A system includes cloud-based resources having backend storage for maintaining single copies of display content items, such as images, and a frontend content delivery network (CDN) to dynamically format and deliver the content items in response to client requests. In the case of digital images, for example, a single copy of full resolution version of the image is stored in the backend storage, rather than dozens of versions with different resolutions. The frontend CDN is configured with multiple graphic processing units (GPUs) configured to rescale the full resolution image in real-time at the time of delivery to the client device. By rescaling images on the fly at the edge of the cloud-based resources, the system reduces storage costs while minimizing any negative impact on latency.
FROM CONTENT STORAGE

FROM CONTENT STORAGE

CPUs 202

COMPUTER-READABLE MEDIA 204

MARKER READER MODULE 304

CODEC MODULE 208

IMAGE SEGMENTING MODULE 210

GPUS 142

COMMUNICATION INTERFACE 206

CONTENT DELIVERY NETWORK 122

GPU 142(1)

GPU 142(2)

GPU 142(3)

GPU 142(4)

222(A)

222(B)

222(C)

222(D)

224(A)

224(B)

224(C)

224(D)

226

DELIVERY TO CLIENT

FIG. 3
POST NEW IMAGE 402

STORE SINGLE FULL RESOLUTION IMAGE IN CLOUD STORAGE 404

OPTIONALLY COMPRESS THE IMAGE 406

OPTIONALLY ADD BOUNDARY MARKERS TO IMAGE 408

SUBMIT REQUEST FOR THE IMAGE FROM USER 1 ON DEVICE 1 410

RECEIVE REQUEST 412

ASCERTAIN DISPLAY CAPABILITIES OF CLIENT DEVICE 1 414

RETRIEVE FULL RESOLUTION IMAGE FROM STORAGE 416

CACHE FULL RESOLUTION IMAGE IN SHORT TERM CACHE 418

RESCALE IMAGE IN CACHE USING GPUS 420

SERVE RESCALED IMAGE 422

RECEIVE AND RENDER RESCALED IMAGE ON DEVICE 1 424

FIG. 4
Submit request for the image from User 2 on Device 2 502

Receive request 504

Ascertain display capabilities of client device 2 506

Retrieve full resolution image from cache 508

Rescale image in cache using GPUs 510

Serve rescaled image 512

Receive and render resized image on Device 2 514

FIG. 5
FIG. 7
IMAGE DELIVERY ARCHITECTURE

BACKGROUND

[0001] With the explosion of inexpensive storage and ever increasing network bandwidth, there has been rapid growth of cloud-based storage for various digital content items (e.g., text, audio, images, and video items). Following storage, these content items may be downloaded or streamed from the cloud-based computing services to one or more client devices. Many sites exist today allow users to post or otherwise store their own content in cloud-based storage for either private or public consumption. For instance, sites like Facebook®, Pinterest®, YouTube®, and others allow users to post content in the form of text, images, audio, and video for others to view.

[0002] Conventional cloud-based architectures typically involve logical arrangements of large numbers of servers to perform various functions. One common arrangement involves client-facing servers that receive requests from client devices, process those requests to discern what content the user is seeking, and then ultimately deliver the requested content to the client devices. A logical grouping of servers that deliver content to clients is often referred to as a content delivery network or CDN. The common arrangement further includes backend servers that perform various functions such as data aggregation, storage, analysis, and searching. Typically, the client-facing or frontend servers communicate with the backend servers as part of the process for serving content to a user.

[0003] Over the years, the primary focus of cloud-based architectures has been improving speed and quality to the clients. For sites that permit storage of content, site operators continually weigh customer experience (e.g., latency from the time a request for a content item is received to delivery of the content item to the user) against storage costs and/or bandwidth requirements. The problem is complicated by the number of devices currently available to consume the content items because the various devices have different processing, interface, and display capabilities. For example, an image or video clip may be presented on any number of devices, including on a smart phone, tablet, laptop, desktop computer, or television. Each of these devices may require different formats of the image or video clip.

[0004] To illustrate these challenges facing content site operators in more detail, consider a scenario of an example site that allows users to post images. Suppose a user posts a full resolution image (e.g., 1024×768 or 1 MB) to the site. Today, it is common for site operators to process this full resolution image and create multiple versions of this image that can be served to different client devices. For instance, an intake server may use its central processing units (CPUs) to create multiple versions, such as a smaller version of the image (e.g., 100×76 or 3 kB), a medium version of the image (e.g., 512×384 or 20 kB), and a bigger version of the image (e.g., 640×480 or 50 kB). All of these multiple versions, along with the original image, are stored in content storage for subsequent retrieval and distribution. In some cases, it is not uncommon for site operators to re-render dozens of image versions having different sizes, resolutions, or other features for delivery to myriad client devices.

[0005] When the content delivery network receives a request for the image, it determines the display requirements of the client device and initiates a fetch to retrieve the appropriate version of the image. Suppose, in our example, that the user requests the image from her smartphone. The CDN may fetch the small version of the image from the backend content storage. This small version is ultimately served to the client device. Now, suppose the user (or another user) requests the image from a browser in her desktop computer. In this case, the CDN may fetch the bigger version or the full resolution version of the image from the backend content storage. The CDN then serves this larger version of the image to the client device.

[0006] With the current architectures, creating and storing multiple versions of the image in the backend storage results in higher operation costs in terms of heightened processing resources and increased storage requirements. Further, each request for a different image version requires a fetch operation that may affect customer experience through increased latency. In some situations, the fetched images may be temporarily cached at the CDN to accommodate future requests for the same image version. Unfortunately, this exacerbates the storage inefficiencies as these multiple versions are maintained at the edge of the cloud in addition to the full set of image versions being maintained at the backend content storage.

[0007] Accordingly, there remains a continuing need for improved architectures that reduce operating costs while improving customer experience.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The detailed description is set forth with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The use of the same reference numbers in different figures indicates similar or identical items or features.

[0009] FIG. 1 illustrates an example image delivery architecture in which only a single copy of a content item, such as an image, is maintained and any new version of the content item is produced at the time of delivery.

[0010] FIG. 2 is a block diagram showing select components in a content delivery network used in the image delivery architecture of FIG. 1. FIG. 2 further illustrates how the content item is scaled at the time of delivery.

[0011] FIG. 3 is a block diagram showing additional select components in the content delivery network similar to FIG. 2, but further illustrating use of boundary markers to segment the content item for parallel processing during content transformation.

[0012] FIGS. 4 and 5 show a flow diagram illustrating example process executed by various devices in the architecture to store a single version of a content item and transform the content item at the time of delivery to different client devices.

[0013] FIG. 6 illustrates another example image delivery architecture in which only a single copy of a content item is maintained and versions of the content item are scaled by the client device after delivery.

[0014] FIG. 7 is a flow diagram illustrating an example process executed by various devices in the architecture of FIG. 6 to store a single version of a content item and transform the content item after delivery to the customer.

DETAILED DESCRIPTION

[0015] This disclosure describes architectures and techniques for storing content items in cloud-based storage and
delivering the content items from the cloud-based storage to client devices in response to requests from users. Unlike conventional arrangements where multiple copies of a content item, such as an image, are maintained in the cloud-based storage and then individually fetched based on the display capabilities of the client, the architecture described herein may store as little as just a single copy of the content item. In particular, in the case of digital images, a full resolution version of the image is stored in the cloud-based storage. More generally, the architecture significantly reduces the number of copies of the content item (e.g., 1, 2 or a few) to be stored in comparison to conventional architectures that stored a full set of possible rendered versions (e.g., on a scale of dozens of copies in various formats and sizes).

[0016] The frontend content delivery network (CDN) is configured with servers that are equipped with one or more graphic processing units (GPUs). A GPU is a specialty electronic circuit designed to rapidly manipulate and accelerate the creation of visual images for output to a display. When a user requests the image, the CDN determines the display capabilities of the requesting device either from data received with the request or through independent processes. The CDN retrieves the image from the cloud-based storage and uses the GPUs to rescale the image in real-time at the time of delivery to the client device.

[0017] The following example architectures are described in the context of delivering digital images. However, aspects of the systems and techniques described herein may be applied to other content items, such as audio and video. Furthermore, while this architecture is described in the context of cloud-based storage, aspects of the systems and techniques described herein may be applied to other forms of storage, such as storage area networks (SAN), RAID systems, or hard drives.

Example System Architecture

[0018] FIG. 1 illustrates an example architecture of a computing system 100 that includes cloud-based storage and computing resources 102 to receive content from, and serve content to, client devices associated with various users. As one non-limiting example, the cloud resources 102 may be representative of various sites on the web that allow users to store content and subsequently share or retrieve that content using any number of devices. Examples of such sites might include entertainment sites, organization or collection sites, photo sites, social network sites, e-commerce sites, advertising sites, or the like.

[0019] In this illustration, a first user 104 uses a first client device 106, such as a desktop computer, to post content items for storage in the cloud resources 102. Depending upon the implementation, the content items may include any type of media, such as text, audio, images, and video. The content items may take the form of textual posts, digital photographs, songs, movies, and so forth. In this particular example, the user 104 is on a site that allows the user to post digital images. In other scenarios, the content items stored in the cloud resources 102 may be placed there through other techniques than a user uploading the items. For instance, the cloud resources 102 may communicate with other resources (e.g., such as other websites or storage locations) to retrieve and store the content items.

[0020] A graphical user interface (GUI) 108 is depicted on the display of the client device 106. The GUI 108 may be rendered based on instructions provided by the site to the computer’s browser. The GUI 108 includes multiple post areas 110, each of which has an associated image or collection of images. As shown here, ten images are shown posted, including images 111, 112 and a new image 114 of a person’s portrait, as represented by the smiley face symbol. The GUI 108 allows the user to upload the digital images for storage in the cloud resources 102 and to search the user’s personal images, as well as other people’s images to which the user has been given permission or that were made public by the poster. While a desktop computer is shown, essentially any computing device with memory, processing, and display technologies may be used to depict the GUI 108 and post the images, such as new image 112.

[0021] The user’s client device 106 is coupled to communicate with the cloud resources 102 via one or more network(s) 114. The network(s) 114 are representative of generally any type of wireless or wired networks including, non-limiting examples, an intranet, a home network, the Internet, cable networks, cellular networks, radio networks, near-field networks, a LAN, WAN, VPN, Wi-Fi, and so on. Protocols and components for communicating via such networks are well known and will not be discussed herein in detail.

[0022] The cloud resources 102 include processing, storage, and networking technologies. As shown in FIG. 1, the resources are generally arranged to include frontend resources that interface with the client devices and backend resources that perform various functions such as data aggregation, content storage, analysis, and searching. The frontend resources include UI and upload servers 120 and a content delivery network (CDN) 122. The backend resources include content storage 124. While the figures may imply that the frontend and backend resources are co-located in a single location, it is to be appreciated that these various resources may be distributed across different locations. Moreover, the resources within the logical groups, such as the CDN 122, may be distributed across multiple datacenters at different locations.

[0023] The UI and upload servers 120 include one or more servers 126 that are configured with code to serve the GUI 108 when the user visits the site. The servers 120 further facilitate ingestion of images or other content items that the user desires to store in the content storage 124. The servers 126 may be implemented as a single server, a cluster of servers, a server farm or datacenter with racks of server blades, virtual servers, and so forth, although other computer architectures (e.g., a mainframe architecture) may also be used. Further, the described functionality may be provided by the servers of a single entity or enterprise, or may be provided by the servers and/or services of multiple entities or enterprises.

[0024] For purposes of continuing discussion, suppose the user 104 decides to post a new image 112 through the GUI 108 to be uploaded and stored on the cloud-based storage. The user 104 may employ various techniques to identify the new image 112 and initiate the process to transfer that image from the client device 106 over the networks 114 to the upload servers 126 at the frontend of the cloud resources 102. In one implementation, the user 104 selects a full resolution version of the image. In one example format today, a full resolution image may be a digital color image that is capable of rendering at a resolution of 1024×768 pixels (i.e., a width of 1024 pixels and a height of 768 pixels). This image resolution may equate to a size of approximately 1 MB of data. Furthermore, various image formats may be used, such as JPG, GIF, PNG,
The servers 126 send the image to the content storage 124 at the backend of the cloud resources 102.

The content storage 124 includes one or more servers 128 that are configured to store images or other content items in persistent storage or datastores, as represented by a datastore 130. A single copy of the image 112 is shown stored in the datastore 130. The servers 128 may be implemented in any number of ways, including as database servers, file servers, and so on. In a very basic configuration, the servers include components such as processors (e.g., central processing units or CPUs) and computer-readable media. Each processor may itself comprise one or more processors or cores.

The computer-readable media may be an example of non-transitory computer storage media and may include volatile and nonvolatile memory and/or removable and non-removable media implemented in any type of technology for storing information such as computer-readable instructions, data structures, program modules or other data. Such computer-readable media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other computer-readable media technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, solid state storage, magnetic disk storage, RAID storage systems, storage arrays, network attached storage, storage area networks, cloud storage, or any other medium that can be used to store content items and which can be accessed by the processors directly or through another computing device. Accordingly, the computer-readable media may be computer-readable media able to maintain instructions, modules, or components executable by the processor, as well as the content items such as image 112.

It is important to note that as little as one copy or version of the image 112 can be stored. In the architecture 100, there is no need for additional versions of various sizes/resolutions to be stored in the content storage 124. Accordingly, unlike conventional architectures where dozens of versions of the same image may be stored, maintaining as little as a single copy or version of the image 112 provides significant savings in processing and storage costs. More generally, processing and storage cost savings may be realized when the number of versions stored is less than the total number of possible versions suitable for presentation on a full array of electronic devices.

In some implementations, the content storage 124 may further include compression modules 132 that execute on the server processors to compress the content items prior to storage. The compression modules 132 may use any forms of compression algorithms, depending in part upon the content type being compressed. For images, the compression format may include JPEG, GIF, JPG, etc. Accordingly, the image 112 may be stored in its full size (e.g., 1 MB) or compressed to a smaller file size (e.g., <1 MB) without loss of resolution quality when decompressed for serving.

In some implementations, the content storage 124 may also implement a marker module 134 that executes on the server processors to mark certain features or locations of the images or other content items. For instance, in video items, restart markers or key frames may be used to identify sets of multiple frames. Markers may also be used for JPEG images. These markers identify various boundaries in the image that allow the image to be segmented. These markers may be used for a compressed or non-compressed image so that image segments may be processed without first being decoded or decompressed. In this manner, segments of the images may be processed in parallel by GPUs during delivery to improve responsiveness (i.e., decrease latency), as will be described below in more detail.

While not shown, it is noted that other types of modules for processing images or other content items may be stored and executed at the content storage 124.

The content delivery network 122 includes one or more servers 140 to receive requests from users for various content items stored in the content storage 124 and deliver those requested content items to the users. The servers 140 generally include processors and storage media to store and execute various modules, data, information, and instructions. The servers 140 may be arranged in any number of ways, including as a single server, a cluster of servers, a server farm or data center with racks of server blades, virtual servers, and so forth, although other computer architectures (e.g., a mainframe architecture) may also be used.

Of particular note, the servers 140 are equipped with many graphic processing units (GPUs) 142. These GPUs 142 may be any type of specialty processors designed to process image data at a much faster rate than traditional CPUs or other types of processors. The servers 140 may or may not include CPUs, but are specially configured to process the images and other content items using the GPUs. In various implementations, the GPUs 142 may be used to rescale the images, such as image 112, into any number of resolutions or sizes. Due to the processing speeds, the GPUs 142 are able to rescale the images in real-time, in response to user requests for delivery of the images. One example type of GPU is any of the GPUs available from NVIDIA Corporation. Further, while the term GPU is being used herein to describe a special function processing unit, other arrangements of massively parallel server cores configured to specifically process visual content items rapidly may be used for the purposes defined herein as graphic processing units.

The content delivery network 122 may further include a device type identifier 144 to discern the types of clients making requests. Identifying the types of clients helps to ascertain what type of display capabilities and formats are supported by the client devices when rendering the content items. For instance, a smartphone has different size and format constraints for rendering a digital image than a tablet or desktop computer. Furthermore, in some situations, different resolution images may be rendered on the same display but within different screen sizes. For instance, in a social networking or e-commerce context, a small image (e.g., a thumbnail) may be rendered initially in a small location on one screen, and then upon selection of the small image, a larger version of that image may be rendered in a larger display area.

The content delivery network 122 may also maintain a short-term cache 146 to temporarily store content items for delivery to requesting users. The cache 146 may be formed of any type of memory or storage media, such as RAM, flash memory, or other computer-readable media technology.

As shown in FIG. 1, a population of multiple users 150 may use any number of client devices 152 to request content from the cloud resources 102. The users 150 are representative of the people who use or access the site supported by the cloud resources 102. Depending upon the type of site (e.g., social network, photo sharing, collections, e-commerce, etc.), the users 150 may represent consumers,
friends, family members, sellers, buyers, hobbyists, and so forth. The users’ population 150 may further include the original posting user 104.

[0035] Each of these users 150 may access one or more content items stored on the backend content storage 124. Suppose, for discussion purposes, that each of the users 150 may request access to the image 112 posted by the user 104. The users 150 may be from the general population, if no restrictions are placed on who can access the image 112, or part of a select group within a network defined by the posting user 104 or other content managers.

[0036] The devices 152 used by the users 150 may be of any type or profile of electronic devices that can be used to consume content items. These devices 152 generally include processing and memory resources, but are also equipped with (or able to access) a display for rendering the content items. Any type of display technologies may be employed, such as liquid crystal display, plasma display, light emitting diode display, organic light emitting diode display, and so forth. Representative client devices shown in FIG. 1 include a desktop computer 152-1, a tablet 152-2, a television or large display 152-3, a personal digital assistant 152-4, a laptop 152-5, and a smart phone or other communication device 152-6. Each of these devices 150 has different display capabilities, including various size constraints, processing capabilities, resolution levels, and so forth.

[0037] The users 150 utilize the various devices 152 to access the cloud resources 102 via one or more networks 154. The network(s) 154 may be one or more types of wired and wireless networks, including cellular, cable, Wi-Fi, LAN, WAN, Internet, and so forth. The devices 152 may be equipped with one or more communication interfaces that support both wired and wireless connection to the networks 154, such as cellular, radio, Wi-Fi, short-range or near-field networks (e.g., Bluetooth®), infrared, and so forth.

[0038] For discussion purposes, suppose a first user (e.g., user 150-1) uses a desktop computer 152-1 to access the image 112 posted by the original user 104. A request for the image 112 may be spawned by the user merely visiting the site and the desired image 112 is automatically served to the device 152-1 as a page of content. In other situations, the request may be explicit from the user who identifies and requests the image. To illustrate, suppose the image 112 is posted to a collections or photo-sharing site and by accessing this site, the user 150-1 initially sees a small version of this image associated with content from the posting user 104. After viewing this initial page, suppose that the user selects the image to see it in a larger form. Both accessing the site and user selection may be interpreted by the site as a request for versions of the image 112 maintained in the backend content storage 124.

[0039] Upon receipt of such requests, the content delivery network 122 fetches or otherwise retrieves the image 112 from the persistent datastore 130 of the content storage 124. The CDN 122 may temporarily store the image 112 in the short-term cache 146. Meanwhile, the device type identifier 144 attempts to discern the type of client device making the request. In some implementations, the client request may include metadata that informs the CDN 122 of the display capabilities or the size of the image it is requesting. In other implementations, the request may include metadata indicating the type of client device. In this latter case, the device type identifier 144 uses the metadata to look up the device’s display capabilities to understand what types of software, hardware, and/or display technologies are present to render the image. As used herein, the display capabilities might include hardware capabilities (e.g., whether the display has sufficient pixel density to render high resolution images), software capabilities imposed by the operating system or application (e.g., size of the window or space allocated to display the image), or a combination of hardware and software capabilities. Identifying the device type and retrieving the image 112 may be done in parallel to save time.

[0040] Based on the display capabilities of the requesting device, the GPUs 142 of the CDN 122 rescale the image 112 from its full resolution as stored in the datastore 130 to one or more versions that are different than the original resolution of the image. This rescaling is performed in real-time with minimal added latency as part of the process of delivering the image to the requesting user. For example, in response to the desktop computer 152-1 accessing the site, the GPUs 142 may rescale the image 112 to a small version of the image (e.g., 100×76 or 3 kB) that appears as a thumbnail image in association with the posting user. When the user selects the image to view it in a larger format, a new request is submitted to the CDN 122 and the GPUs 142 either serve the full original image 112 or rescale it to a larger version of the image (e.g., 640×480 or 50 kB) for rendering on the larger desktop display. This is represented by the rendering of the full version or larger version of the image 112(L) on the display screen of the computer 152-1. In some cases, the GPUs may oversize or scale up the image from its original resolution.

[0041] Similarly, suppose the user 150-2 uses a smart phone 152-6 to access the image 112. In this example, the device type identifier 144 may recognize that the request is from a mobile device (e.g., received as part of a cellular communication format, etc.) and hence, the GPUs 142 rescale the image to a medium version of the image (e.g., 512×384 or 20 kB); that is capable of being displayed on the mobile device. This is represented by the rendering of the medium version of the image 112(M) on the display screen of the computer 152-1.

[0042] In this manner, the GPUs 142 are configured to process the images at the edge or from one of the cloud services during fulfillment of a client request. Due to the high processing speeds tailored for visual images, the GPUs 142 are able to perform the rescaling in real-time without noticeable degradation in the user experience. As a result, only one copy of the image is maintained in the content storage (apart from perhaps usual redundancy measures where images may be redundantly maintained in separate systems or locations), thereby saving storage costs.

[0043] In addition, a temporary copy of the full resolution image may be maintained for a brief time in the short-term cache 146. In this way, when subsequent requests for the image are received, the CDN 122 can immediately rescale the images on the fly without making another fetch to the backend storage 124. This further decreases latency, thereby improving user experience. Indeed, the client device can request essentially any size or resolution desired, and the GPUs 142 can rescale the image to satisfy the request.

Edge-Based Gpu Image Processing Techniques

[0044] FIG. 2 shows select components in the content delivery network 122 and a corresponding illustration to represent how a content item, such as an image, is scaled at the time of delivery. The servers 140 of the CDN 122 include multiple CPUs 202, computer-readable media 204, multiple
GPUs 142, and one or more communication interfaces 206. The CPUs 202 may include multiple computing units or multiple cores. The CPUs 202 are configured to fetch and execute computer-readable instructions stored in the computer-readable media 204 or other computer-readable media, such as a coder/decoder (codec) module 208 and an image segmenting module 210.

[0045] The computer-readable media 204 may include volatile and nonvolatile memory and/or removable and non-removable media implemented in any type of technology for storage of information, such as computer-readable instructions, data structures, program modules or other data. Such computer-readable media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, solid state storage, magnetic disk storage, RAID storage systems, storage arrays, network attached storage, storage area networks, cloud storage, or any other medium that can be used to store the desired information and that can be accessed by a computing device. Depending on the configuration of the servers 140, the computer-readable media 204 may be a type of computer-readable storage media and may be a non-transitory storage media.

[0046] The computer-readable media 204 may be used to store any number of functional components that are executable by the CPUs 202. In many implementations, these functional components comprise instructions or programs that are executable by the CPUs 202 and that, when executed, implement operational logic for performing the actions attributed to the CDN 122. The communication interface(s) 206 may include one or more interfaces and hardware components for enabling communication with various client devices 152 and/or backend computing devices, such as the content storage servers 128. For example, communication interface(s) 206 may facilitate communication through one or more of the Internet, cable networks, cellular networks, wireless networks (e.g., Wi-Fi, cellular) and wired networks.

[0047] The GPUs 142 may be implemented by multiple specially designed processors for handling large calculations involved in rendering or transforming images. The GPUs 142 may be multiple computing units or multiple cores, and may be configured to run in parallel with one another to improve throughput speeds.

[0048] Various instructions, methods, and techniques described herein may be considered in the general context of computer-executable instructions, such as program modules stored on computer storage media and executed by the processors herein. Generally, program modules include routines, programs, objects, components, data structures, etc., for performing particular tasks or implementing particular abstract data types. These program modules and the like, may be executed as native code or may be downloaded and executed, such as in a virtual machine or other just-in-time compilation execution environment. Typically, the functionality of the program modules may be combined or distributed as desired in various implementations. An implementation of these modules and techniques may be stored on computer storage media or transmitted across some form of communication media.

[0049] Illustrated to the right of the CDN components is a process flow diagram that shows how the CDN 122 can rapidly process an image (or other content item) in real-time as part of responding to a request. In this example, the image is received in a compressed format 220 from the content storage 124. The compressed image may be in any suitable format, such as JPG, GIF, PNG, and so on. The codec module 208 decompresses the received compressed image to restore a full resolution image 222. The codec module 208 may be implemented in software, hardware, and/or firmware. For instance, the codec module 208 may be software executed by the CPUs 202. In this manner, only one copy of a full resolution image is maintained in a compressed format by the content storage 124.

[0050] The image segmenting module 210 is then used to segment the image into multiple image segments, as represented by the four image segments 222(A), 222(B), 222(C), and 222(D). Each of these image segments 222(A)-(D) is provided in parallel to GPUs 142(1)-(4), respectively. In this example, the GPUs 142(1)-(4) are downsizing the image to a smaller resolution, as represented by the smaller image segments 224(A), 224(B), 224(C), and 224(D). The smaller image segments 224(A)-(D) are put back together to form a resized image 226, which can then be delivered to the client device. In one implementation, the GPUs 142 may put the image segments back together and the CPUs 202 may manage delivery of the resized image 226 to the client devices.

[0051] The use of GPUs at the edge CDN 122 and the ability to process pieces of the image in parallel greatly increases the speed at which the resized image can be delivered. This allows a single copy of the image to be stored in the persistent content storage 124, while rapidly serving scaled versions of the image in response to user requests.

[0052] FIG. 3 shows select components in the content delivery network 122 additional select components in the content delivery network similar to FIG. 2, but further illustrates use of boundary markers to segment the image. In this example, the image 222 is stored in full resolution in the content storage 124 and it may or may not be maintained in a compressed format. Boundary markers 302 are added to the full resolution image 222 or the compressed image 220 to allow segmentation of the image for parallel processing by the GPUs 142(1)-(4). The marker module 134 of the content storage 124 adds the markers during ingestion of the images prior to storage in the datastore 130. The CDN 122 implements a marker reader module 304 to locate the markers and pass it to the image segmentation module 210 for use in segmenting the image regardless of whether it is compressed or uncompressed. Through user of boundary markers in the compressed image, the image does not need to be decoded prior to processing by the GPUs 142. Rather, the image segments of the compressed image may be processed in parallel by the GPUs 142.

Example Processes

[0053] FIGS. 4 and 5 show an example process 400 executed by various devices in the system 100 to store a single version of a content item, such as an image, and scale the content item at the time of delivery. The process is illustrated as a collection of blocks in logical flow diagrams, which represent a sequence of operations, some or all of which can be implemented in hardware, software or a combination thereof. In the context of software, the blocks represent computer-executable instructions stored on one or more computer-readable media that, when executed by one or more processors, perform the recited operations. Generally, computer-executable instructions include routines, programs, objects, components, data structures and the like that perform...
particular functions or implement particular abstract data types. The order in which the operations are described should not be construed as a limitation. Any number of the described blocks can be combined in any order and/or in parallel to implement the process, or alternative processes, and not all of the blocks need be executed. For discussion purposes, the processes are described with reference to the architectures and environments described in the examples herein, although the processes may be implemented in a wide variety of other architectures or environments.

For purposes of discussion, certain acts of the processes are illustrated in columns beneath the devices or systems that perform these acts in the example implementation. For instance, acts performed by the original poster's computer 106, cloud resources 102, and other user devices 152 are shown beneath their respective images.

At 402, a new image is posted for cloud-based storage. In the example architecture of FIG. 1, the user 104 uses a user interface 108 to select and post an image via the computer 106. The image is transferred from the computer 106 over the network 114 to the cloud resources 102. The upload servers 120 receive the image and move it to the backend content storage 124.

At 404, the content storage 124 of the cloud resources 102 stores a single copy of the image, in full resolution, within persistent storage. With reference to FIG. 1, this may be accomplished by storing the full resolution image 112 in the datastore 130. Optionally, at 406, the image may be compressed prior to storage. These may be accomplished, for example, by the compression module 132 using any number of known compression algorithms. Additionally, at 408, the boundary markers may be added to the image prior to storage. For instance, markers may be added to define possible segmentations of the image for parallel processing. The marker module 134 executing on the backend servers 128 may be used to place these markers during ingestion of the image.

At 410, a request for the image is received from a user (e.g., user 1) via a client device (e.g., client device 1). The request is transmitted over the network 154 to the content delivery network 122. At 412, the request is received at the cloud resources 102, and namely the CDN 122.

At 414, the display capabilities of the client device are ascertained. There are multiple ways to discern the display capabilities. In one approach, the request may include metadata describing the display capabilities or defining the size of the image need to fill an area of the screen. In another approach, a device type identifier module 144 may be employed to ascertain the display capabilities of the client device by retrieving the display capabilities based on an identifier of the client device or any other clues that may be present in the request.

At 416, the full resolution image is retrieved from the persistent storage. The image may be in a compressed or uncompressed format. At 418, the image may be cached in a short-term cache. The cache 146 is local to the CDN 122 and separate from the persistent storage, such as datastore 130. In this manner, subsequent requests for the image may be handled judiciously without fetching the image again from the backend content storage 124. The cached image may be subsequently terminated or deleted according to any type of protocols, such as time-based protocols, usage-based protocols, and so forth.

At 420, the image is resized using one or more graphic processing units to form a resized image at a resolution different from the full resolution, assuming the display capabilities do not support the image in the full resolution. As used herein, the display capabilities include both hardware capabilities (e.g., whether the display has sufficient pixel density to support high resolution images), software capabilities (e.g., size of the window or space allocated to display the image, such as a thumbnail, or larger location), or a combination of hardware and software capabilities. The resizing may include segmenting the image into multiple segments and processing the multiple segments in parallel with multiple GPUs. The resizing may also include decompressing the image if the image was retrieved in a compressed format.

Furthermore, the GPUs may perform other image processing tasks. For instance, the GPUs may perform such tasks as cropping, adding borders, modifying backgrounds, altering color, and so forth.

At 422, the resized image is served back to the requesting device. At 424, the resized image is received and rendered on the requesting client device.

FIG. 5 continues the process 400 for subsequent requests of the image, which is being temporarily stored in the cache. At 502, a request for the image is received from another user (e.g., user 2) via a different or second client device (e.g., client device 2). The request is transmitted over the network 154 to the content delivery network 122. At 504, the request is received at the cloud resources 102, and namely the CDN 122.

At 506, the display capabilities of the second client device are ascertained. At 508, the full resolution image is retrieved from the local cache at the CDN 122, rather than fetching the image from the backend content storage 124. In this way, the time to retrieve the image is reduced. At 510, the image is resized using one or more GPUs to form yet another resized image that might be at a resolution different from the full resolution or the first resolution served to the first client device.

At 512, the resized image is served back to the requesting device. At 514, the resized image is received and rendered on the requesting client device.

The process 400 in FIGS. 4 and 5 allows for a single copy of an image to be stored in a persistent datastore of the backend content storage, rather than dozens of copies at different resolutions. Upon receiving a request for the image, the process 400 allows for resizing of the images on the fly using high-powered GPUs at the edge of the cloud resources. Subsequent requests for the same image may then be resized on the fly using the GPUs without even having to fetch the image from the datastore. This allows for reduced storage costs, while avoiding any notable increase in latency and in some cases reducing latency.

Second Example System Architecture and Process

In the architecture described above, the image is stored once and resized in real-time by GPUs near the edge of the cloud services 102. The bandwidth within the cloud network (not shown) between the frontend servers and backend servers is typically very high relative to the bandwidth of the distribution network 154 between the cloud resources and the client device. As a result, the real-time image processing is performed as close to the edge as possible, such as in the CDN 122. However, as bandwidth improves in the distribution networks 154, the image processing may be pushed closer to the end user. Accordingly, in the architecture described below, the image is resized after it is delivered to the client.
side over the network 154, such as by the client itself or a more proximal/local computing device that is equipped with GPUs. FIG. 6 shows another architecture of a computing system 600 that includes cloud-based storage and computing resources 602 that receive content from, and serve content to, client devices of various users. The cloud resources 602 are similar to the cloud resources 102 of FIG. 1, with the exception that the content delivery network 604 implements servers 606 that are not specially modified with many GPUs. Instead, the CDN 604 retrieves the full resolution image 112 from the backend content storage 124 and serves the full resolution image 112 over the network 154 to one or more local resources. In the illustrated example, the client device 608 may be equipped with its own GPU 610 that rescales the image in real-time as the image is received. Alternatively, a local area computing device 612 (e.g., a box, home-based server, entertainment device, etc.) may be equipped with a GPU 614 to rescale the image as appropriate for a local display. Once rescaled, the computing device 612 may transfer the rescaled image to the client device 608 via a wireless or wired connection.

FIG. 7 shows an example process 700 executed by various devices in the system 600 to store a single version of a content item, such as an image, and scale the content item after delivery to the customer. At 702, a request for the image is submitted by the user’s device, such as the tablet 608. The request may be transmitted over the network 154 to the CDN 604, which either fetches the full resolution image from the datastore 130 or from the local cache 140 if previously cached.

At 704, the request is received at the cloud resources, namely the CDN 604. The CDN 604 retrieves the full resolution image either from the persistent storage 130 or from the local cache 140 at 706. At 708, the full resolution image is served to over the distribution network 154 to the client side.

At 710, the full resolution image is received by either the client device 608 or a client-side computing device 612. Either or both of these devices are equipped with a GPU, such as GPUs 610 and 614. At 712, the client-side GPU rescales the image to fit the display space available for displaying the image. The rescaled image is then presented on the display, at 714.

Accordingly, the process 700 in FIG. 7 allows for a single copy of an image to be stored in a persistent datastore of the backend content storage, rather than dozens of copies at different resolutions. Upon receiving a request for the image, the process 700 transfers the full resolution image over the network to the client-side, where the image is subsequently rescaled using high-powered GPUs on the client side. This process allows for reduced storage costs, while avoiding any notable increase in latency.

It is further noted that aspects pertaining to image compression and use of boundary markers may be used in the process 700 in a manner similar to that described above with respect to process 400.

CONCLUSION

Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features and acts as described. Rather, the specific features and acts are disclosed as example forms of implementing the claims.

What is claimed is:
1. A method comprising:
   storing a single copy of an image in a first resolution within persistent storage without storing multiple copies of the image in different resolutions in the persistent storage; receiving, from a client device, a request for the image; ascertaining display capabilities of the client device; retrieving the image, in the first resolution, from the persistent storage;
   rescaling the image, using one or more graphic processing units, to form a rescaled image of a second resolution different from the first resolution in an event that the display capabilities do not support rendering of the image in the first resolution; and serving the rescaled image in the second resolution.
2. The method as recited in claim 1, wherein the first resolution is full resolution and the second resolution is less than full resolution.
3. The method as recited in claim 1, wherein receiving a request for the image comprises receiving a request that includes at least one of the display capabilities of the client device or a size that the image is to be rendered.
4. The method as recited in claim 1, wherein receiving a request for the image comprises receiving a request that includes an identifier of the client device, and ascertaining the display capabilities of the client device comprises retrieving the display capabilities based on the identifier of the client device.
5. The method as recited in claim 1, further comprising caching the image, in the first resolution, in a short-term cache separate from the persistent storage.
6. The method as recited in claim 1, wherein the client device comprises a client device having first display capabilities, and further comprising:
   receiving, from a second client device having second display capabilities different from the first display capabilities, a request for the image;
   ascertaining the second display capabilities of the second client device;
   rescaling the image from the short-term cache, using one or more graphic processing units, to form a rescaled image of a third resolution different from the first resolution and the second resolution in an event that the second display capabilities do not support rendering of the image in the first resolution; and serving the rescaled image in the third resolution to the second client device.
7. The method as recited in claim 5, further comprising deleting the image from the short-term cache.
8. The method as recited in claim 1, wherein storing a single copy of an image comprises compressing the image and storing the image in compressed form, and the method further comprising decompressing the image following retrieval.
9. The method as recited in claim 1, further comprising segmenting the image into multiple image segments and rescaling the image segments in parallel using multiple graphic processing units.
10. The method as recited in claim 1, further comprising:
   adding boundary markers to the image to delineate segments of the image.
segmenting the image, using the boundary markers, into multiple image segments; and resealing the image segments in parallel using multiple graphic processing units.

11. A method comprising:

- storing a copy of an image in a first resolution within persistent storage without storing multiple copies of the image in different resolutions in the persistent storage;
- receiving, from a client device, a request for the image;
- retrieving the image, in the first resolution, from the persistent storage;
- serving the image, in full resolution, to the client device where the client device is configured to rescale the image, using at least one graphic processing unit, to form a rescaled image of a second resolution different from the first resolution.

12. The method as recited in claim 11, wherein the first resolution of the image is full resolution and the second resolution of the rescaled image is less than full resolution.

13. The method as recited in claim 11, further comprising caching the image, in the first resolution, in a short-term cache separate from the persistent storage.

14. The method as recited in claim 13, further comprising subsequently deleting the image from the short-term cache.

15. A cloud-based computing system comprising:

- a persistent content storage to store displayable content items, wherein at least some of the displayable content items are stored as single copies in a first format without secondary versions of the displayable content items being stored in the same or different formats; and
- a content delivery network having multiple servers configured to communicate with the content storage, each of the multiple servers comprising multiple graphic processing units to process the displayable content items such that, upon receipt of a request from a client device for a displayable content item, the content delivery network being configured to ascertain rendering constraints of the client device and retrieve the displayable content item, in the first format, from the content storage, the content delivery network being further configured to transform the displayable content item, using the graphic processing units, into a transformed displayable content item of a second format different from the first format, wherein the second format is supported by the display capabilities of the client device, and the content delivery network is configured to serve the transformed displayable content item in the second format to the client device.

16. The cloud-based computing system as recited in claim 15, wherein the displayable content item comprises an image, the first format comprises full resolution, and the content delivery network is configured to rescale the image to a resolution that is different than full resolution.

17. The cloud-based computing system as recited in claim 15, wherein the displayable content item is compressed prior to storage in the content storage.

18. The cloud-based computing system as recited in claim 15, wherein the content delivery network further comprises a cache to temporarily store the displayable content item in the first format.

19. The cloud-based computing system as recited in claim 18, wherein upon receipt of a second request for the content item from a second client device different than the first client device, the content delivery network is configured to ascertain display capabilities of the second client device and retrieve the displayable content item, in the first format, from the cache, the content delivery network being further configured to transform the displayable content item retrieved from the cache, using the graphic processing units, to form a second transformed displayable content item of a third format different from the first and second formats, wherein the third format is supported by the display capabilities of the second client device, and the content delivery network is configured to serve the second transformed displayable content item in the third format to the second client device.

20. The cloud-based computing system as recited in claim 15, further comprising a computing unit configured to add boundary markers to the content item to delineate multiple segments of the content item, and wherein the content delivery network is configured to segment, using the boundary markers, the displayable content item into multiple content segments and rescale the content segments in parallel using the graphic processing units.

21. One or more computer-readable media maintaining instructions executable by one or more processors of a computing system to perform operations comprising:

- storing a single copy of a displayable content item within persistent storage without storing multiple copies of the displayable content item in the persistent storage;
- receiving, from a client device, a request for the displayable content item;
- retrieving the displayable content item from the persistent storage;
- altering the displayable content item in real-time to form a second version of the displayable content item that is in a format suitable for presentation on the client device; and
- serving the second version of the displayable content item to the client device.

22. The computer-readable media as recited in claim 21, further comprising instructions executable by the one or more processors of the computing system to perform one or more additional operations comprising: caching the displayable content item, in the first format, in a short-term cache separate from the persistent storage.

23. The computer-readable media as recited in claim 22, further comprising instructions executable by the one or more processors of the computing system to perform one or more additional operations comprising:

- receiving, from a second client device having second display capabilities different than the first display capabilities, a request for the displayable content item;
- altering the displayable content item in real-time to form a third version of the displayable content item that is in a format suitable for presentation on the second client device; and
- serving the third version of the displayable content item to the second client device.

24. One or more computer-readable media maintaining instructions executable by one or more processors of a computing system to perform operations comprising:

- storing a single copy of an image within persistent storage without storing multiple copies of the image in the persistent storage;
- adding boundary markers to the image that define segments of the image;
- receiving, from a client device, a request for the image;
- retrieving the image from the persistent storage;
segmenting the image, using the boundary markers, into multiple image segments;
rescaling the image segments in parallel using multiple graphic processing units;
reforming the rescaled image segments to form a rescaled image; and
serving the rescaled image to the client device.

25. The computer-readable media as recited in claim 24, further comprising instructions executable by the one or more processors of the computing system to perform one or more additional operations comprising storing the single copy of the image in a compressed form and decompressing the image after retrieving the image from the persistent storage.