268733	A1	10-10-2013	NONE	

US 2011107030	A1	05-05-2011	NONE	

US 6826599	B1	30-11-2004	US 6826599 B1	30-11-2004
			US 7107321 B1	12-09-2006

US 2014280679	A1	18-09-2014	NONE	

US 2012054445	A1	01-03-2012	CN 103154912 A	12-06-2013
			CN 103168293 A	19-06-2013
			CN 104850510 A	19-08-2015
			EP 2612249 A1	10-07-2013
			EP 2612250 A2	10-07-2013
			EP 2746954 A2	25-06-2014
			US 2012054445 A1	01-03-2012
			US 2012054447 A1	01-03-2012
			WO 2012030900 A1	08-03-2012
			WO 2012030903 A2	08-03-2012

EP 2830285	A1	28-01-2015	EP 2830285 A1	28-01-2015
			EP 2830288 A1	28-01-2015
			US 2015033255 A1	29-01-2015

US 2006090040	A1	27-04-2006	NONE	
