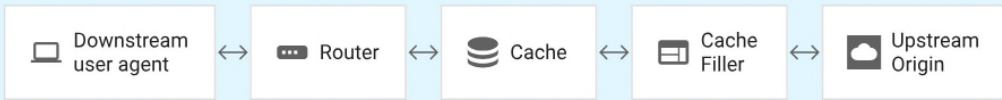


Exhibit D

U.S. Patent No. 10,924,573

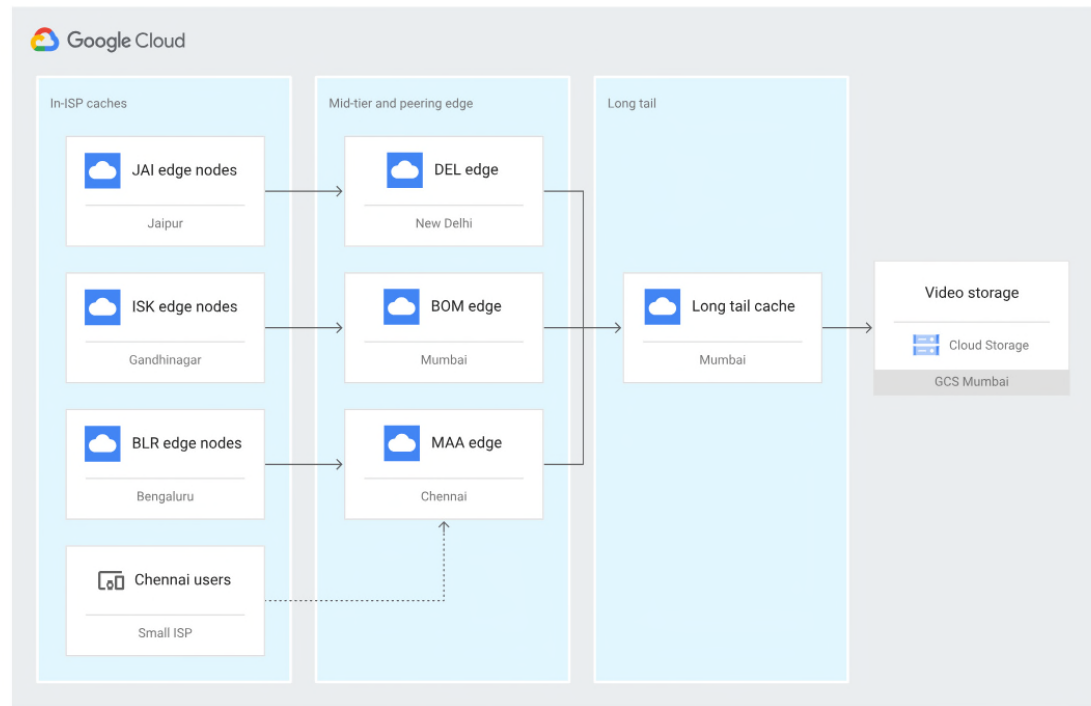
Sandpiper CDN

10,924,573		Google CDN
Claim 1		
1[p]	1. A method of content delivery in a content delivery network comprising:	Google uses and provides content delivery network(s) (“Google CDN”), including Google’s internal services and services Google offers to third parties that are used to provide content, such as webpages and/or video streams, over a network. Google CDN includes services such as Cloud DNS, for example, along with Google’s own use of technologies to provide content in a network. Google Cloud DNS is a scalable, reliable and managed authoritative Domain Name System (DNS) service running on the same infrastructure as Google. It has low latency, high availability and is a cost-effective way to make your applications and services available to your users. https://console.cloud.google.com/marketplace/product/google-cloud-platform/cloud-dns Use Google’s infrastructure for production quality and high-volume authoritative DNS serving. Your users will have reliable, low-latency access from anywhere in the world using our anycast name servers. Cloud DNS can scale to large numbers of DNS zones and records. You can reliably create and update millions of DNS records. Our name servers automatically scale to handle query volume. https://cloud.google.com/dns?hl=en
1(a)	receiving, at a first server of a first tier of servers of the content delivery network, a request from a requesting device for a resource available	Google receives, at a first server of a first tier of servers of the content delivery network, a request from a requesting device for a resource available from the content delivery network. For example, Google CDN receives requests for content available on the content delivery network at a first tier of servers. As one example, Google provides a method of content delivery that includes receiving a request from a device for a resource available from a content delivery network at a server in a first tier of the

	from the content delivery network;	network. For example, a server in the Google Media CDN “edge” (e.g., “EdgeCacheService”), as discussed below, includes a server in a first tier that receives client HTTP requests. You can control how content is cached for each URI you serve in a <i>route</i>. Using a route lets you optimize behavior based on the type of content, client attributes, and your freshness requirements for each route you define with Media CDN. The router and cache component are configured with the IP addresses and protocols that they use, security and cache policies, and a routing configuration. The configuration is contained in an <code>EdgeCacheService</code> resource. The router uses the <code>HeaderAction</code>, described in <code>EdgeCacheService</code>, to determine how the router modifies requests (including header manipulation and path rewrites). Media CDN terminates user HTTP requests at a reverse proxy. The reverse proxy matches the HTTP requests to a <code>Routing configuration</code>. The routing configuration defines how HTTP requests are routed, modified, cached, and filled. Media CDN logs each HTTP request between the client and edge, and between Media CDN and the origin, to <code>Cloud Logging (Logging)</code>. Logs are  <pre> graph LR A[Downstream user agent] <--> B[Router] B <--> C[Cache] C <--> D[Cache Filler] D <--> E[Upstream Origin] </pre> https://cloud.google.com/media-cdn/docs/overview
--	---	--

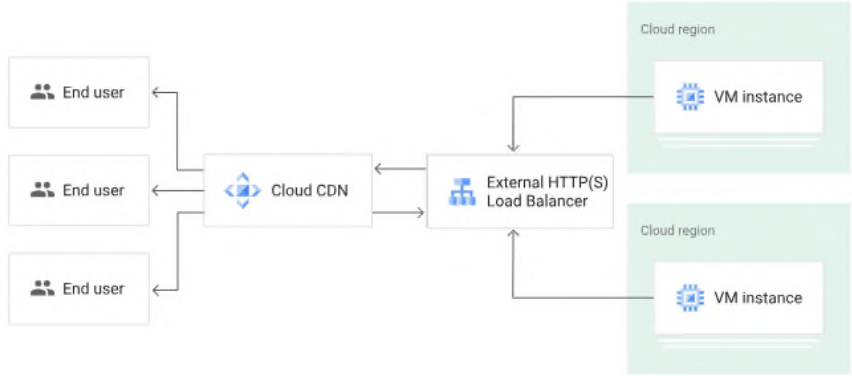
		The Google CDN includes a multi-tiered network with a “long tail” cache at the top of the hierarchy. The Google CDN includes long-tail caches with “substantial storage capacity,” indicating that a mid-tier and/or edge nodes in the Google CDN have less storage capacity. Origin shielding ⇄ Media CDN provides a deeply tiered edge infrastructure that is designed to actively minimize cache fill wherever possible. There are three primary layers of caching infrastructure: 1. Deep edge caches, which serve the majority of traffic and off-load within a service providers network. 2. Google's peering edge, which is connected to thousands of ISPs and acts as the mid-tier cache for edge caches, and for cases where those are not present within a given ISP, the user-facing cache. 3. Long-tail caches within Google's network that other downstream caches fill from prior to the origin. These caches support significant fan-in, have substantial cache storage capacity, and act as an origin <i>shield</i>.
--	--	--

An overview of this topology is shown here:



Topology overview (click to enlarge).

		All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google's own edge infrastructure is entirely over Google's global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding In an example, Google CDN receives requests from end users to the CDN platform, those requests are then distributed to resource servers that host the requested content through a load balancer that content once retrieved is then sent back to the end users on the network.
--	--	--

		Cloud CDN overview 🔖 Send feedback Cloud CDN (Content Delivery Network) uses Google's global edge network to serve content closer to users, which accelerates your websites and applications. Cloud CDN works with the global external Application Load Balancer or the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests. Cloud CDN content can be sourced from various types of backends. In Cloud CDN, these backends are also called <i>origin servers</i>. Figure 1 illustrates how responses from origin servers that run on virtual machine (VM) instances flow through an external Application Load Balancer before being delivered by Cloud CDN. In this situation, the Google Front End (GFE) comprises Cloud CDN and the external Application Load Balancer.  Figure 1. Responses flow from origin servers through Cloud CDN to clients. https://cloud.google.com/cdn/docs/overview
1(b)	accessing a popularity service associated with the	Google accesses a popularity service associated with the content delivery network to determine a popularity designation associated with the requested resource.

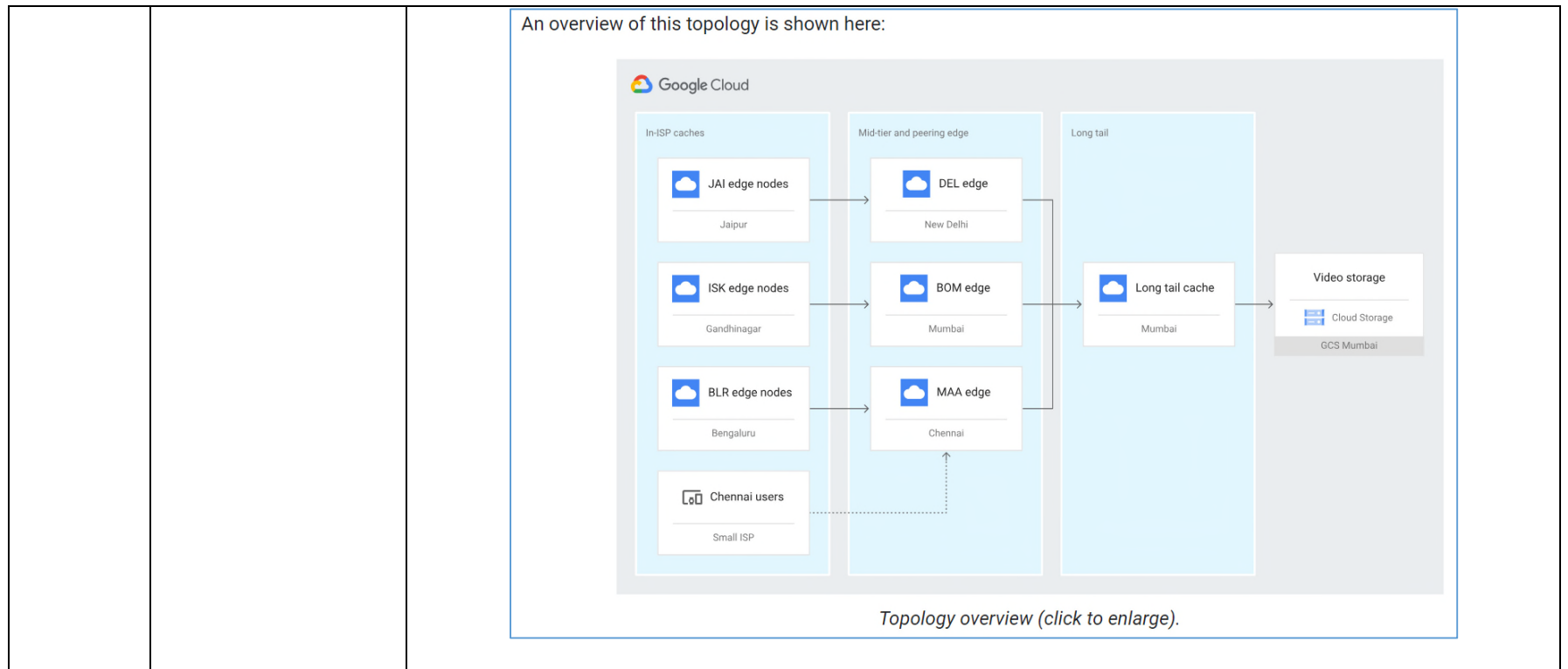
	content delivery network to determine a popularity designation associated with the requested resource;	For example, Google Media CDN caches “the most popular content” closer to users, while eventually evicting “unpopular content.” Google CDN analyzes the “relative’ popularity of content. As one specific example, on information and belief, the Google CDN includes caches that “cross-fill from each other where advantageous to do so,” indicating a popularity is determined. All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google’s own edge infrastructure is entirely over Google’s global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding
--	---	---

		Revalidation, eviction, and expiry Content delivery networks, including Media CDN, operate by caching the most popular content as close to users as possible. Media CDN's extensive storage, as well as origin shielding, limits the need to evict even unpopular content. Content that is accessed a small number of times per day might eventually be evicted. https://cloud.google.com/media-cdn/docs/caching the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers. <i>Unpopular content</i> means content that hasn't been accessed in a while. A <i>while</i> and <i>unpopular</i> are both relative to the bulk of other items in the cache. Multiple Google Cloud projects share a common pool of cache space because the projects are served from the same set of GFEs. The relative popularity of content is compared across multiple projects, not only within a single project. https://cloud.google.com/cdn/docs/overview
--	--	--

		Time to live (TTL) overrides let you set default TTL values for cached content and override TTL values set in the <code>max-age</code> and <code>s-maxage</code> cache control directives (or <code>Expires</code> headers) set by your origins. TTLs, whether set by overrides or by using a cache directive, are optimistic. Content that is rarely accessed or unpopular might be evicted from the cache prior to the TTL being reached. https://cloud.google.com/media-cdn/docs/caching#cache-modes Content delivery networks, including Media CDN, operate by caching the most popular content as close to users as possible. Media CDN's extensive storage, as well as origin shielding, limits the need to evict even unpopular content. Content that is accessed a small number of times per day might eventually be evicted. • Cached responses that reach their configured TTL might not be immediately evicted. For popular content, Media CDN revalidates that the cached response is the latest version by issuing a <code>HEAD</code> request to the origin to confirm that the headers have not changed. Under some circumstances, Media CDN instead sends a request to the origin with an <code>If-None-Match</code> and/or <code>If-Modified-Since</code> request header. In this case, correctly configured origins should return an HTTP 304 (Not Modified) response, without the body bytes, if the cache has the "latest" copy of that response. https://cloud.google.com/media-cdn/docs/caching#cache-modes As another specific example, Google Cloud CDN utilizes caching services to store content in edge servers closer to end users. Popular content which is accessed more frequently is more likely to be stored in cache servers. Cached content which are not accessed frequently are evicted from cache servers after a maximum of 30 days.
--	--	---

		Serving content from a cache After you enable Cloud CDN, caching happens automatically for all cacheable content. Your origin server uses HTTP headers to indicate which responses are cached. You can also control cacheability by using cache modes. When you use a backend bucket, the origin server is Cloud Storage. When you use VM instances, the origin server is the web server software that you run on those instances. Cloud CDN uses caches in numerous locations around the world. Because of the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers. For more information about what Cloud CDN caches and for how long, see the Caching overview. To see what Cloud CDN is serving from a cache, you can view logs. https://cloud.google.com/cdn/docs/overview#serving-from-cache Cache requests that include additional HTTP headers or HTTP cookies in the request are cached on the third request in a cache location for that cache key. This reduces the impact of high-cardinality header or cookie values on your cache eviction rates. Under normal circumstances and user traffic conditions, this shouldn't be noticeable and helps ensure that popular content remains cached. https://cloud.google.com/cdn/docs/caching#http-headers-cookies If you are using Cloud Storage and serving web content from your bucket, you can use Cloud CDN in front of your storage bucket to set web security headers and cache popular content. https://cloud.google.com/cdn/docs/web-security-best-practices#best_practices_for_web_security
--	--	---

		Eviction There is no guarantee that a cache entry remains in the cache until it expires because unpopular entries can be evicted before they expire <i>at any time</i> to make room for new content. As an upper bound, cache entries that aren't accessed for 30 days are automatically evicted. For more information, see Eviction and expiration. https://cloud.google.com/cdn/docs/caching#eviction
1(c)	requesting the resource from a second server of the content delivery network;	Google requests the resource from a second server of the content delivery network. Origin shielding ⇄ Media CDN provides a deeply tiered edge infrastructure that is designed to actively minimize cache fill wherever possible. There are three primary layers of caching infrastructure: 1. Deep edge caches, which serve the majority of traffic and off-load within a service providers network. 2. Google's peering edge, which is connected to thousands of ISPs and acts as the mid-tier cache for edge caches, and for cases where those are not present within a given ISP, the user-facing cache. 3. Long-tail caches within Google's network that other downstream caches fill from prior to the origin. These caches support significant fan-in, have substantial cache storage capacity, and act as an origin <i>shield</i>.



		All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google's own edge infrastructure is entirely over Google's global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding As another example, Google Cloud CDN requests content from secondary servers. In an example, Google Cloud CDN utilizes backend services which provide information from the external servers hosting content the CDN is requesting. These backend services can reference a multitude of different resources from which the CDN can request content. Setup overview  Cloud CDN works with the global external Application Load Balancer and the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends (or <i>origins</i>) that respond to the requests. Cloud CDN content can be sourced from various types of backends. https://cloud.google.com/cdn/docs/using-cdn
--	--	--

Backend services 

A backend service provides configuration information to the load balancer so that it can direct requests to its backends—for example, Compute Engine instance groups or network endpoint groups (NEGs). For more information about backend services, see [Backend services overview](#).

For an example showing how to set up a load balancer with a backend service and a Compute Engine backend, see [Setting up an external Application Load Balancer with a Compute Engine backend](#).

Backend service scope

The following table indicates which backend service resource and scope is used by external Application Load Balancers:

Load balancer mode	Backend service resource
Global external Application Load Balancer	backendServices (global)
Classic Application Load Balancer	backendServices (global)
Regional external Application Load Balancer	regionBackendServices (regional)

<https://cloud.google.com/load-balancing/docs/https#backend-service>

		A backend service defines how Cloud Load Balancing distributes traffic. The backend service configuration contains a set of values, such as the protocol used to connect to backends, various distribution and session settings, health checks, and timeouts. These settings provide fine-grained control over how your load balancer behaves. To get you started, most of the settings have default values that allow for fast configuration. A backend service is either global or regional in scope. Load balancers, Envoy proxies, and proxyless gRPC clients use the configuration information in the backend service resource to do the following: • Direct traffic to the correct <i>backends</i>, which are instance groups or network endpoint groups (NEGs). • Distribute traffic according to a <i>balancing mode</i>, which is a setting for each backend. • Determine which health check is monitoring the health of the backends. • Specify <i>session affinity</i>. • Determine whether other services are enabled, including the following services that are only available for certain load balancers: • Cloud CDN • Google Cloud Armor security policies • Identity-Aware Proxy • Designate regional backend services as a service in App Hub, which is in preview. You set these values when you create a backend service or add a backend to the backend service. https://cloud.google.com/load-balancing/docs/backend-service
1(d)	processing, at the first server of the first tier of servers, a redirect instruction from the second server of the content delivery network to obtain	Google processes, at the first server of the first tier of servers, a redirect instruction from the second server of the content delivery network to obtain the resource from a content server of the content delivery network. For example, Google CDN processes requests for content at the first server tier (<i>e.g.</i>, the CDN server). Upon receiving the request, Google CDN, performs a redirect from another server, which matches the request URL to a content server (<i>e.g.</i>, the designated host of the content).

	the resource from a content server of the content delivery network;	Redirect requests Media CDN supports three types of redirects: • Host redirects, which redirect only the host (domain), keeping the path and query parameters unchanged. • Path redirects, which replace the path in full. • Path prefix redirects, which only replace the matching prefix. Redirects default to <code>HTTP 301 (Moved Permanently)</code>, but can be configured to return different redirect status codes on a per-route basis. The following configuration is an example of a prefix-based redirect, where you redirect users visiting <code>https://cdn.example.com/on-demand/*</code> to <code>https://cdn.example.com/streaming/*</code>. <pre>name: prod-service routing: hostRules: - hosts: - cdn.example.com pathMatcher: example_routes pathMatchers: - name: example_routes routeRules: - priority: 10 matchRules: - prefixMatch: "/on-demand/" urlRedirect: # The prefix matched in matchRules.prefixMatch is replaced # by this value prefixRedirect: "/streaming/" redirectResponseCode: TEMPORARY_REDIRECT # corresponds to a HTTP 307</pre> https://cloud.google.com/media-cdn/docs/routing#redirect-requests
--	--	--

URL redirects

A *URL redirect* redirects your domain's visitors from one URL to another.

A *default URL redirect* is not conditioned on matching any particular pattern in the incoming request. For example, you might want to use a default URL redirect to redirect *any* hostname to a hostname of your choice.

There are several ways to create a URL redirect, as outlined in the following table.

Method	Configuration
URL map's default URL redirect	Top-level <code>defaultUrlRedirect</code>
A path matcher's default URL redirect	<code>pathMatchers[] .defaultUrlRedirect[]</code>
A path matcher's path rule's URL redirect	<code>pathMatchers[] .pathRules[] .urlRedirect</code>
A path matcher's route rule's URL redirect	<code>pathMatchers[] .routeRules[] .urlRedirect</code>

Inside of a `defaultUrlRedirect` or `urlRedirect`, `pathRedirect` always works as follows:

- The entire request path is replaced with the path you specify.

Inside of a `defaultUrlRedirect` or `urlRedirect`, how the `prefixRedirect` works depends on how you use it:

- If you use a `defaultUrlRedirect`, `prefixRedirect` is effectively a prefix prepend because the matched path is always `/`.
- If you use a `urlRedirect` within a path matcher's route rule or path rule, `prefixRedirect` is a prefix replacement based on how the requested path was matched *as defined in the path rule or route rule*.

<https://cloud.google.com/load-balancing/docs/url-map-concepts#url-redirects>

		Use third-party storage backends Media CDN supports connecting to publicly reachable HTTP endpoints outside of Google Cloud, including AWS S3 storage buckets, Azure Blob Storage, and other storage providers. This can be useful if you have a multicloud architecture, or are yet to migrate data to Cloud Storage using the Storage Transfer Service. A minimal origin configuration that configures a virtual-hosted bucket in AWS S3: <pre>name: MY_S3_ORIGIN originAddress: BUCKET-NAME.s3.REGION.amazonaws.com</pre> If you aren't using a bucket name that matches the hostnames configured for your <code>EdgeCacheService</code> resources, you must also configure a host rewrite for routes associated with this origin (or origins). Otherwise, the Host header set by the client request is used when fetching from the origin. For example, to map all requests with a path prefix of <code>/legacy/</code> to your external bucket, you can configure both a <code>hostRewrite</code> and a <code>pathPrefixRewrite</code> to strip this prefix from the origin request: <pre>routeRules: - description: legacy backend matchRules: - prefixMatch: "/legacy/" routeAction: urlRewrite: hostRewrite: BUCKET-NAME.s3.REGION.amazonaws.com pathPrefixRewrite: / cdnPolicy: cacheMode: CACHE_ALL_STATIC defaultTtl: 3600s</pre> https://cloud.google.com/media-cdn/docs/routing#third-party-buckets
1(e)	receiving an instruction to not cache the portion of the resource at the	Google receives an instruction to not cache the portion of the resource at the first server of the first tier of servers when the portion of the resource is obtained from the content server of the content delivery network.

	first server of the first tier of servers when the portion of the resource is obtained from the content server of the content delivery network; and	For example, Google receives instructions to not cache, for example in Media CDN, when caches “cross-fill from each other,” a cache providing a resource includes at least some popularity information for a receiving cache to be used to decide whether to cache the resource or not in response to an instruction. All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google's own edge infrastructure is entirely over Google's global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding
--	--	--

		Revalidation, eviction, and expiry Content delivery networks, including Media CDN, operate by caching the most popular content as close to users as possible. Media CDN's extensive storage, as well as origin shielding, limits the need to evict even unpopular content. Content that is accessed a small number of times per day might eventually be evicted. https://cloud.google.com/media-cdn/docs/caching As another example, Google receives an instruction to not cache content if the cache request is not yet a third request. In these cases, the requested resource is obtained from a content server of the CDN. Cache requests that include additional HTTP headers or HTTP cookies in the request are cached on the third request in a cache location for that cache key. This reduces the impact of high-cardinality header or cookie values on your cache eviction rates. Under normal circumstances and user traffic conditions, this shouldn't be noticeable and helps ensure that popular content remains cached. https://cloud.google.com/cdn/docs/caching#http-headers-cookies
1(f)	providing the obtained resource to the requesting device.	Google provides the requested resources to the requesting device, for example via Google Media CDN. Media CDN uses YouTube's infrastructure to bring video streams (VoD and live) and large file downloads closer to users for fast and reliable delivery. https://cloud.google.com/cdn?hl=en

		In another example, when the requested content is not stored on an edge cache server (e.g., a cache miss), Google CDN retrieves data from the origin server. How Cloud CDN works When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user. If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN. Cache hits and cache misses A cache is a group of servers that stores and manages content so that future requests for that content can be served faster. The cached content is a copy of cacheable content that is stored on origin servers. If the GFE looks in the Cloud CDN cache and finds a cached response to the user's request, the GFE sends the cached response to the user. This is called a <i>cache hit</i>. When a cache hit occurs, the GFE looks up the content by its cache key and responds directly to the user, shortening the round-trip time and saving the origin server from having to process the request. A <i>partial hit</i> occurs when a request is served partially from cache and partially from a backend. This can happen if only part of the requested content is stored in a Cloud CDN cache, as described in Support for byte range requests. The first time that a piece of content is requested, the GFE determines that it can't fulfill the request from the cache. This is called a <i>cache miss</i>. When a cache miss occurs, the GFE forwards the request to the external Application Load Balancer. The load balancer then forwards the request to one of your origin servers. When the cache receives the content, the GFE forwards the content to the user. If the origin server's response to this request is cacheable, Cloud CDN stores the response in the Cloud CDN cache for future requests. Data transfer from a cache to a client is called <i>cache egress</i>. Data transfer to a cache is called <i>cache fill</i>. https://cloud.google.com/cdn/docs/overview#how-cdn-works
--	--	--

Claim 2		
2.	2. The method as recited in claim 1 wherein the popularity designation associated with the requested resource indicates that the requested resource is not popular.	Google's popularity designation associated with the requested resource indicates that the requested resource is not popular. For example, resources which are of lower popularity are slated for eviction or expiration based on caching configurations. A resource which has a lower popularity will be evicted from cache server. The popularity designation of content can be monitored based on the amount times that a resource endpoint was requested Monitoring Cloud CDN example queries Monitoring lets you create custom dashboards. Dashboards can use any of the monitoring metrics for external Application Load Balancers. Following are some example MQL snippets that you can paste into custom Monitoring dashboards. Request bytes count broken out by cache result This query focuses on backends that have Cloud CDN enabled, which is done by including <code>filter (metric.cache_result != 'DISABLED')</code>. <pre>fetch https_lb_rule metric 'loadbalancing.googleapis.com/https/response_bytes_count' filter (metric.cache_result != 'DISABLED') align rate(1m) every 1m group_by [metric.cache_result], [value_response_bytes_count_aggregate: aggregate(value.response_bytes_count)]</pre> https://cloud.google.com/cdn/docs/logging#monitoring-queries

		Client roundtrip TCP latency at 95% for a specific backend target This query includes <code>filter (resource.backend_target_name = 'example-backend')</code>, which narrows down traffic to the backend <code>example-backend</code>. A backend can be a Cloud Storage bucket, a Compute Engine VM group, or an external backend. <pre> fetch https_lb_rule metric 'loadbalancing.googleapis.com/https/frontend_tcp_rtt' filter (resource.backend_target_name = 'example-backend') group_by 1m, [value_frontend_tcp_rtt_aggregate: aggregate(value_frontend_tcp_rtt)] every 1m group_by [metric.proxy_continent], [value_frontend_tcp_rtt_aggregate_percentile: percentile(value_frontend_tcp_rtt_aggregate, 95)] </pre> Request count broken out by response code class for Cloud CDN-enabled backends This query breaks out traffic by response code class (2xx, 3xx, 4xx, 5xx) to help separate client successes, client errors, and server errors. <pre> fetch https_lb_rule metric 'loadbalancing.googleapis.com/https/request_count' filter (metric.cache_result != 'DISABLED') group_by 1h, [row_count: row_count()] every 1h group_by [metric.response_code_class], [row_count_aggregate: aggregate(row_count)] </pre> https://cloud.google.com/cdn/docs/logging#monitoring-queries
Claim 3		
3.	3. The method as recited in claim 1 wherein the popularity	Google's popularity designation associated with the requested resource indicates that the requested resource is popular, and Google receives an instruction to cache the resource at the first server of the first tier of servers when the resource is obtained from the content server of the content delivery network.

	designation associated with the requested resource indicates that the requested resource is popular, the method further comprising: receiving an instruction to cache the resource at the first server of the first tier of servers when the resource is obtained from the content server of the content delivery network.	See, e.g., evidence of use for claim element 1 and the following. Google CDN caches resources via a popularity designation, for example, based on access frequency. Popular resources are cached in the first tier of servers where subsequent requests for said resource are retrieved. Serving content from a cache After you enable Cloud CDN, caching happens automatically for all cacheable content. Your origin server uses HTTP headers to indicate which responses are cached. You can also control cacheability by using cache modes. When you use a backend bucket, the origin server is Cloud Storage. When you use VM instances, the origin server is the web server software that you run on those instances. Cloud CDN uses caches in numerous locations around the world. Because of the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers. For more information about what Cloud CDN caches and for how long, see the Caching overview. To see what Cloud CDN is serving from a cache, you can view logs. https://cloud.google.com/cdn/docs/overview#serving-from-cache Cache requests that include additional HTTP headers or HTTP cookies in the request are cached on the third request in a cache location for that cache key. This reduces the impact of high-cardinality header or cookie values on your cache eviction rates. Under normal circumstances and user traffic conditions, this shouldn't be noticeable and helps ensure that popular content remains cached. https://cloud.google.com/cdn/docs/caching#http-headers-cookies
--	---	---

		Content which is more popular will receive a higher “cache hit ratio”, where content which is more popular is designated by a cache key which expedites future retrievals of said content. Cache hits and cache misses ↗ A cache is a group of servers that stores and manages content so that future requests for that content can be served faster. The cached content is a copy of cacheable content that is stored on origin servers. If the GFE looks in the Cloud CDN cache and finds a cached response to the user's request, the GFE sends the cached response to the user. This is called a <i>cache hit</i>. When a cache hit occurs, the GFE looks up the content by its cache key and responds directly to the user, shortening the round-trip time and saving the origin server from having to process the request. A <i>partial hit</i> occurs when a request is served partially from cache and partially from a backend. This can happen if only part of the requested content is stored in a Cloud CDN cache, as described in Support for byte range requests. The first time that a piece of content is requested, the GFE determines that it can't fulfill the request from the cache. This is called a <i>cache miss</i>. When a cache miss occurs, the GFE forwards the request to the external Application Load Balancer. The load balancer then forwards the request to one of your origin servers. When the cache receives the content, the GFE forwards the content to the user. If the origin server's response to this request is cacheable, Cloud CDN stores the response in the Cloud CDN cache for future requests. Data transfer from a cache to a client is called <i>cache egress</i>. Data transfer to a cache is called <i>cache fill</i>.
--	--	---

Figure 2 shows a cache hit and a cache miss:

1. Origin servers running on VM instances send HTTP(S) responses.
2. The external Application Load Balancer distributes the responses to Cloud CDN.
3. Cloud CDN delivers the responses to end users.

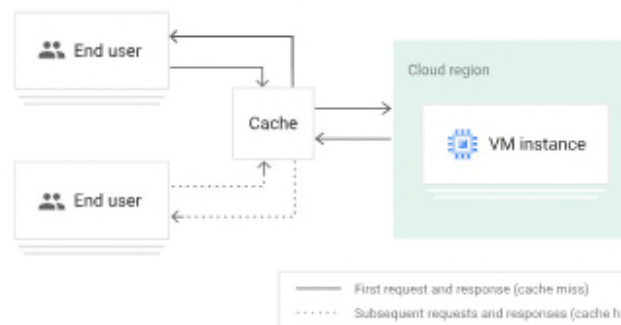


Figure 2. The initial response is served by the origin server while subsequent responses are served by the GFE from cache.

https://cloud.google.com/cdn/docs/overview#cache_hits_and_cache_misses

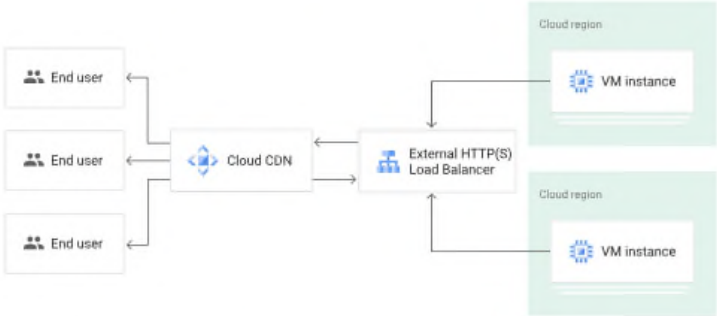
Cache hit ratio

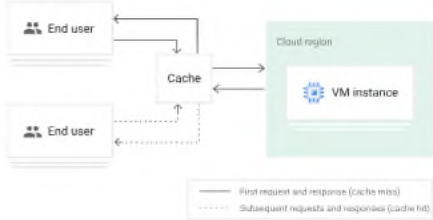
The *cache hit ratio* is the percentage of times that a requested object is served from the cache. If the cache hit ratio is 60%, it means that the requested object is served from the cache 60% of the time and must be retrieved from the origin 40% of the time.

For information about how cache keys can affect the cache hit ratio, see [Using cache keys](#). For troubleshooting information, see [Cache hit ratio is low](#).

https://cloud.google.com/cdn/docs/overview#cache_hit_ratio

Claim 6		
6.	6. The method as recited in claim 1 wherein the popularity designation associated with the requested resource comprises an indication whether a current popularity value for the resource exceeds a first predetermined popularity threshold.	Upon information and belief, Google’s popularity designation associated with the requested resource comprises an indication whether a current popularity value for the resource exceeds a first predetermined popularity threshold. For example, content satisfying a popularity threshold based on frequent access is more likely to be stored in Google’s cache servers. Cached content which are not accessed frequently are evicted from cache servers after a threshold period of days, such as 30 days. <i>See, e.g.,</i> evidence of use for claim element 1(b).
Claim 10		
10[p]	10. The method as recited in claim 3 further comprising:	<i>See, e.g.,</i> evidence of use for claim elements 1[p] and 3.
10(a)	receiving, at the first server of the first tier of servers of the content delivery network, an additional request for the resource available from the content delivery network;	Google CDN provides means to make subsequent requests for content from content sources such as servers in the first tier of a content delivery network (<i>e.g.,</i> edge cache servers).

		Cloud CDN overview 🔖 Send feedback Cloud CDN (Content Delivery Network) uses Google's global edge network to serve content closer to users, which accelerates your websites and applications. Cloud CDN works with the global external Application Load Balancer or the classic Application Load Balancer to deliver content to your users. The external Application Load Balancer provides the frontend IP addresses and ports that receive requests and the backends that respond to the requests. Cloud CDN content can be sourced from various types of backends. In Cloud CDN, these backends are also called <i>origin servers</i>. Figure 1 illustrates how responses from origin servers that run on virtual machine (VM) instances flow through an external Application Load Balancer before being delivered by Cloud CDN. In this situation, the Google Front End (GFE) comprises Cloud CDN and the external Application Load Balancer.  Figure 1. Responses flow from origin servers through Cloud CDN to clients. How Cloud CDN works When a user requests content from an external Application Load Balancer, the request arrives at a GFE that is at the edge of Google's network as close as possible to the user. If the load balancer's URL map routes traffic to a backend service or backend bucket that has Cloud CDN configured, the GFE uses Cloud CDN. https://cloud.google.com/cdn/docs/overview
--	--	---

10(b)	determining that the resource is cached at the first server of the first tier of servers; and	Google CDN determines when data is cached on the first server in a first tier of servers. For example, cached content is stored on a first tier of servers, such as on an edge cache, when the content is popular and cacheable. Google CDN determines if this content is in the cache and designates that status through “cache hits” or “cache misses”. Cache hits and cache misses A cache is a group of servers that stores and manages content so that future requests for that content can be served faster. The cached content is a copy of cacheable content that is stored on origin servers. If the GFE looks in the Cloud CDN cache and finds a cached response to the user's request, the GFE sends the cached response to the user. This is called a <i>cache hit</i>. When a cache hit occurs, the GFE looks up the content by its <i>cache key</i> and responds directly to the user, shortening the round-trip time and saving the origin server from having to process the request. A <i>partial hit</i> occurs when a request is served partially from cache and partially from a backend. This can happen if only part of the requested content is stored in a Cloud CDN cache, as described in Support for byte range requests. The first time that a piece of content is requested, the GFE determines that it can't fulfill the request from the cache. This is called a <i>cache miss</i>. When a cache miss occurs, the GFE forwards the request to the external Application Load Balancer. The load balancer then forwards the request to one of your origin servers. When the cache receives the content, the GFE forwards the content to the user. If the origin server's response to this request is <i>cacheable</i>, Cloud CDN stores the response in the Cloud CDN cache for future requests. Data transfer from a cache to a client is called <i>cache egress</i>. Data transfer to a cache is called <i>cache fill</i>. Figure 2 shows a cache hit and a cache miss: 1. Origin servers running on VM instances send HTTP(S) responses. 2. The external Application Load Balancer distributes the responses to Cloud CDN. 3. Cloud CDN delivers the responses to end users.  https://cloud.google.com/cdn/docs/overview#cache_hits_and_cache_misses
-------	---	---

10(c)	providing the cached resource from the first server of the first tier of servers in response to the additional request for the resource.	Google CDN serves a cached resource from the first server of the first tier of servers when additional requests are made for a resource. When subsequent requests for the cached content is made, Google CDN will serve that content from said cache server. Serving content from a cache After you enable Cloud CDN, caching happens automatically for all cacheable content. Your origin server uses HTTP headers to indicate which responses are cached. You can also control cacheability by using cache modes. When you use a backend bucket, the origin server is Cloud Storage. When you use VM instances, the origin server is the web server software that you run on those instances. Cloud CDN uses caches in numerous locations around the world. Because of the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers. For more information about what Cloud CDN caches and for how long, see the Caching overview. To see what Cloud CDN is serving from a cache, you can view logs. Removing content from the cache To remove an item from a cache, you can invalidate cached content. For more information, see: • Cache invalidation overview • Invalidating cached content https://cloud.google.com/cdn/docs/overview#serving-from-cache
Claim 19		
19[p]	19. A method of content delivery in a content delivery	Google implements a method of content delivery in a content delivery network. <i>See e.g., evidence of use for claim elements 1[p] and 1(a).</i>

	network comprising:	
19(a)	receiving, at a first server of a first tier of servers of the content delivery network, a request from a requesting device for a resource available from the content delivery network;	<i>See, e.g.</i> , evidence of use for claim elements 1[p] and 1(a).
19(b)	accessing a popularity service associated with the first server of the first tier of the content delivery network to determine a popularity designation associated with the requested resource;	Google accesses a popularity service to determine a popularity designation associated with a requested resource. For example, Google Media CDN caches “the most popular content” closer to users, while eventually evicting “unpopular content.” Google CDN analyzes the “relative” popularity of content. <i>See, e.g.</i>, evidence of user for claim element 1(b) and the following. As one specific example, on information and belief, the Google CDN includes caches that “cross-fill from each other where advantageous to do so,” indicating a popularity is determined.

		All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google's own edge infrastructure is entirely over Google's global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding Revalidation, eviction, and expiry Content delivery networks, including Media CDN, operate by caching the most popular content as close to users as possible. Media CDN's extensive storage, as well as origin shielding, limits the need to evict even unpopular content. Content that is accessed a small number of times per day might eventually be evicted. https://cloud.google.com/media-cdn/docs/caching
--	--	---

		the nature of caches, it is impossible to predict whether a particular request is served out of a cache. You can, however, expect that popular requests for cacheable content are served from a cache most of the time, yielding significantly reduced latencies, reduced cost, and reduced load on your origin servers. <i>Unpopular content</i> means content that hasn't been accessed in a while. A <i>while</i> and <i>unpopular</i> are both relative to the bulk of other items in the cache. Multiple Google Cloud projects share a common pool of cache space because the projects are served from the same set of GFEs. The relative popularity of content is compared across multiple projects, not only within a single project. https://cloud.google.com/cdn/docs/overview Time to live (TTL) overrides let you set default TTL values for cached content and override TTL values set in the <code>max-age</code> and <code>s-maxage</code> cache control directives (or <code>Expires</code> headers) set by your origins. TTLs, whether set by overrides or by using a cache directive, are optimistic. Content that is rarely accessed or unpopular might be evicted from the cache prior to the TTL being reached. https://cloud.google.com/media-cdn/docs/caching#cache-modes Content delivery networks, including Media CDN, operate by caching the most popular content as close to users as possible. Media CDN's extensive storage, as well as origin shielding, limits the need to evict even unpopular content. Content that is accessed a small number of times per day might eventually be evicted. • Cached responses that reach their configured TTL might not be immediately evicted. For popular content, Media CDN revalidates that the cached response is the latest version by issuing a <code>HEAD</code> request to the origin to confirm that the headers have not changed. Under some circumstances, Media CDN instead sends a request to the origin with an <code>If-None-Match</code> and/or <code>If-Modified-Since</code> request header. In this case, correctly configured origins should return an HTTP 304 (Not Modified) response, without the body bytes, if the cache has the "latest" copy of that response. https://cloud.google.com/media-cdn/docs/caching#cache-modes
--	--	---

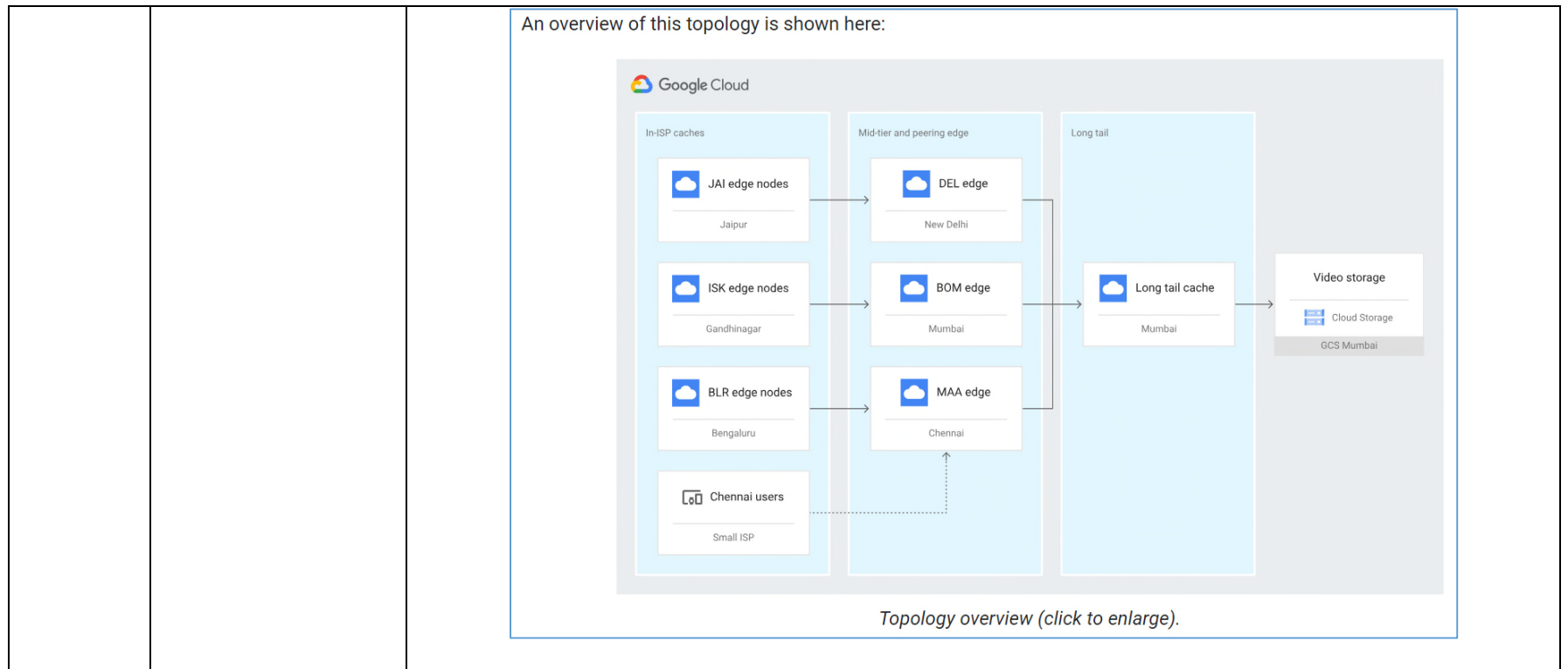
19(c)	receiving an instruction to not cache the resource at the first server of the first tier of servers when the portion of the resource is obtained from the content server of the content delivery network;	Google receives instructions to not cache, for example in Media CDN when caches “cross-fill from each other,” a cache providing a resource includes at least some popularity information for a receiving cache to be used to decide whether to cache the resource or not in response to an instruction. See e.g., evidence of user for claim element 1(e) and the following. All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google's own edge infrastructure is entirely over Google's global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding Revalidation, eviction, and expiry Content delivery networks, including Media CDN, operate by caching the most popular content as close to users as possible. Media CDN's extensive storage, as well as origin shielding, limits the need to evict even unpopular content. Content that is accessed a small number of times per day might eventually be evicted. https://cloud.google.com/media-cdn/docs/caching
-------	---	--

		In one example, Edge caches additionally can refuse to store partial content retrieved from origin servers if said content is non-cacheable, or if configured to refuse content. Support for byte range requests A response that satisfies the following criteria indicates that the origin server supports byte range requests: • Status code: 200 OK or 206 Partial Content • Header: Accept-Ranges: bytes • Header: Content-Length, and for a 206 Partial Content response, a Content-Range value that indicates the complete length of the origin object. For example, Content-length: 0-100/999 is cacheable whereas Content-length: 0-100/* isn't. • Header: Last-Modified and ETag with a strong validator. Cloud Storage supports byte range requests for most objects. However, Cloud Storage does not support byte range requests for objects with Content-Encoding: gzip metadata unless the client request includes an Accept-Encoding: gzip header. If you have Cloud Storage objects larger than 10 MB, make sure that they don't have Content-Encoding: gzip metadata. For information about how to edit object metadata, see Viewing and editing object metadata.
--	--	---

		Popular web server software also supports byte range requests. Consult the documentation for your web server for details about how to enable support. For more information about byte range requests, see the HTTP specification. When an origin server supports byte range requests, a Cloud CDN cache declines to store an otherwise cacheable response the first time it is requested if either of the following is true: • The response body is incomplete because the client requested only part of the content. • The response body is larger than 1 MB (1,048,576 bytes). When this happens and the response would otherwise satisfy the normal cacheability requirements, the cache records that the origin server supports byte range requests for that cache key and forwards the origin server's response to the client. On a cache miss, the cache checks whether the origin server is known to support byte range requests. If byte range requests are known to be supported for the cache key, the cache does not forward the client request to the external Application Load Balancer. Instead, the cache initiates its own byte range cache fill requests for the missing parts of the content. The origin server returns a <code>206 Partial Content</code> response when Cloud CDN initiates its own byte range cache fill request. For a <code>206 Partial Content</code> response to be considered while caching, it must include a <code>Content-Range</code> header with a <code>complete-length</code> directive that doesn't include asterisks, for example, <code>0-100/999</code>. Cloud CDN then caches that returned <code>206 Partial Content</code> response and uses it to respond to future client requests for that content. A cache stores a <code>206 Partial Content</code> response only when it is received in response to a byte range request that the cache initiated. Because a cache doesn't initiate a byte range request unless it had previously recorded that the origin server supports byte range requests for that cache key, a given cache doesn't store content that's larger than 1 MB until the second time that content is accessed. Due to its distributed nature, Cloud CDN might sometimes fetch the final chunk from the origin more than once per location. This only impacts the first few requests per cache key. https://cloud.google.com/cdn/docs/caching#byte-range-requests
19(d)	requesting the resource from a second server of the content delivery network based at least on the	Google requests a resource from a second server of the Google CDN based on a popularity designation. See <i>e.g.</i>, evidence of use for claim element 1(c) and the following.

	popularity designation associated with the requested resource; and	In one example, Google CDN caches content a first server in the geographical location closest to where the request is made. In one example, Google CDN stores a less popular resource on secondary servers (e.g., “long-tail” caches). These secondary servers generally store content which is valuable but not popular. Upon information and belief, Google CDN uses these servers to store such content. Long-tail content For long-tail cases, setting a single threshold becomes meaningless. Most of your content would never make it into the cache, and if it did, it would almost never be served from cache again. It would be soon replaced by content requested after it, pushing the hit ratio very low.  An example could be an educational platform with a wide range of subjects. These sites have enormous traffic, but there is no top content for the majority of users. Instead, people choose whatever they currently need from an extensive archive. https://www.cdn77.com/blog/tackle-long-tail-with-multilayer-cache
--	--	---

19(e)	providing the obtained resource to the requesting device.	Google provides a resource to the requesting device. For example, Google Media CDN provides the video streams, large file downloads, and other resources to a requesting device. <i>See, e.g.,</i> evidence of use for claim element 1(f).
Claim 20		
20	20. The method as recited in claim 19 wherein the second server of the content delivery network is a content server storing the requested content.	Google includes a second server of the CDN that is a content server storing the requested content. For example, Google Media CDN retrieves data from content servers under situations such as cache bypassing, where data is retrieved from a secondary server regardless of its status on a first tier server. <i>See e.g.,</i> evidence of user for claim element 19 and the following. Origin shielding ↗ Media CDN provides a deeply tiered edge infrastructure that is designed to actively minimize cache fill wherever possible. There are three primary layers of caching infrastructure: 1. Deep edge caches, which serve the majority of traffic and off-load within a service providers network. 2. Google's peering edge, which is connected to thousands of ISPs and acts as the mid-tier cache for edge caches, and for cases where those are not present within a given ISP, the user-facing cache. 3. Long-tail caches within Google's network that other downstream caches fill from prior to the origin. These caches support significant fan-in, have substantial cache storage capacity, and act as an origin <i>shield</i>.



		All layers of caching support request collapsing (or <i>coalescing</i>) to further reduce origin load. Based on observed, real-world workloads at scale: • > 95% of cache fill uses a dedicated long-tail cache node within the region, in order to reduce cache fill costs and latency. • Cache fill between the origin and Google's own edge infrastructure is entirely over Google's global private backbone network, which reduces cache fill latency and improves reliability—both are active benefits for live streaming workloads. • Caches cross-fill from each other where advantageous to do so, further driving down cache fill rates. • Media CDN has significant storage capacity across caches, which minimizes the eviction rates even for long-tail, less popular content. Customers might see different offload rates depending on their cache configuration, user load, workloads (such as live versus on-demand), user distribution, and how much long-tail content (total corpus size) they serve to users across regions. https://cloud.google.com/media-cdn/docs/origins#origin-shielding Bypassing the cache  Cache bypass allows requests containing specific request headers to bypass the cache, even if the content was previously cached. This section provides information about bypassing the cache with HTTP headers, such as <code>Pragma</code> and <code>Authorization</code>. This feature is useful when you want to make sure that your users or customers always have the latest content fetched fresh from the origin server. You might want to do this for testing, setting up staging directories, or scripts. If a specified header matches, the cache is bypassed for all cache mode settings, even <code>FORCE_CACHE_ALL</code>. Cache bypass results in a large number of cache misses if the specified headers are common to many requests.
--	--	---

		Before you begin • Ensure that Cloud CDN is enabled; for instructions, see Using Cloud CDN. • If necessary, update to the latest version of the Google Cloud CLI: <pre>gcloud components update</pre> ★ Note: Make sure that you're using gcloud CLI version 324.0.0 or later. https://cloud.google.com/cdn/docs/caching#bypassing-cache
--	--	--