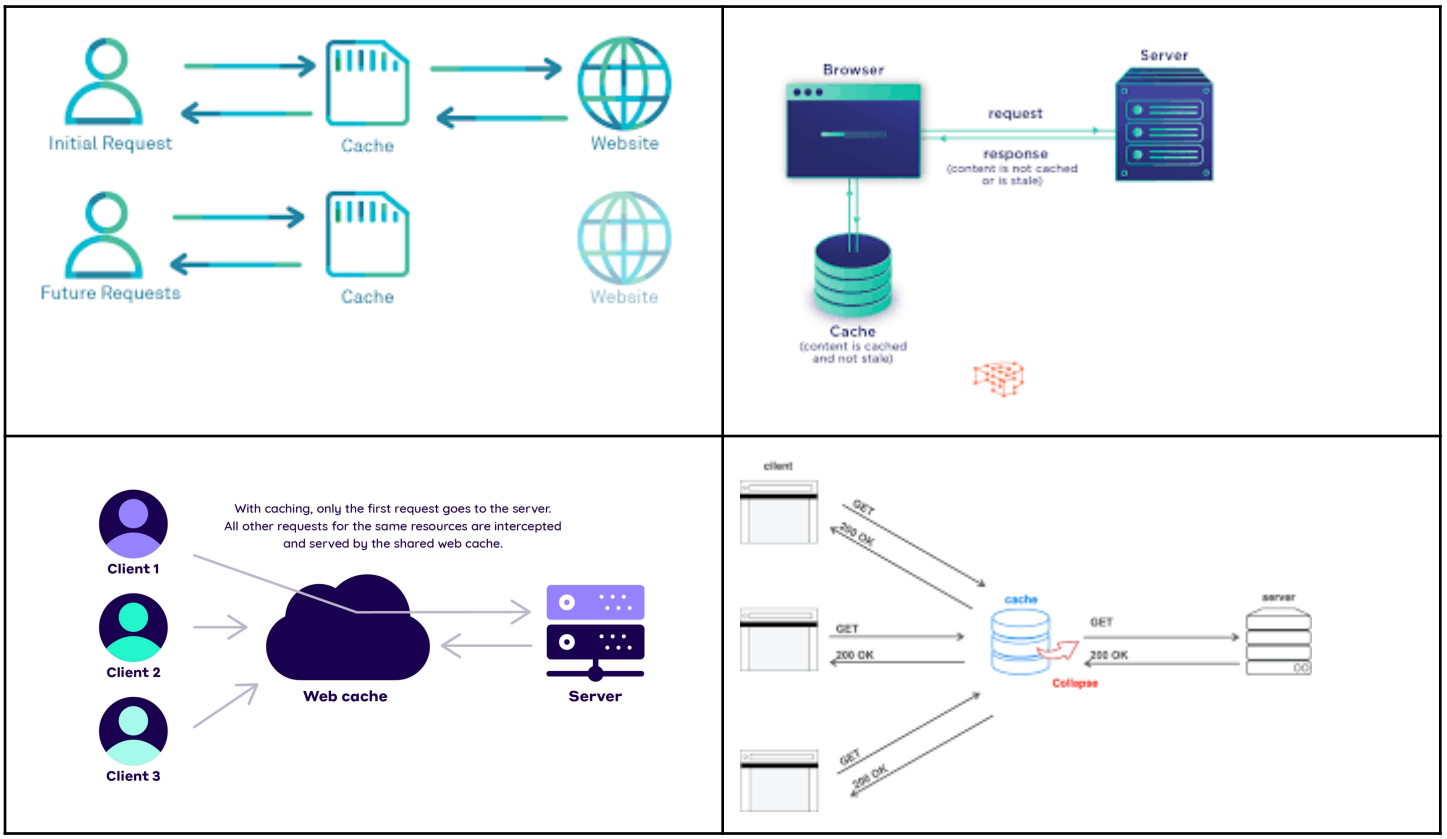
Bildungszentrum Uster
Höhere Berufsbildung Uster
HBU

HTML und CSS

02 Der Cache



Dozent Bernhard Frei

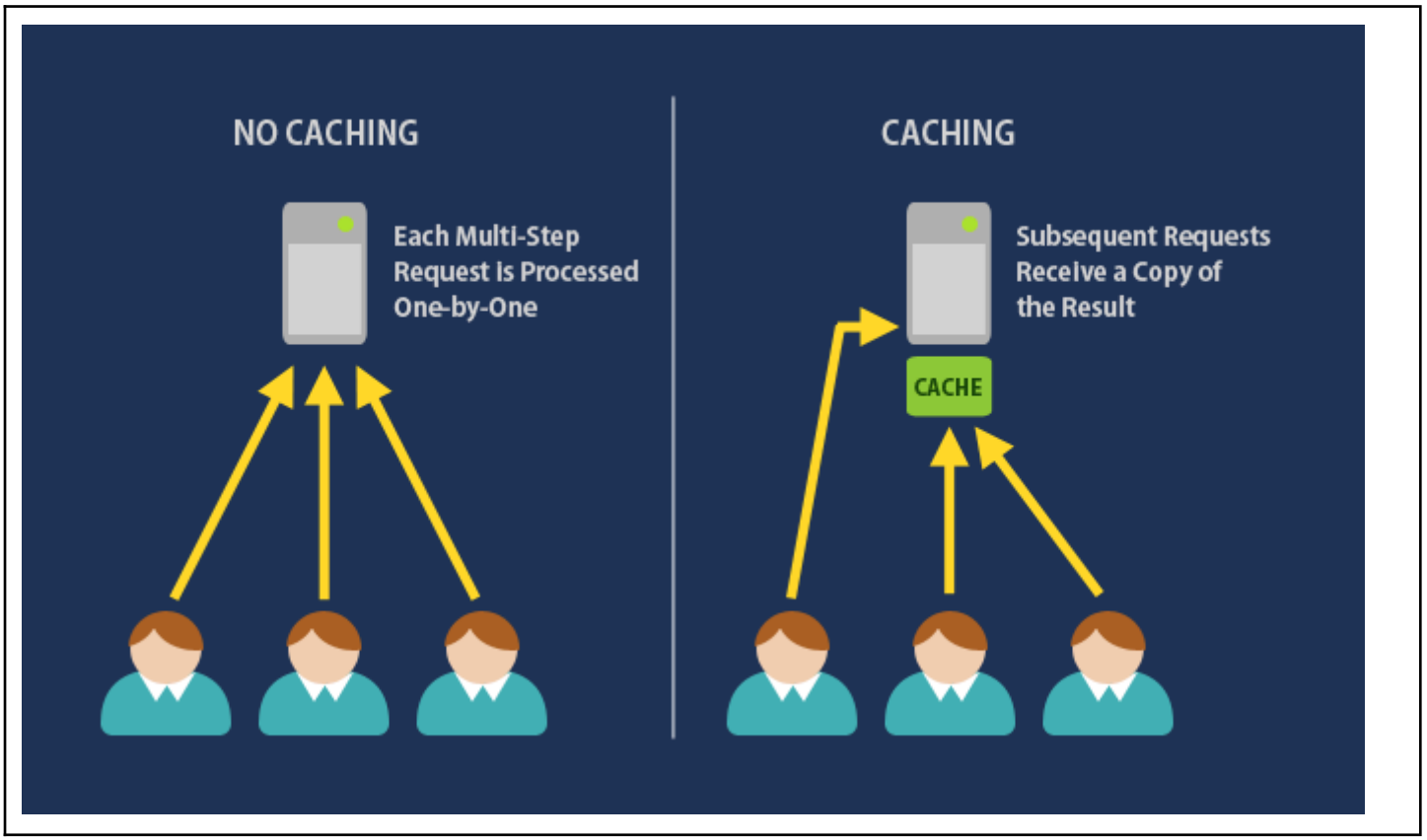
edu.bernhard.frei@gmail.com

Lernziele

- Sie verstehen was ein Cache ist und welche Arten es gibt.
- Sie kennen die Tücken des Browser-Cache und können damit umgehen.
- Sie verstehen die Wichtigkeit von Cache in verschiedenen Anwendungen

Caches

Unter einem Cache versteht man einen digitalen Zwischenspeicher, der einmal abgefragte Daten für einen späteren wiederholten Zugriff aufbewahrt. Die nachfolgenden Anfragen können dann direkt aus dem Cache beantwortet werden, ohne dass die eigentliche Anwendung hierfür kontaktiert werden muss. Ein typisches Einsatzszenario sind Webbrowser: Sie verfügen über einen eigenen Cache, der bestimmte Inhalte einer Website zwischenspeichern kann. Wird diese Seite zu einem späteren Zeitpunkt erneut besucht, kann sie schneller geladen werden, da die aufbewahrten Daten direkt aus dem Cache und nicht mehr vom Server abgerufen werden.



Definition: Cache

Ein Cache ist ein Zwischenspeicher, der Daten für wiederholte Zugriffe bereithält. Er verringert die Zeit für den wiederholten Zugriff auf dieselben Daten. Caches befinden sich als transparente Schicht zwischen dem Nutzer und der eigentlichen Quelle der Daten. Der Prozess, Daten im Cache zu speichern, wird als „Caching“ bezeichnet.

Das Bereithalten von Ressourcen, die häufig benötigt oder zusammen verwendet werden, ist ein sehr nützlicher und ganz alltäglicher Vorgang. In der digitalen Welt werden diese verwandten Prozesse allesamt unter dem Begriff „Caching“ zusammengefasst.

Was ist der Zweck eines Cache?

Der primäre Zweck eines Cache ist es, die Zugriffszeit auf wichtige Daten zu reduzieren. Als „wichtige“ Daten gelten:

- Daten, die häufig benötigt werden: In diesem Fall wäre es verschwenderisch, die Daten immer wieder aus dem hinter dem Cache liegenden, langsameren Speicher zu laden. Stattdessen werden diese mit kürzerer Zugriffszeit aus dem Cache ausgeliefert.

- Daten, deren Erzeugung ein aufwendiger Prozess zugrunde liegt: Manche Daten sind das Ergebnis einer rechenintensiven Verarbeitung, oder die Daten müssen aus verschiedenen Teilen zusammengefügt werden. In diesen Fällen bietet es sich an, die fertigen Daten für weitere Abfragen in einem Cache zu speichern.

- Daten, die zusammen benötigt werden: In diesem Fall wäre es ineffizient, die Daten erst dann zu laden, wenn Sie abgerufen werden. Stattdessen ist es sinnvoll, die Daten gemeinsam im Cache vorzuhalten.

Wie funktioniert ein Cache?

Nun gehen wir genauer auf die Funktionsweise eines Cache ein. Dabei klären wir u. a. die Fragen, wie ein solcher Zwischenspeicher grundsätzlich arbeitet und wo genau er sich befindet.

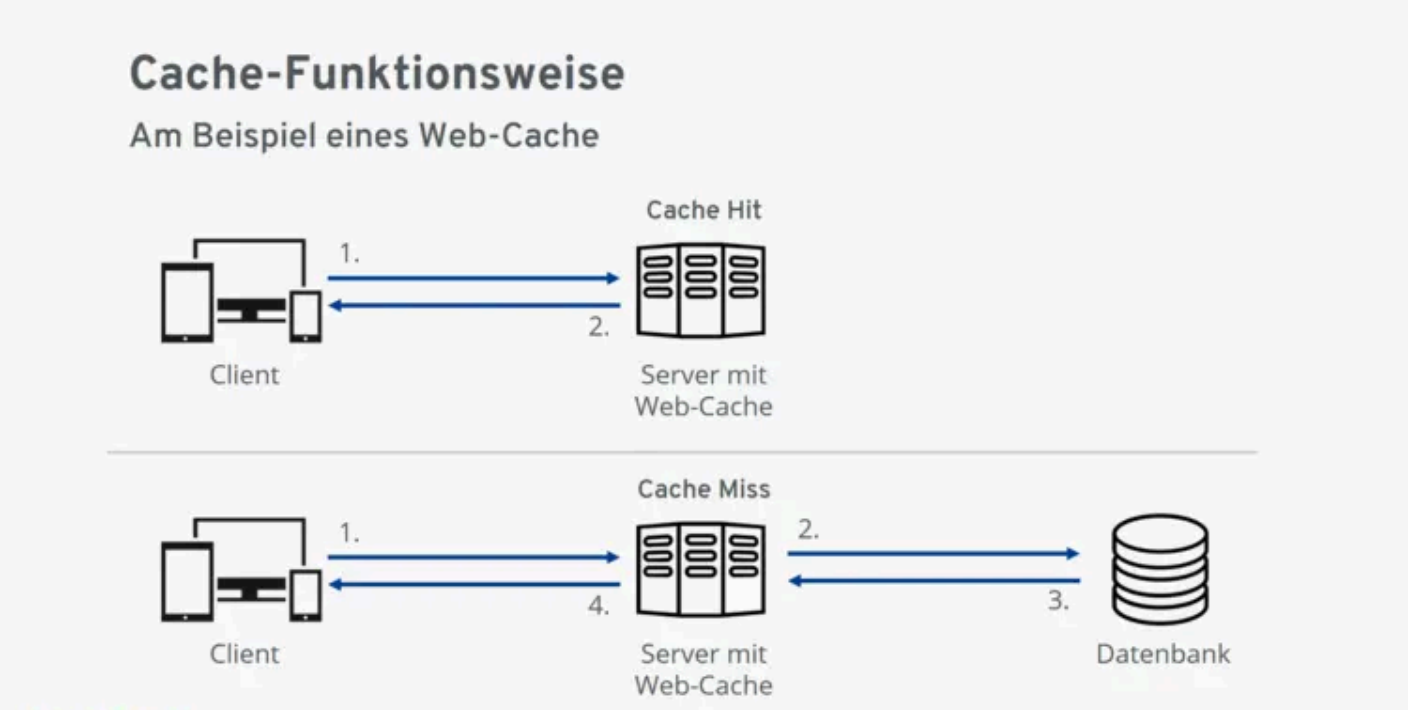
Grundlegendes Cache-Schema

- Eine Anfrage nach einer Ressource wird an das System bzw. die Software, die über einen Cache verfügt, gestellt.

- Ist die Ressource bereits im Cache enthalten, wird sie aus dem Cache ausgeliefert. Dieser Fall wird als „Cache Hit“, also „Cache-Treffer“, bezeichnet.

- Ist die Ressource nicht im Cache enthalten, wird sie zunächst aus dem dahinterliegenden System in den Zwischenspeicher geladen und anschließend ausgeliefert. Dieser Fall wird als „Cache Miss“, in etwa „Cache-Fehlschuss“, bezeichnet.

- Wird dieselbe Ressource in der Zukunft erneut abgefragt, kann sie ebenfalls als Cache Hit aus dem Cache ausgeliefert werden.



Abgebildet ist die generelle Funktionsweise eines Cache, dargestellt am spezifischen Beispiel eines Web-Cache.

Die Abbildung verdeutlicht das grundlegende Schema: Ein Client stellt eine Anfrage nach einer Ressource an den Server (1). Im Fall des „Cache Hit“ ist die Ressource bereits im Cache enthalten und wird sogleich an den Client ausgeliefert (2). Im Fall des „Cache Miss“ ist die abgefragte Ressource nicht im Cache enthalten und wird daher aus dem hinter dem Server liegenden System (hier eine Datenbank) abgefragt (2). Sobald die Ressource vorhanden ist (3), wird sie an den Client ausgeliefert (4) und für die weitere Nutzung im Cache gespeichert.

Wo befindet sich ein Cache?

Eine Grundeigenschaft eines Cache ist, dass er „im Verborgenen“ arbeitet. Dieser Umstand spiegelt sich schon im Ursprung des Wortes wider: Das Wort „Cache“ stammt aus dem Französischen und bedeutet „Versteck“.
Der Cache liegt für Nutzer unsichtbar vor dem eigentlichen Datenspeicher. Das bedeutet, dass Sie als Nutzer nichts über die internen Eigenschaften des Cache wissen müssen. Sie stellen Anfragen an den Datenspeicher, ohne zu bemerken, dass diese eigentlich vom Cache beantwortet werden.

Wie viele Caches gibt es typischerweise?

Für gewöhnlich sind innerhalb eines digitalen Systems immer eine Reihe an Caches aktiv.
Stellen Sie sich den Zugriff auf eine Webseite vor: Ihr Browser kommuniziert mit einem Server und ruft eine Reihe von Ressourcen ab. Bis Ihnen die Inhalte der Seite im Browser angezeigt werden, laufen Teile davon vermutlich durch folgende Caches: Prozessor-Cache, System-Cache und Browser-Cache auf Ihrem Gerät sowie CDN (Content Delivery Network) und Web-Cache auf Serverseite.

Vor- und Nachteile eines Cache

Ob eine Anwendung mit einem Cache ausgestattet wird, liegt prinzipiell im Ermessen des Entwicklers. Wir haben die Vor- und Nachteile der Zwischenspeicher-Lösung zusammengefasst.

Vorteil: Enormer Zuwachs an Geschwindigkeit

Die Verwendung eines Cache bietet als potenziellen Vorteil einen enormen Zuwachs an Geschwindigkeit. Eine Beschleunigung um einen Faktor von einhundert ist nicht ungewöhnlich. Die Beschleunigung ergibt sich jedoch nur beim wiederholten Zugriff auf dieselben Daten. Wie groß der Zugewinn tatsächlich ausfällt, wird also je nach Anwendungsfall stark variieren.

Vorteil: Lastreduzierung für das hinter dem Cache liegende System

Da ein Cache Daten sehr schnell ausliefert, wird die Last auf das hinter ihm liegende System deutlich reduziert.
Stellen Sie sich als Beispiel vor, dass eine dynamische HTML-Seite aus einem PHP-Template erzeugt wird: Zur Erzeugung der Seite wird auf eine Datenbank zugegriffen. Dieser Zugriff ist vergleichsweise aufwendig. Ferner ist es nicht trivial, Datenbankserver zu skalieren, weshalb der Datenbankzugriff als „bottleneck“ (zu Deutsch: „Flaschenhals“) den Gesamtdurchsatz des Systems begrenzen kann. In diesem Fall ist es vorteilhaft, die generierte HTML-Seite in einem Web-Cache zwischenzuspeichern, um die Kapazität des Datenbankservers für andere Aufgaben zu nutzen.

Nachteil: Cache-Invalidierung ist schwierig

Cache-Invalidierung ist ein Mechanismus, der verwendet wird, um die Konsistenz zwischen den zwischengespeicherten Daten (Cache) und den Originaldaten sicherzustellen. Caches werden verwendet, um häufig genutzte Daten temporär zu speichern und den Zugriff auf diese Daten zu beschleunigen, anstatt sie jedes Mal aus der ursprünglichen Quelle abzurufen. Wenn sich die Originaldaten jedoch ändern, müssen die zwischengespeicherten Daten aktualisiert oder entfernt werden, um sicherzustellen, dass keine veralteten Informationen verwendet werden. Dieser Prozess der Aktualisierung oder Entfernung wird als Cache-Invalidierung bezeichnet.
Der Begriff Cache-Invalidierung bezeichnet also die Entscheidung darüber, wann zwischengespeicherte Daten nicht mehr aktuell sind und erneuert werden müssen. Wenn eine Ressource fehlt, die aber noch benötigt wird, kommt es zu einem Cache Miss.
Da der Cache Miss kostspielig ist (muss nachgeladen werden), zielt die optimale Caching-Strategie darauf ab, diesen möglichst zu vermeiden. Andererseits kann das dazu führen, dass der Cache Daten ausliefert, die nicht mehr aktuell sind. Dieses Problem wird noch verschärft, wenn mehrere, hierarchisch angrenzende

Caches aktiv sind. Es kann dann schwierig sein zu bestimmen, wann welche Daten im Cache als nicht mehr aktuell markiert werden sollen.
Liefert ein Cache Daten aus, die nicht mehr aktuell sind, führt dies oft auf der Seite des Anwenders zu seltsamen Problemen: Die besuchte Website hat Darstellungsfehler oder es erscheinen beim Datenabruf Fragmente aus der Vergangenheit. Mitunter kann es schwierig sein, die genaue Herkunft der Probleme zu ermitteln, weshalb in diesem Fall das Leeren des Cache die beste Lösung darstellt.

Zitat
„There are only two hard things in Computer Science: cache invalidation and naming things."
„In der Informatik gibt es nur zwei schwierige Probleme: Cache-Invalidierung und Namen vergeben."
Phil Karlton, https://www.martinfowler.com/bliki/TwoHardThings.html

Cache-Invalidierung ist ein wesentlicher Bestandteil von Cache-Mechanismen, um Datenkonsistenz zu gewährleisten. Die Wahl der richtigen Invalidierungsstrategie hängt stark von den Anforderungen der Anwendung, der Architektur des Systems und den Leistungszielen ab. Ein gut durchdachtes Invalidierungssystem kann die Performance erheblich steigern und gleichzeitig sicherstellen, dass Benutzer immer auf aktuelle Daten zugreifen.

Welche Arten von Caches gibt es?

Caches bestehen aus Hardware- oder Software-Komponenten. Ein Hardware-Cache ist ein schneller Pufferspeicher, der die Zugriffszeiten auf den dahinterliegenden Datenspeicher verringert. Prinzipiell sind Hardware-Caches immer sehr klein im Vergleich zur Gesamtgröße des beschleunigten Datenspeichers. Dagegen können in Software implementierte Caches die Größe der hinter ihnen liegenden Ressource sogar übertreffen. Dies ist insbesondere dann der Fall, wenn sich mehrere Versionen einer Ressource im Cache befinden.

Hier eine Übersicht gängiger mit Caches versehener Ressourcen. Dargestellt sind die Größe des Cache, die Zugriffszeit auf den Cache sowie eine Angabe, wie viel langsamer der Zugriff auf die Ressource ohne Zwischenspeicher wäre.

Ressource	Cache	Größe des Cache	Zugriffszeit mit Cache	× langsamer ohne Cache
Hauptspeicher	Level 1-Cache (Hardware)	Dutzende Kilobyte (KB)	Weniger als eine Nanosekunde (ns)	200 ×
Festplatte	Festplatten-Cache (Hardware)	Dutzende Megabyte (MB)	Hunderte Nanosekunden (ns)	100 ×
Browser	Browser-Cache (Software)	Mehrere Gigabyte (GB)	Dutzende Millisekunden (ms)	10–100 ×
Websites	CDNs, Proxys, Google Page Cache, Wayback Machine (Software)	Tausende Terabyte (Petabyte, PB)	Wenige Sekunden (s)	2–5 ×

Hardware-Caches

Prozessor-Cache

Ein moderner Prozessor arbeitet unglaublich schnell. Die Abläufe auf dem Chip benötigen nur Bruchteile von Nanosekunden — das ist eine Milliardstel Sekunde! Im Gegensatz dazu ist der Zugriff auf den Hauptspeicher mit Hunderten von Nanosekunden vergleichsweise langsam. Aus diesem Grund verfügen moderne Prozessoren über eine Hierarchie von Prozessor-Caches.
Ein Cache Hit auf dem schnellsten Prozessor-Cache, bekannt als „Level 1-Cache“ oder „L1-Cache“, ist rund 200 Mal schneller als ein Zugriff auf den Hauptspeicher.

Festplatten-Cache

Eine Festplatte rotiert mit mehreren Tausend Umdrehungen pro Minute. Der Schreib-Lese-Kopf rast über die Scheiben und liest dabei sequenziell Daten aus. Da es sich um einen physischen Prozess handelt, ist der Zugriff auf eine Festplatte vergleichsweise langsam.
Aus diesem Grund verfügt jede Festplatte über einen eigenen kleinen Cache. So müssen zumindest die am häufigsten genutzten Daten – etwa Teile des Betriebssystems – nicht immer wieder aufwendig von der Festplatte gelesen werden.
Durch den Festplatten-Cache können essenzielle Daten ca. 100 Mal schneller geladen werden. Sie sind dadurch für den Nutzer gefühlt „sofort“ vorhanden.

Software-Caches

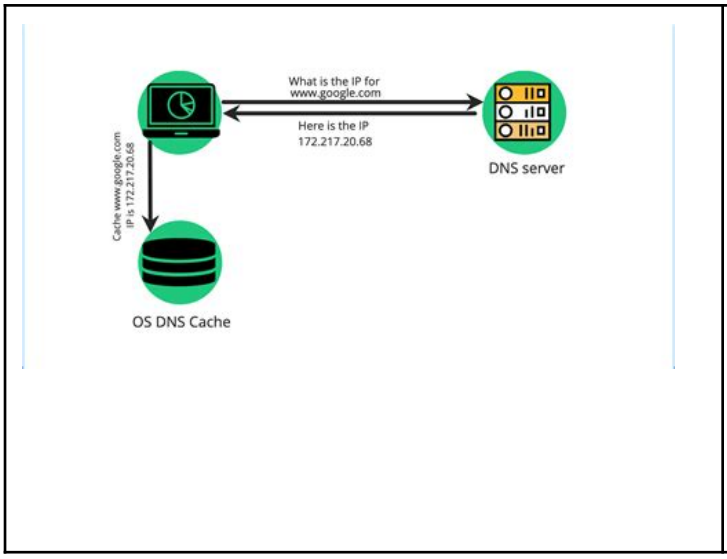
Browser-Cache

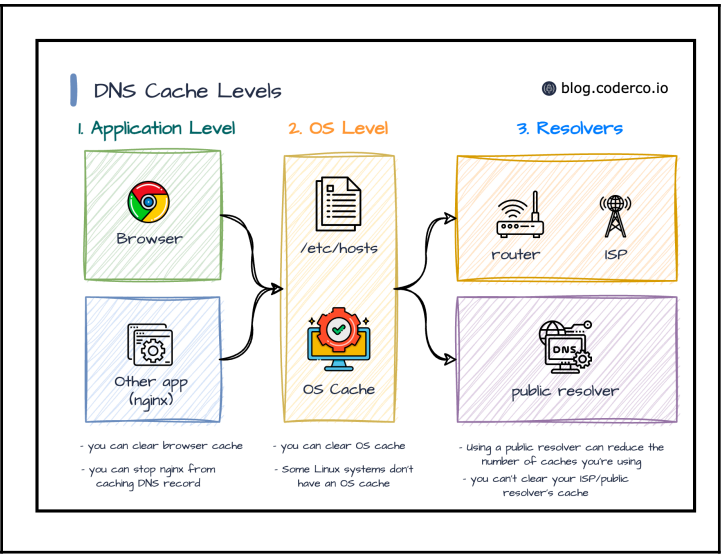
Beim Besuch einer Website werden viele Daten der Seite auf dem Gerät des Besuchers zwischengespeichert. Neben den eigentlichen Inhalten gehören dazu verschiedene Ressourcen wie Bilder, Stylesheets und JavaScript-Dateien. Für gewöhnlich werden viele dieser Ressourcen auf mehreren Seiten benötigt. Um das Laden der Seiten zu beschleunigen, ist es vorteilhaft, diese immer wieder benötigten Ressourcen im Browser-Cache des lokalen Geräts zu speichern.
So praktisch der Browser-Cache für das Surfen im Netz ist: Er kann auch Probleme verursachen – etwa dann, wenn die Entwickler Änderungen an einer Ressource der Website vorgenommen haben, im Browser-Cache aber noch die alte Version der Ressource vorhanden ist. In diesem Fall kann es zu Darstellungsfehlern kommen. Abhilfe schafft dann das Leeren des Browser-Cache.

Google Page-Cache

Googles „Im Cache“-Funktion hält die Seiten vieler Websites vorrätig. So kann auf die Seiten auch noch zugegriffen werden, wenn die ursprüngliche Website offline ist. Der Zustand der Seiten entspricht dem Datum der letzten Indizierung durch den Googlebot.

DNS-Cache



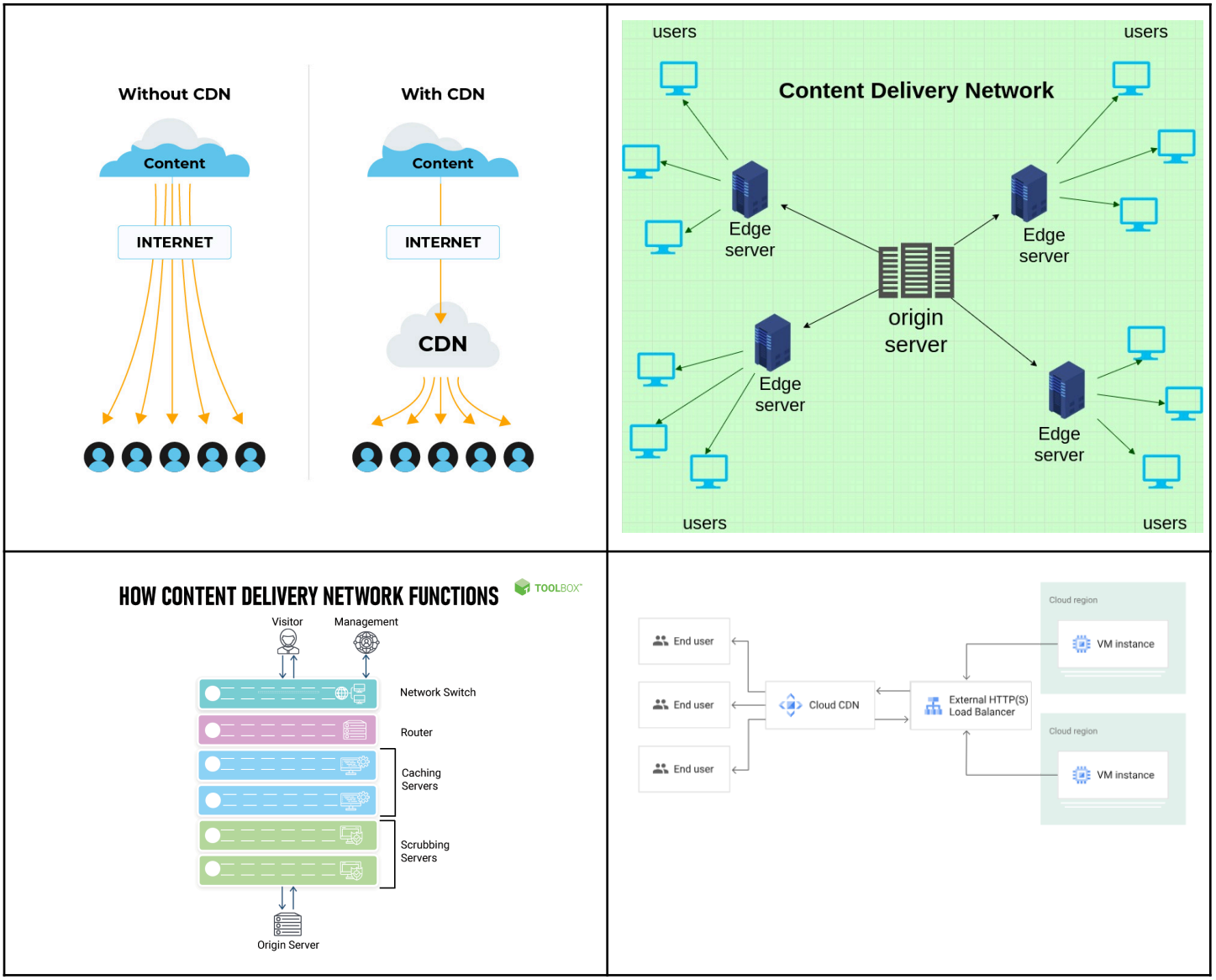


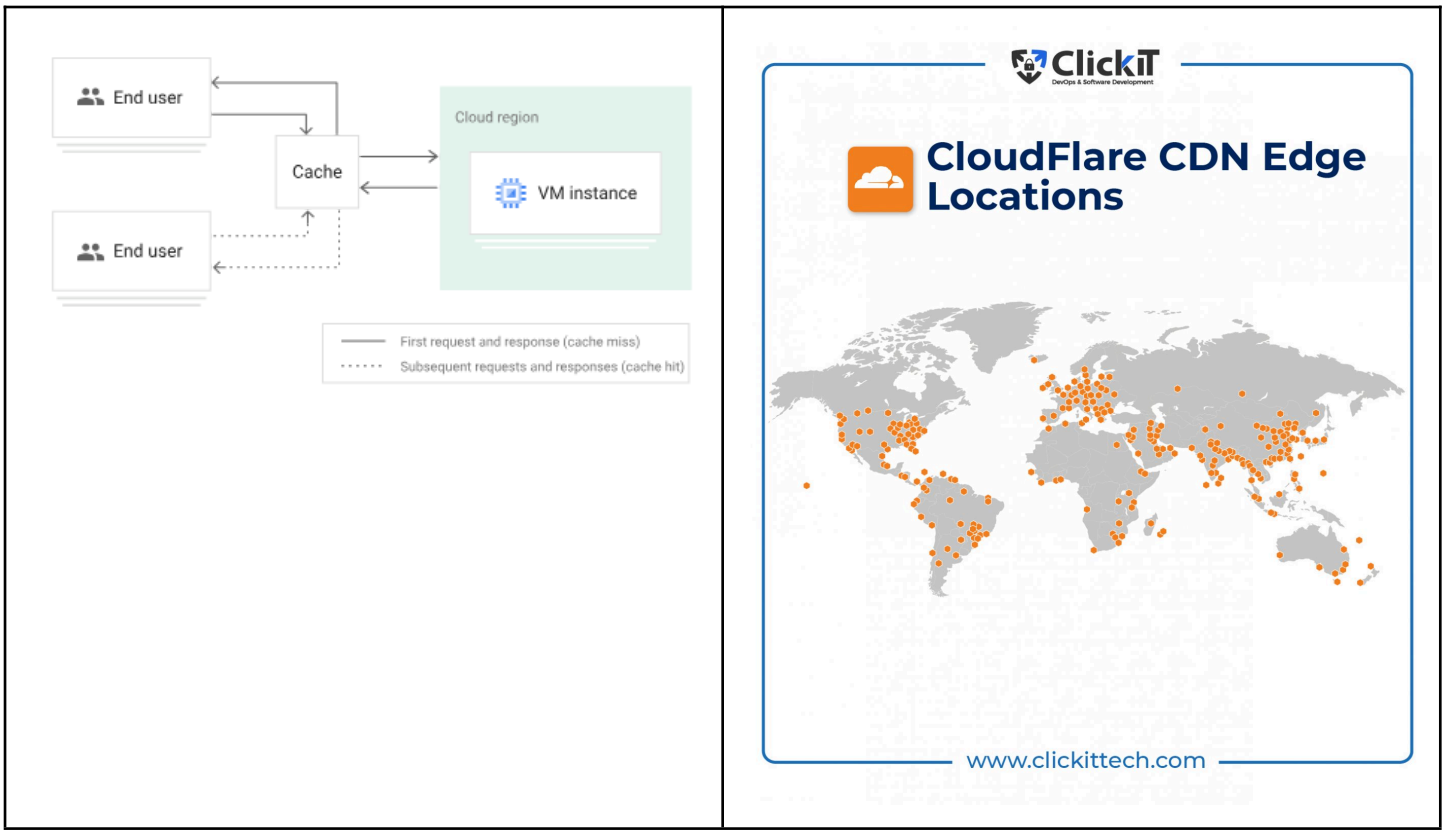
Das Domain Name System, kurz DNS, ist ein global verteiltes System zur Übersetzung von Internetdomains in IP-Adressen (bzw. umgekehrt). Das DNS liefert für einen Domänennamen eine IP-Adresse zurück. Beispielsweise wird für die Domain ionos.de die IP-Adresse 217.160.86.40 zurückgeliefert.
Bereits beantwortete Anfragen an das DNS werden lokal auf dem eigenen Gerät im DNS-Cache

zwischengespeichert. So ist jede Auflösung immer gleich schnell.
Doch auch die Nutzung des DNS-Cache kann zu Problemen führen – beispielsweise, wenn sich durch einen Serverumzug die zu einer Domain gehörige IP-Adresse geändert hat, die alte Adresse jedoch noch im lokalen DNS-Cache hinterlegt ist. In diesem Fall schlägt der Verbindungsaufbau mit dem Server fehl. Abhilfe schafft dann das Löschen des DNS-Cache (DNS-Flush im Betriebssystem).

Content Delivery Network (CDN)

Global verteilte Content Delivery Networks (dt. „Inhalte-Auslieferungs-Netzwerke“) halten einen Großteil der Daten beliebter Websites auf sogenannten Edge Nodes („randnahen Knoten“) vor. Diese Edge Nodes spiegeln die Daten am „Rand“ des Internets. Die Knoten sind lokal nahe am Benutzer angesiedelt und technisch darauf ausgelegt, Daten möglichst schnell auszuliefern. Ein CDN fungiert als Cache für die Daten der enthaltenen Websites. Dadurch werden die Zugriffszeiten insbesondere auf Streaming-Angebote und Websites minimiert.





Ein Content Delivery Network (CDN) ist also ein global verteiltes Netzwerk von Servern, das entwickelt wurde, um die Leistung und Verfügbarkeit von Websites, Anwendungen und anderen internetbasierten Diensten zu verbessern. CDNs optimieren die Übertragung von Webinhalten, indem sie Daten von den zentralen Ursprungsservern zu geografisch verteilten Edge Nodes replizieren, welche die Daten zwischenspeichern (cachen). Dadurch wird die physische Distanz zwischen den Benutzern und den Servern, die die Inhalte bereitstellen, minimiert, was zu kürzeren Ladezeiten und einer besseren Nutzererfahrung führt.

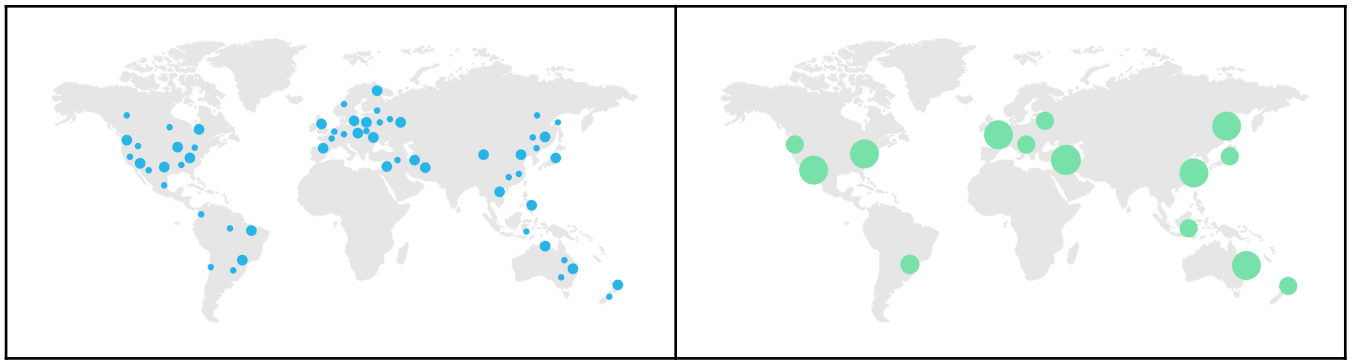
Komponenten von einem CDN

Technisch gesehen bestehen CDNs aus verschiedenen Komponenten:

1. Edge Nodes und PoPs (Points of Presence): CDNs verwenden eine große Anzahl von Edge Nodes, die in Points of Presence (PoPs) zusammengefasst sind. Diese PoPs sind strategisch in Regionen mit hohem Datenverkehr platziert, um die Netzwerk-Latenz zu verringern. Die Edge Nodes speichern häufig angeforderte Inhalte, wie HTML-Seiten, Bilder, Videos und andere Dateien, lokal. Wenn ein Benutzer eine Anfrage stellt, wird der Inhalt von dem geografisch nächstgelegenen Edge Node geliefert, wodurch die Wartezeit reduziert und die Übertragungsgeschwindigkeit erhöht wird.
2. Caching und Cache-Invalidierung: CDNs nutzen Caching-Strategien, um Inhalte auf Edge Nodes zu speichern. Caches sind temporäre Speicher, die die am häufigsten abgerufenen Daten lokal halten, sodass diese nicht jedes Mal vom Ursprungsserver geladen werden müssen. CDNs bieten verschiedene Cache-Strategien wie „Time-to-Live“ (TTL), um zu bestimmen, wie lange Inhalte auf den Edge Nodes gespeichert bleiben, bevor sie als veraltet gelten und aktualisiert werden müssen.

Cache-Invalidierung ist ein Mechanismus, der es ermöglicht, zwischengespeicherte Inhalte gezielt zu löschen oder zu aktualisieren, wenn sich die Originaldaten ändern.

3. Lastverteilung und Failover-Mechanismen: CDNs bieten auch Lastverteilung (Load Balancing), um Anfragen effizient über mehrere Server zu verteilen und so eine Überlastung einzelner Server zu vermeiden. Failover-Mechanismen stellen sicher, dass im Falle eines Serverausfalls die Anfragen automatisch zu einem anderen funktionsfähigen Edge Node weitergeleitet werden. Dadurch wird die Ausfallsicherheit und Verfügbarkeit der Dienste erhöht.
4. Protokolloptimierung und Sicherheitsfunktionen: CDNs nutzen verschiedene Protokolloptimierungen, wie z. B. HTTP/2, QUIC und TCP-Optimierungen, um die Datenübertragungsrate zu verbessern. Außerdem bieten sie eine Reihe von Sicherheitsfunktionen, wie DDoS-Schutz (Distributed Denial of Service), Web Application Firewalls (WAF) und TLS-Verschlüsselung (Transport Layer Security), um den Datenverkehr zu sichern und Angriffe abzuwehren.
5. Origin Shielding: Eine zusätzliche Schutzschicht, genannt Origin Shielding, hilft, die Belastung des Ursprungsservers weiter zu reduzieren. Hierbei wird ein spezieller PoP als „Shield“ konfiguriert, der als letzte Verteidigungslinie agiert, bevor Anfragen den Ursprungsserver erreichen. Dadurch wird die Anzahl der direkten Zugriffe auf den Ursprungsserver minimiert und seine Leistung optimiert.



6. Anwendungsfälle und dynamische Inhalte: Während CDNs traditionell für die Bereitstellung statischer Inhalte wie Bilder und Videos verwendet wurden, unterstützen sie mittlerweile auch die Beschleunigung dynamischer Inhalte. Dies geschieht durch Techniken wie Edge-Scripting und Edge-Computing, bei denen bestimmte Teile der Anwendung direkt auf den Edge Nodes ausgeführt werden, um die Latenz weiter zu reduzieren und die Rechenlast auf den Ursprungsservern zu verringern.

Durch diese technischen Mechanismen ermöglichen CDNs eine schnellere, sicherere und zuverlässigere Bereitstellung von Webinhalten weltweit, was zu einer verbesserten Benutzererfahrung führt und gleichzeitig die Netzwerkkosten für den Inhaltseigentümer reduziert.

Die vier Säulen des CDN-Designs

Leistung

Eine der Hauptaufgaben eines CDN besteht darin, die Latenz zu minimieren. Aus architektonischer Sicht bedeutet dies, dass optimale Konnektivität geschaffen werden muss, wobei PoPs an wichtigen Netzwerkknotenpunkten liegen, an denen Daten übertragen werden.

Die physischen Einrichtungen sind ein weiterer wichtiger Aspekt. In der Regel möchten Sie, dass sich Ihr PoP immer in einem erstklassigen Rechenzentrum befindet, in dem Backbone-Anbieter miteinander peeren und Ihr CDN-Anbieter Peering-Vereinbarungen mit anderen CDNs und großen Carriern geschlossen hat. Solche Vereinbarungen ermöglichen es CDNs, die Roundtrip-Zeiten erheblich zu verkürzen und die Bandbreitennutzung zu verbessern.

Skalierbarkeit

CDNs sind für Hochgeschwindigkeits- und Hochvolumen-Routing konzipiert und müssen jede Menge Datenverkehr bewältigen können. Die CDN-Architektur sollte diese Erwartungen erfüllen, indem sie auf allen Ebenen ausreichend Netzwerk- und Verarbeitungsressourcen bereitstellt – bis hin zu den Rechen- und Caching-Ressourcen, die auf jedem der Caching-Server verfügbar sind.

Wie zu erwarten, haben CDNs, die DDoS-Schutzdienste anbieten, viel höhere Skalierbarkeitsanforderungen. Um diesen Anforderungen gerecht zu werden, setzen sie dedizierte Server ein, die für die Abwehr von DDoS-Angriffen (Scrubber) entwickelt wurden. Diese können einzeln netzwerkgroße Datenmengen verarbeiten und dabei Dutzende Gigabyte pro Sekunde verarbeiten.

Zuverlässigkeit

Die Größe der CDN-Infrastruktur macht ein störungsfreies System statistisch unwahrscheinlich. Diese gleiche Größe kann jedoch dazu beitragen, eine Rekord-Ausfallsicherheit und hohe Verfügbarkeit sicherzustellen, sodass CDN-Anbieter sich zu Service Level Agreements (SLAs) von 99,9 % und 99,999 % verpflichten können.

In der Regel verfolgen kommerzielle CDNs einen Ansatz, bei dem es keine einzelnen Ausfallpunkte gibt, indem sie Wartungszyklen sorgfältig planen und zusätzliche Hardware- und Softwareredundanz integrieren. Viele verwalten auch interne Failover- und Notfallwiederherstellungssysteme, die den Datenverkehr automatisch um ausgefallene Server herumleiten. Für zusätzliche Redundanz arbeiten CDN-Anbieter auch mit mehreren Carriern zusammen und verlassen sich auf dedizierte Out-of-Band-Verwaltungskanäle, die es ihnen ermöglichen, im Katastrophenfall mit Servern zu interagieren.

Reaktionsfähigkeit

Bei einem globalen Netzwerk sind CDNs ständig bestrebt, die Reaktionsfähigkeit zu verbessern – gemessen an der Zeit, die netzwerkweite Konfigurationsänderungen benötigen, um wirksam zu werden.
Bedenken Sie, dass selbst kleine Konfigurationsänderungen, wie die Anweisung, ein bestimmtes Bild aus dem Cache zu löschen oder die Aufnahme einer Adresse in eine schwarze Liste von IP-Adressen, über alle PoPs hinweg kommuniziert werden müssen. Je größer und geographisch verteilter das Netzwerk ist, desto länger dauert dies.
Um eine gute Servicequalität für Kunden sicherzustellen, sollte das CDN so konzipiert sein, dass eine schnelle Konfigurationsverbreitung möglich ist.

Scrubbing Servers

Die Hauptfunktionen eines Scrubbing Centers umfassen die folgenden Aspekte:

- Echtzeit-Überwachung: Das Scrubbing Center überwacht kontinuierlich den ein- und ausgehenden Datenverkehr in Echtzeit, um verdächtige Aktivitäten zu erkennen.
- Erkennung von Bedrohungen: Es identifiziert schädlichen oder verdächtigen Datenverkehr, der auf Cyberbedrohungen wie Distributed Denial of Service (DDoS)-Angriffe, Botnet-Aktivitäten, Malware-Verbreitung und andere Angriffe hinweisen kann.
- Datenverkehrsfiltration: Das Zentrum filtert den schädlichen Datenverkehr aus und leitet ihn von den schützenswerten Netzwerkressourcen ab, um die Netzwerkintegrität und Dienstverfügbarkeit sicherzustellen.
- Anomalieerkennung: Scrubbing Centers erkennen Anomalien und ungewöhnliche Muster im Datenverkehr, die auf potenzielle Bedrohungen hinweisen könnten.
- DDoS-Abwehr: Sie sind besonders darauf spezialisiert, Distributed Denial of Service (DDoS)-Angriffe zu erkennen und abzuwehren, indem sie den schädlichen Verkehr von legitimen Anfragen trennen.
- Protokollierung und Berichterstattung: Scrubbing Centers führen Protokolle über alle erkannten Vorfälle und stellen umfassende Berichte und Analysen zur Verfügung, um Kunden über die Sicherheitslage zu informieren.
- Lastverteilung: In einigen Fällen bieten Scrubbing Centers Lastverteilungsfunktionen, um den Datenverkehr auf verschiedene Server oder Rechenzentren zu verteilen und so eine höhere Ausfallsicherheit zu gewährleisten.
- Zusätzliche Sicherheitsdienste: Neben der DDoS-Abwehr bieten einige Scrubbing Centers auch zusätzliche Sicherheitsdienste wie Intrusion Detection und Intrusion Prevention Systems (IDS/IPS) sowie Schutz vor anderen Arten von Angriffen.

Diese Funktionen sind entscheidend, um die Netzwerksicherheit zu gewährleisten, die Verfügbarkeit von Diensten aufrechtzuerhalten und schädliche Aktivitäten zu blockieren.

Web-Cache

Ein Web-Cache hält Webdokumente wie HTML-Seiten, Bilder, Stylesheets, oder JavaScript-Dateien für die wiederholte Nutzung vor. Moderne Web-Caches wie Varnish und Redis legen häufig benutzte Daten im Arbeitsspeicher ab und erzielen damit besonders geringe Antwortzeiten.
Werden die Daten erneut abgefragt, erfordert die Antwort nur einen sehr schnellen Speicherzugriff. So werden die Antwortzeiten drastisch verringert und die Last für die hinter dem Cache liegenden Systeme, wie Webserver und Datenbank, reduziert. Weitere bekannte Web-Caches sind OPcache und der Alternative PHP Cache (APC).

Arten von Web-Cache

Es gibt mehr als eine Art von Web-Caches. Sie sind an mehreren Stellen auf dem Weg zwischen Ihrem Browser und dem Quell-/Ursprungsserver vorhanden:

Browser-Caches

Browser-Caches verwenden einen Teil des lokalen Festplattenspeichers Ihres Geräts, um statische Kopien von Inhalten wie zuvor besuchten Webseiten zu speichern, um Ihr Online-Erlebnis zu beschleunigen. Wenn Sie eine Seite erneut besuchen, wird sie möglicherweise, für Sie unsichtbar, direkt aus der zwischengespeicherten Kopie Ihres eigenen PCs geladen. Es kann sein, dass die Anforderung Ihren Computer nie verlässt.

Proxy-Web-Caches

Web-Proxys werden häufig von Organisationen eingesetzt, um Anfragen aller Benutzer der Organisation gemeinsam zwischenzuspeichern. Sie befinden sich am Netzwerkrand der Organisation und können sehr effektiv sein, wenn viele Benutzer auf gemeinsame Ressourcen wie Nachrichten-Websites zugreifen.

Internetdienstanbieter (ISPs)

ISPs betreiben in der Regel auch Proxy-Caches über Interception-Proxys in ihrem zugrunde liegenden Netzwerk und nutzen die Größe von möglicherweise Hunderttausenden von Benutzern, um häufig angeforderte Inhalte für alle ihre Abonnenten zwischenzuspeichern.

Content Delivery Networks (CDN)

CDNs wie Akamai oder Speedera sind auf der ganzen Welt verteilt und werden im Allgemeinen von kommerziellen Organisationen gemietet, die Inhalte produzieren. Wenn sie im DNS konfiguriert sind, erhalten

Kunden, die eine DNS-Suche nach dem Ursprungsserver durchführen, eine IP für einen lokalen CDN-Server, der von der CDN-Firma betrieben wird und berechtigt ist, sich als Ursprungsserver auszugeben. Dabei werden Inhalte dieses Anbieters für alle Benutzer in einer bestimmten Region zwischengespeichert und das Volumen der Anfragen an die Ursprungsserver drastisch reduziert.

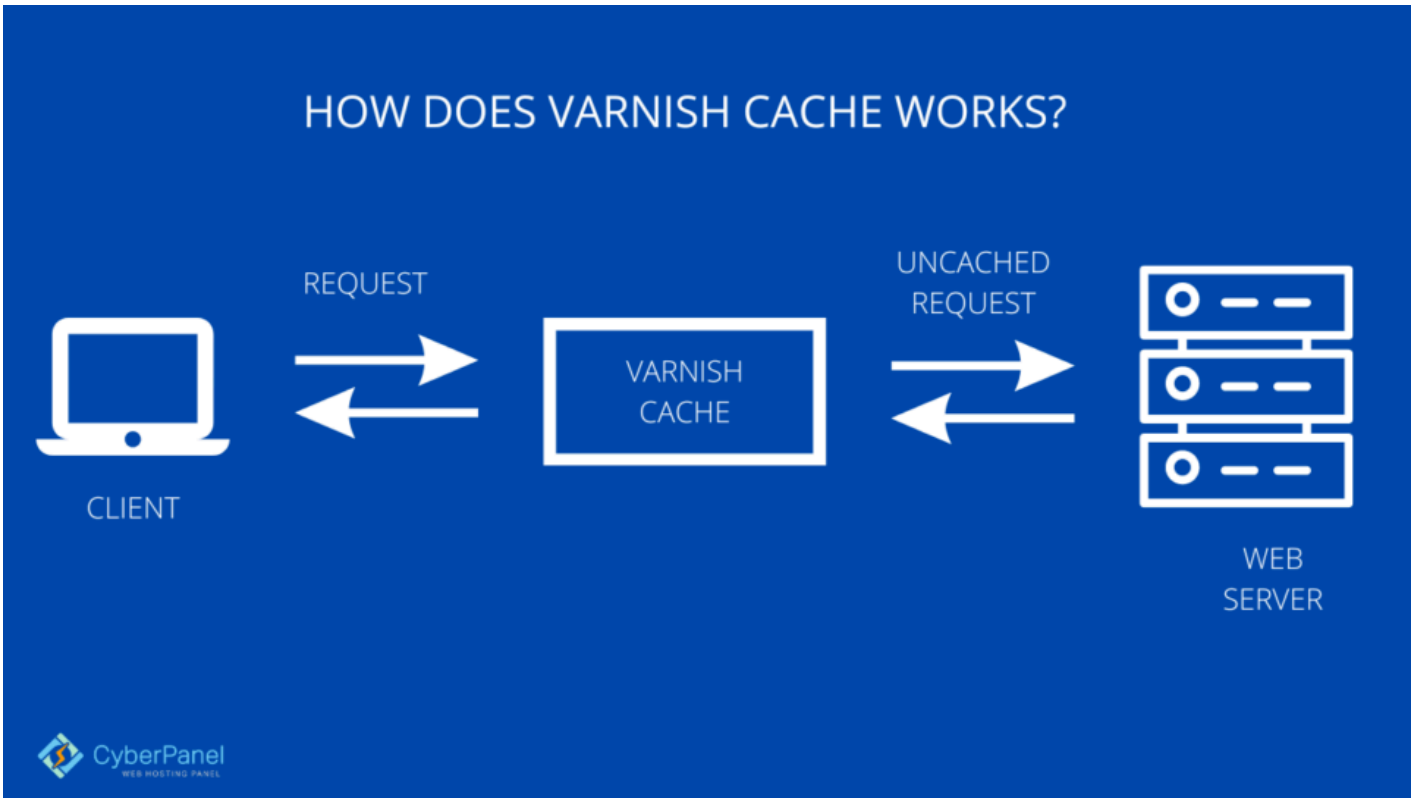
Gateway-Web-Caches

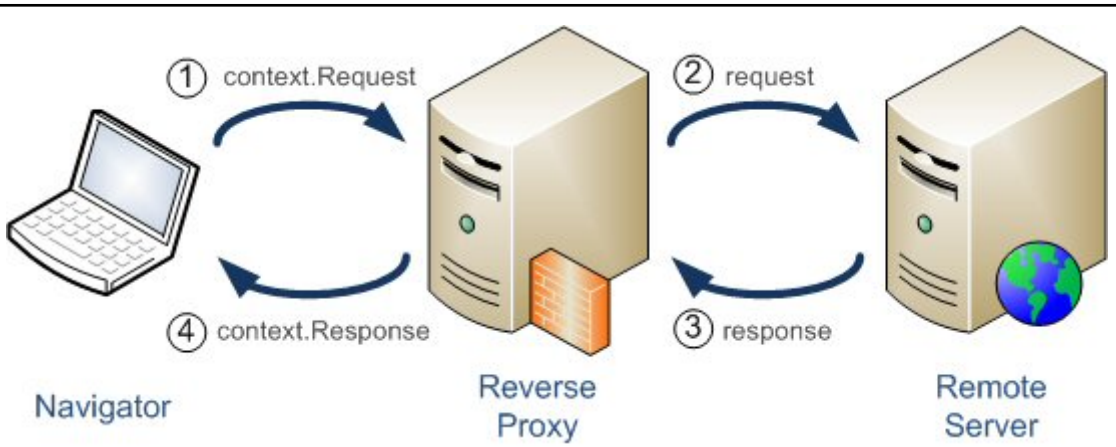
auch als Surrogate-Caches oder Reverse-Proxy-Caches bezeichnet – werden normalerweise von Websitebesitzern oder -managern verwendet, um ihre Websites zuverlässiger und skalierbarer zu machen. Siehe “Varnish Reverse Proxy”

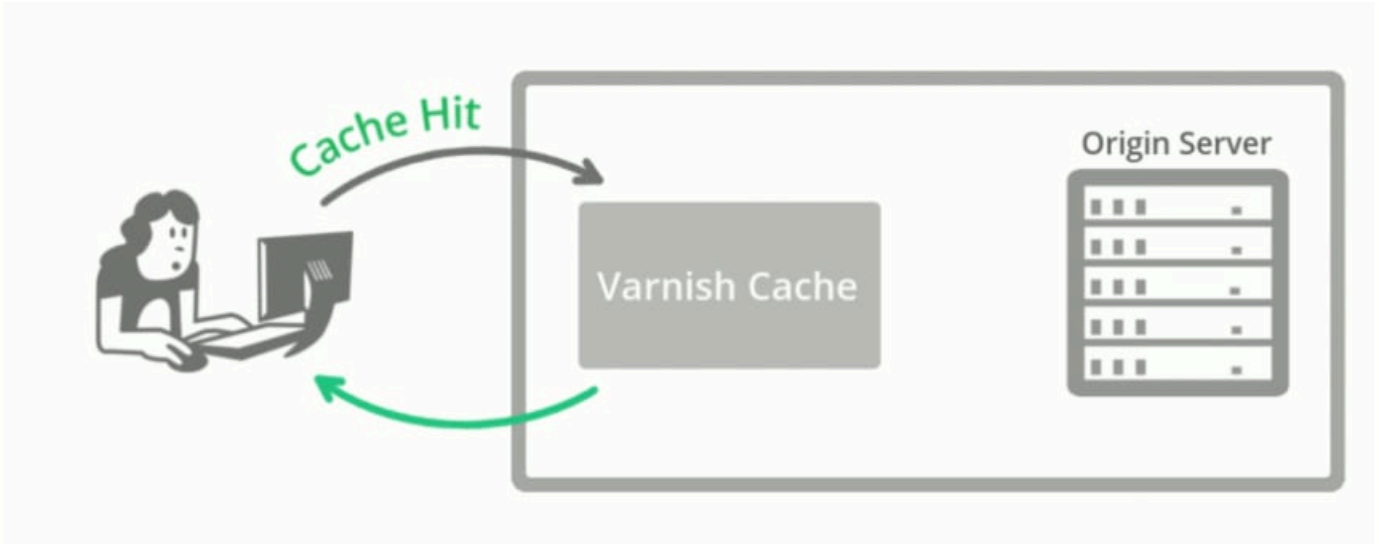
Server-Speichercaches

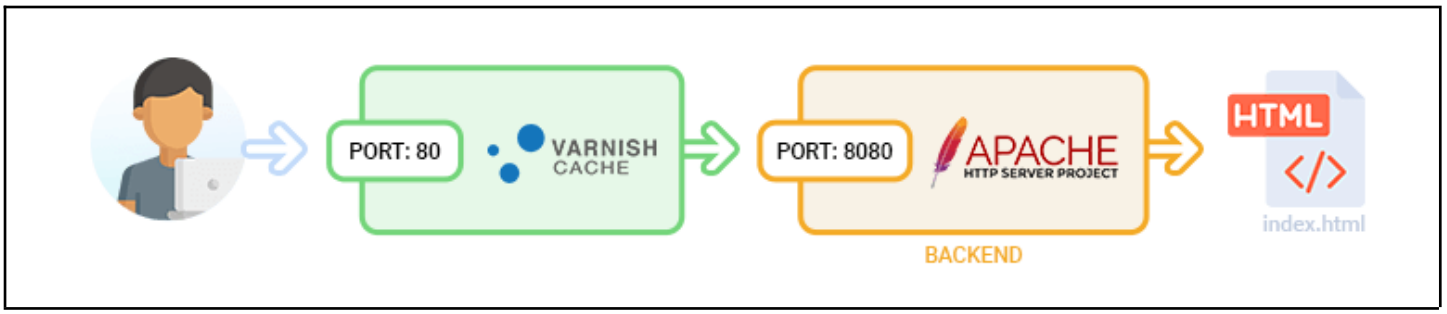
Beispiele hierfür sind Memcached und Varnish. Sie können auf genau demselben lokalen Host wie die Ursprungsquelle des Inhalts ausgeführt werden und erzeugen statisch zwischengespeicherte Kopien dynamisch generierter Seiten, um deren erneute Generierung bei der nächsten Anforderung zu vermeiden.

Varnish Reverse Proxy









Dynamische Webprojekte verlieren mit zunehmender Komplexität und steigenden Nutzerzahlen deutlich an Performance. Um den Webserver zu entlasten und dem Geschwindigkeitsverlust entgegenzuwirken, kann man einen Reverse Proxy verwenden, der die Anfragen an den Webserver stellvertretend beantwortet. Dazu speichert er das angeforderte Material – statische Inhalte wie Bilder sowie Ergebnisse häufig aufgerufener, dynamisch erstellter Seiten – zunächst in seinem Cache. Eine sehr beliebte Caching-Software ist Varnish. Im Gegensatz zu vielen Konkurrenten wurde die freie Software von Grund auf als Webbeschleuniger entwickelt. Die Nutzung von Varnish Cache setzt einen Webserver mit installiertem Unix-Betriebssystem sowie Root-Rechte zur Installation und Konfiguration voraus.

So funktioniert Varnish Cache

Varnish wird als Reverse Proxy direkt vor den Webserver geschaltet, auf dem sich die Website-Inhalte befinden. Kommt es nun zu einem Seitenaufruf, wird dieser zunächst noch vom ursprünglichen Server verarbeitet, der Varnish Proxy speichert jedoch die Anfrage und erforderliche Inhalte. Bei einem erneuten Aufruf dieser Art können die Daten dann direkt aus dem Varnish Cache geladen werden. Die Software legt dabei sämtliche Daten im Arbeitsspeicher ab und lässt das Betriebssystem entscheiden, was auf die Festplatte des Servers ausgelagert werden soll. So wird vermieden, dass das System gleichzeitig Daten im Cache und auf der Festplatte speichert.
Varnish fungiert zudem als Load Balancer. Nach dem Round-Robin-Verfahren werden die eingehenden Anfragen der Clients jeweils als separate Arbeits-Threads (Arbeitsschritte) gewertet, die nacheinander vom Varnish Cache abgehandelt werden. Ein festgelegtes Limit bestimmt, wie viele gleichzeitig aktive Threads bearbeitet werden können. Ist dieses erreicht, gelangen alle weiteren Anfragen in eine Warteschlange. Erst wenn auch das Limit der Warteschlange erreicht ist, werden eingehende Verbindungen blockiert.
Die Konfiguration des Varnish-Reverse-Proxys wird hauptsächlich über die Varnish Configuration Language (VCL) gesteuert. Diese ermöglicht es, Hooks (Schnittstellen) zu schreiben, mit deren Hilfe sich fremder Code in die Anwendung integrieren lässt. Lädt ein solches VCL-Skript, wird es in die Programmiersprache C übersetzt und in eine Programmbibliothek kompiliert; die VCL-Anweisungen werden mit dem Varnish Cache verknüpft. Wenn das verwendete CMS, die eingesetzte Shop-Software oder die zugrundeliegende Webapplikation die Auszeichnungssprache ESI (Edge Side Includes) beherrschen, kann Varnish außerdem als Ganzes zwischengespeicherte Seiten ausliefern. Die Auszeichnungssprache erzeugt in den HTML-Dateien der

Seiten sogenannte ESI-Tags, die dynamische Inhalte auszeichnen. Bei einer Client-Anfrage erkennt der Varnish Cache diese Tags und lädt die entsprechenden Inhalte nach.

Vor- und Nachteile von Varnish Hosting

Die eigene Hosting-Lösung mit einem Varnish Cache zu optimieren, kann in vielen Fällen die Antwort auf die wachsende Komplexität Ihres Projektes und steigende Besucherzahlen sein. Allerdings ist der Einsatz der Software nicht für alle Webpräsenzen zu empfehlen. Zum besseren Überblick haben wir die Vor- und Nachteile von Varnish Hosting für Sie zusammengefasst:

Vorteile:	Nachteile:
schnellere Ladezeiten dank Inhalten im Arbeitsspeicher	für Systeme, die ESI nicht beherrschen, bietet Varnish Cache keine wesentliche Optimierung
Entlastung des Webservers	erhöhte Komplexität und Fehleranfälligkeit
ESI-Unterstützung	keine Unterstützung von TLS/SSL bzw. HTTPS
Betriebssystem lagert die Inhalte auf Server-Festplatte aus	Einrichtung und Konfiguration ist sehr aufwendig und erfordert entsprechendes Know-how
Lastverteilung nach dem Round-Robin-Verfahren	nur für Unix-Betriebssysteme
flexible Konfigurationsmöglichkeiten dank VCL	

Die Gegenüberstellung macht deutlich, dass Varnish Hosting für Sie nur dann eine Alternative zu den vorhandenen Caching-Funktionen von Clients und Webservern darstellt, wenn Sie mit einer Webapplikation, die die Auszeichnungssprache ESI beherrscht, arbeiten. Ferner ist die Einrichtung und Konfiguration des Varnish Cache inklusive der ESI-Tags mit einem hohen Aufwand verbunden. Da Varnish keine TLS-/SSL-Verbindungen unterstützt, benötigen Sie einen weiteren Proxy-Server für die sichere Übertragung. Ein richtig konfigurierter Varnish Cache inklusive ESI-Tags verspricht Ihnen allerdings eine Beschleunigung Ihres Webprojektes, die mit gängigen Caching-Methoden nicht erreicht wird. Das verringert die Ladezeiten für Ihre Besucher enorm und wird Ihnen dadurch langfristig zu einer wesentlich höheren Conversion-Rate verhelfen. Zusätzlich profitieren Sie automatisch von einem besseren Suchmaschinenranking und entlasten enorm den Webserver, der nicht mehr allein für das Bearbeiten eingehender Verbindungen zuständig ist. Insbesondere bei Betreibern von Onlineshops und Webpräsenzen mit einer Vielzahl an Inhalten erfreut sich Varnish Hosting daher einer sehr großen Beliebtheit.

Festlegen des Cache-Zeitpunkts

Caching klingt also, als wäre es enorm vorteilhaft, da es sowohl die Netzwerkverzögerung (Lag) als auch die Serverlast reduziert. Allerdings hat Caching auch seine Grenzen. Ein Caching-System muss über mindestens zwei Schlüsselfunktionen verfügen:

- Zeitlimits für die Dauer der Zwischenspeicherung eines Elements (z. B. Datei, HTTP-Anfrage usw.)
- Ein System, das bestimmt, ob eine bestimmte Anfrage mit einer zwischengespeicherten Kopie der Daten übereinstimmt („hits, trifft “) (daher schnelle Antwort) oder diese “misses, verfehlt“ (daher müssen die Daten neu geladen werden).

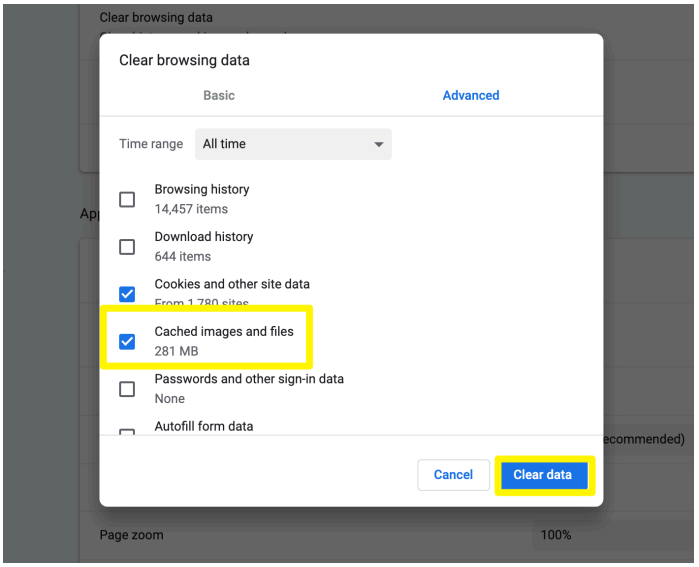
Zeitlimits

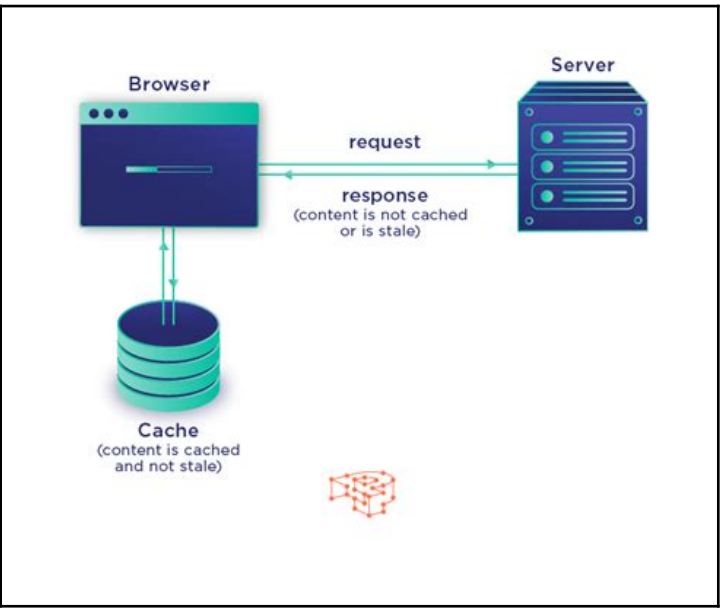
Ressourcen ändern sich und werden aktualisiert, daher muss der Cache für jede zwischengespeicherte Ressource eine begrenzte Lebensdauer haben. Wir wären nicht erfreut, wenn wir am 5. Mai die News-Webseite aufrufen und feststellen würden, dass die Nachrichten, die wir lesen, vom September des Vorjahres stammen. Cache-Elemente haben daher Ablaufzeitbeschränkungen.

Treffer und Fehlschüsse

Wenn ein Cache eine Anfrage für eine Ressource erhält, muss er entscheiden, ob er bereits eine Kopie dieser genauen Ressource gespeichert hat und damit antworten kann oder ob er die Anfrage an den Anwendungsserver weiterleiten muss. Bei statischen Inhalten wie Bildern kann dies einfach sein – ein Katzenfoto ist ein Katzenfoto, egal wer es anfordert. Manche Seiten sind jedoch dynamisch – wenn wir unser E-Banking aufrufen, möchten wir unseren aktuellen eigenen Kontostand sehen, nicht den eines anderen (und umgekehrt). Caches können also nicht einfach alle Inhalte für alle Benutzer zwischenspeichern.
Caches und ihre Ursprungsquellen gehen dieses Problem gemeinsam an, indem sie eine vereinbarte Anordnung von „Cache-Schlüsseln“ verwenden – einige spezifische Komponenten einer HTTP-Anfrage, die zur eindeutigen Identifizierung der angeforderten Ressource dienen. Dies kann manuell mithilfe von HTTP-Headern erfolgen. Der Programmierer kann sicherstellen, dass der Ursprungsserver Header zurückgibt, um eine Ressource als öffentlich oder privat zu kennzeichnen, ein Höchstalter für die Speicherung festlegen, den Browser auffordern, diese Ressource jedes Mal neu zu validieren oder den Browser anweisen, eine bestimmte Ressource nicht im Cache zu speichern.

Der Browser Cache





Der Browser-Cache speichert temporär Dateien von besuchten Internetseiten, um Ladezeiten zu verkürzen und Bandbreite zu sparen. Dies kann zu folgenden Tücken führen:

- Veraltete Inhalte: Der Browser kann eine zwischengespeicherte Version einer Seite anzeigen, auch wenn eine neuere verfügbar ist.
- Nicht aktualisierte Einstellungen: Änderungen auf einer Webseite können nicht sofort sichtbar sein, wenn der Browser immer noch die gecachte Version anzeigt.
- Cookies von Drittanbietern: Der Cache kann Cookies von Drittanbietern speichern, die unnötig sind oder sogar schädlich sein können.

Um diese Tücken zu umgehen, kannst du:

- Den Cache manuell leeren: In den Einstellungen deines Browsers kannst du den Cache löschen, um sicherzustellen, dass du die neueste Version einer Seite siehst.
- Cookies von Drittanbietern blockieren: In den Cookie-Einstellungen kannst du bestimmte Cookies blockieren, um unnötige Cache-Elemente zu entfernen.
- Den Browser so konfigurieren, dass er den Cache ignoriert: Manuell umgehen des Caches kann helfen, sicherzustellen, dass du die neueste Version einer Seite siehst.

Technische Aspekte und Handhabung für Entwickler

Der Browser-Cache speichert temporär Dateien wie HTML-Dokumente, CSS-Stile, JavaScript-Dateien, Bilder und andere Ressourcen von besuchten Websites, um die Ladezeiten zu verkürzen und die Bandbreitennutzung zu reduzieren. Dies verbessert die Benutzererfahrung, kann aber auch zu Problemen führen, wenn veraltete oder zwischengespeicherte Inhalte angezeigt werden. Hier sind einige technische Details und konkrete Tipps für die Handhabung als Entwickler:

HTTP Caching-Mechanismen und Header-Steuerung

Der Browser-Cache wird durch verschiedene HTTP-Header gesteuert, die vom Server an den Browser gesendet werden. Zu den wichtigsten Headern gehören:

- Cache-Control: Bestimmt, wie und für wie lange Ressourcen gecacht werden sollen. Mögliche Werte sind `no-cache`, `no-store`, `public`, `private`, `max-age`, usw. Beispielsweise bedeutet `Cache-Control: max-age=3600`, dass die Ressource für 3600 Sekunden (eine Stunde) gecacht werden kann.
- ETag (Entity Tag): Ein eindeutiger Identifikator für eine spezifische Version einer Ressource. Der Browser speichert den ETag und sendet ihn bei einer erneuten Anfrage zurück, sodass der Server prüfen kann, ob sich die Ressource geändert hat.
- Expires: Ein veralteter, aber immer noch verwendeter Header, der das Datum und die Uhrzeit angibt, nach dem eine zwischengespeicherte Version als veraltet angesehen wird.
- Last-Modified: Gibt an, wann die Ressource zuletzt geändert wurde. Der Browser kann diese Information verwenden, um festzustellen, ob eine neuere Version verfügbar ist.

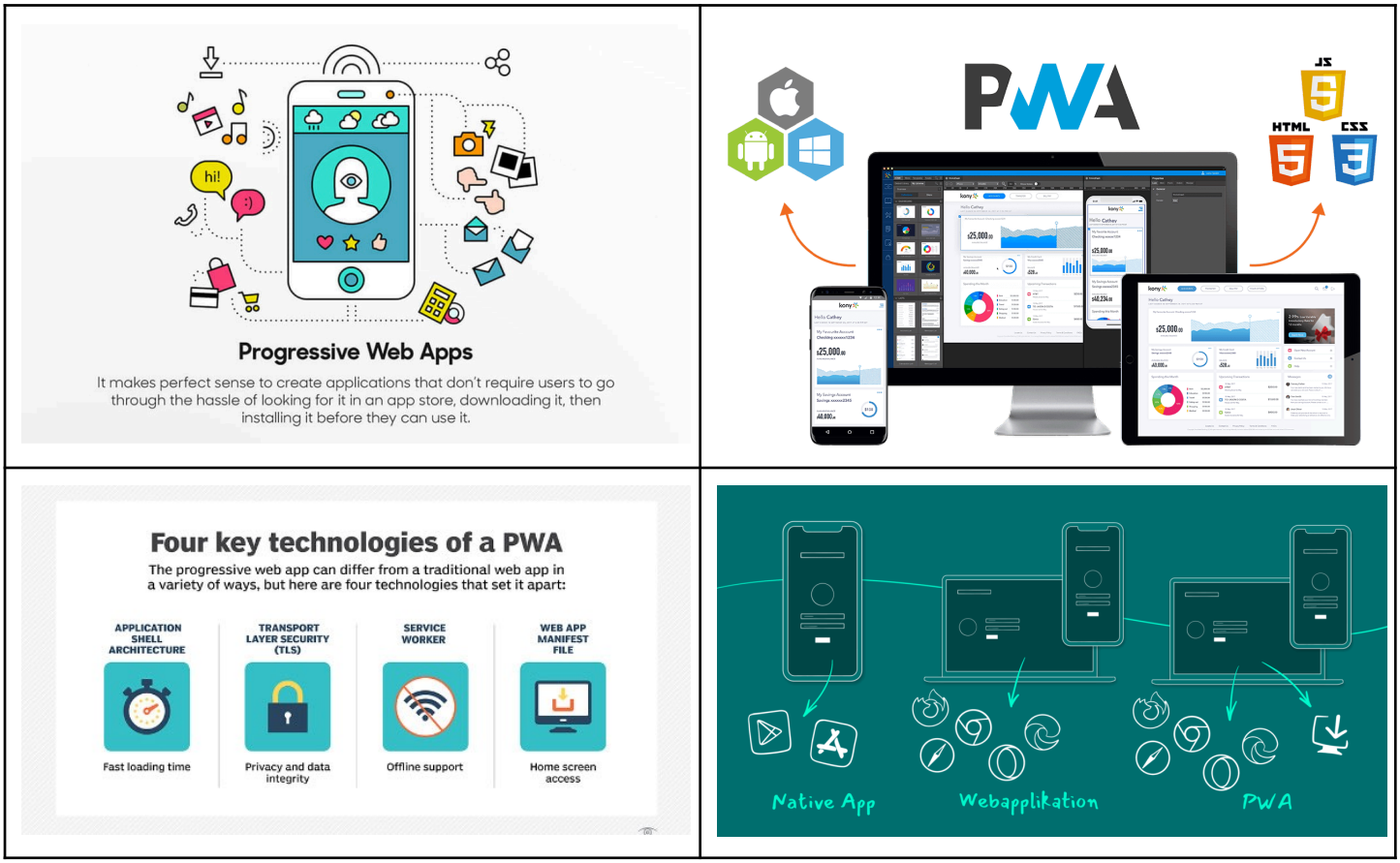
Cache-Busting-Techniken

Als Entwickler kannst du „Cache Busting“ verwenden, um sicherzustellen, dass Benutzer immer die aktuelle Version deiner Website oder Anwendung sehen. Häufige Techniken sind:

- Versionsnummern in URLs: Indem du eine Versionsnummer oder ein Hash in den Dateinamen einfügst (z. B. `styles.v1.css` oder `script.abc123.js`), wird der Browser gezwungen, die neue Version zu laden, sobald sich die Datei ändert.
- Query-Strings: Manchmal werden auch Query-Strings (`?v=1.0.1`) an die URL der Ressource angehängt. Diese Methode funktioniert jedoch nicht immer zuverlässig, da einige Proxy-Server Query-Strings ignorieren.

Progressive Web Apps (PWAs)

Progressive Web Apps (PWAs) sind Webanwendungen, die moderne Webtechnologien nutzen, um eine native App-ähnliche Benutzererfahrung zu bieten. Sie kombinieren die besten Eigenschaften von Websites und mobilen Apps, indem sie schnell, zuverlässig und ansprechend sind. Sie werden durch den Aufruf einer Webseite im Browser installiert (auf Anfrage und als Icon auf dem Desktop oder in der Browserleiste etc.). Diese sind aber nicht in den App-Stores zu finden sondern nur direkt über die Webseite zugänglich.



Wesentliche Merkmale von PWAs:

- Installierbar: PWAs können wie native Apps auf dem Startbildschirm eines Geräts installiert werden, ohne dass sie über einen App Store heruntergeladen werden müssen.
- Offline-Fähigkeit: Mithilfe von Service Workern können PWAs auch ohne Internetverbindung funktionieren, indem sie Ressourcen zwischenspeichern und so eine nahtlose Benutzererfahrung gewährleisten.
- Reaktionsfähig: Sie passen sich an unterschiedliche Geräte und Bildschirmgrößen an, egal ob auf einem Desktop, Tablet oder Smartphone.
- Schnell und zuverlässig: Durch optimiertes Caching und asynchrone Datenübertragung bieten PWAs schnelle Ladezeiten und eine hohe Leistung, selbst bei schlechter Netzwerkverbindung.

- Aktuell: PWAs sind immer auf dem neuesten Stand, da sie automatisch aktualisiert werden, wenn der Benutzer sie öffnet.
- Sicher: PWAs werden über HTTPS bereitgestellt, um eine sichere Verbindung zu gewährleisten und Datenmanipulation zu verhindern.

Vorteile von PWAs:

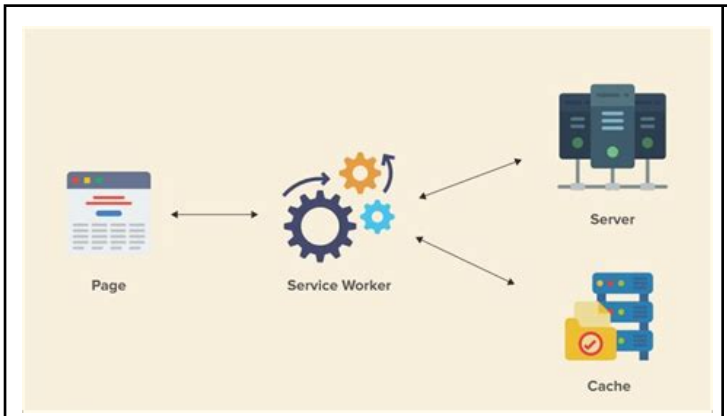
- Kosteneffizient: Einmal entwickelt, funktionieren sie auf allen Plattformen (iOS, Android, Desktop) ohne zusätzliche Anpassung.
- Einfache Bereitstellung: Kein App Store erforderlich, Updates werden automatisch verteilt.
- Verbesserte Nutzererfahrung: Schnelle Ladezeiten, offline nutzbar, Push-Benachrichtigungen und ein natives App-Feeling.

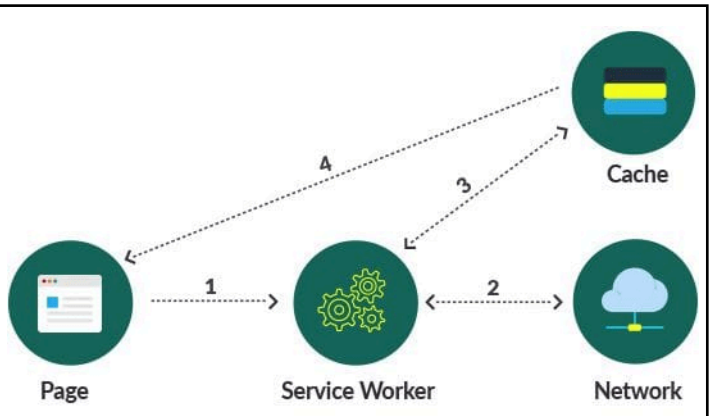
PWAs sind eine zukunftsweisende Lösung, die die Flexibilität des Webs mit der Benutzerfreundlichkeit von nativen Apps vereint.

Caching durch Service Worker

Stellt Offline-Fähigkeit und andere Hintergrundaufgaben zur Verfügung. Ein Service Worker ist ein JavaScript-Programm, das ein Web-Browser im Hintergrund ausführt. Es stellt für Progressive Web Apps essentielle Funktionen wie das Caching für die Offline-Verwendbarkeit bereit. Einmal online abgerufen, können Inhalte beim nächsten Besuch der Seite auch ohne Internetverbindung angezeigt werden (Offline-Betrieb). Auch von nativen Apps bekannte Push-Benachrichtigungen sind mit Service Workern möglich. Service Worker werden eigens programmiert, im JavaScript der Seite registriert und installiert. Service Worker bedingen HTTPS, weshalb jede Progressive Web App mit HTTPS läuft. Zahlreiche Frameworks, wie z. B. Angular mit dem Mobile Toolkit, stellen Service Worker bereit, so dass man diese nicht selbst entwickeln muss.

Service Worker sind also Skripte, die im Hintergrund eines Browsers laufen und Webentwicklern ermöglichen, spezifische Aufgaben unabhängig von einer geöffneten Webseite auszuführen. Sie sind ein wesentlicher Bestandteil von Progressive Web Apps (PWAs) und bieten eine programmatische Kontrolle über Caching-Strategien und Netzwerkoperationen, was zu verbesserter Offline-Fähigkeit und Performance von Webanwendungen führt.





Wie funktionieren Service Worker im Kontext von Caching?

Ein Service Worker agiert als eine Art „Proxy-Server“ zwischen dem Browser und dem Netzwerk. Er fängt Netzwerkanfragen ab und entscheidet anhand von vorgegebenen Regeln, ob die angeforderte Ressource:

- Aus dem Cache geladen werden soll, falls sie bereits gecacht ist,
- Vom Netzwerk abgerufen werden soll, oder
- Beide Strategien kombiniert werden, um eine effiziente Datenbereitstellung zu gewährleisten.

Dies ermöglicht eine gezielte Kontrolle darüber, wie und wann Inhalte bereitgestellt werden, was insbesondere

für PWAs wichtig ist, die auch offline oder bei schlechten Netzwerkbedingungen zuverlässig funktionieren sollen.

Bei der Entwicklung von PWAs wird der Browser-Cache durch Service Worker erweitert, die eine noch feinere Kontrolle über das Caching und die Netzwerkanfragen ermöglichen. Service Worker können programmatisch entscheiden, wann eine Ressource aus dem Cache geladen oder aus dem Netzwerk abgerufen werden soll, indem sie sicherstellen, dass:

- Cache-first, Network-fallback: Statische Inhalte effizient aus dem Cache geladen werden, auch wenn der Benutzer offline ist.
- Network-first, Cache-fallback: Dynamische Inhalte bei Verfügbarkeit aus dem Netzwerk aktualisiert werden, mit einem Fallback zum Cache, wenn das Netzwerk nicht verfügbar ist.

Durch diese Flexibilität wird die Benutzererfahrung in PWAs optimiert, unabhängig von den Netzwerkbedingungen. Dies verbessert die Offline-Fähigkeit und die Performance von Webanwendungen erheblich.

Konkrete Tipps für die Handhabung des Browser-Caches als Entwickler

Effiziente Nutzung von Caching-Headern

- Verwende `Cache-Control` und `ETag` sorgfältig, um sicherzustellen, dass statische Ressourcen (wie Bilder, CSS und JS-Dateien) angemessen zwischengespeichert werden, während dynamische Inhalte häufiger aktualisiert werden.
- Setze `no-cache` für dynamische Seiteninhalte, die sich häufig ändern, und `max-age` für statische Ressourcen, die selten aktualisiert werden.

Cache Invalidierung planen

- Entwickle eine Strategie zur Cache-Invalidierung, die klar definiert, wann und wie oft verschiedene Arten von Inhalten aktualisiert werden sollen. Dies kann durch automatisierte Build-Prozesse oder durch manuelle Versionskontrolle geschehen.

Nutzung von Developer-Tools

- Nutze die Entwicklerwerkzeuge (DevTools) deines Browsers, um den Cache zu debuggen. Unter dem Tab „Network“ kannst du sehen, welche Ressourcen gecacht sind und wie lange sie gültig bleiben. Du kannst auch den „Disable Cache“-Modus aktivieren, um zu testen, wie deine Seite ohne Caching funktioniert.
- Verwende die „Application“-Sektion der DevTools, um den Cache-Status zu überprüfen und zu verwalten. Hier kannst du Cache-Einträge manuell löschen und die Auswirkungen auf die Seite testen.

Strategien für die Cache-Leerung

- Implementiere Mechanismen, um den Cache bei Bedarf zu leeren, etwa durch einen „Clear Cache“-Button auf einer Admin-Seite oder durch die Verwendung von Service-Worker-Updates in PWAs.
- Weise Benutzer darauf hin, den Browser-Cache zu leeren, wenn neue Updates oder wesentliche Änderungen vorgenommen wurden.

Performance-Optimierung durch Caching

- Analysiere regelmäßig die Ladezeiten und Performance deiner Website mithilfe von Tools wie Google PageSpeed Insights, Lighthouse oder WebPageTest.

- Identifiziere Bereiche, in denen das Caching optimiert werden kann, z. B. durch längere Cache-Zeiten für statische Ressourcen oder durch das Vermeiden von unnötigen Cache-Bypassen. Cache-Bypass bedeutet, dass eine Anfrage an eine Ressource absichtlich oder unabsichtlich den Cache überspringt und direkt zum Ursprungsserver geht, anstatt auf eine zwischengespeicherte Version zuzugreifen. Mit anderen Worten: Die Anfrage „umgeht“ den Cache und verursacht eine direkte Abfrage an den Server, was die Vorteile des Cachings (wie reduzierte Ladezeiten und geringere Serverlast) zunichtemacht.

Sicherheitsaspekte berücksichtigen

- Vermeide das Caching von vertraulichen oder sicherheitsrelevanten Daten (z. B. Authentifizierungstokens) durch Verwendung von `Cache-Control: no-store` und `Pragma: no-cache`.
- Implementiere die Sicherstellung, dass sensitive Daten niemals im Cache landen, indem du serverseitige Konfigurationen und Sicherheitsrichtlinien entsprechend anpasst.

Durch die gezielte Verwendung und Verwaltung des Browser-Caches können Entwickler nicht nur die Performance und Nutzererfahrung erheblich verbessern, sondern auch die Sicherheit und Zuverlässigkeit ihrer Webanwendungen erhöhen.

Cache Bypassing

Cache-Bypass bedeutet, dass eine Anfrage an eine Ressource absichtlich oder unabsichtlich den Cache überspringt und direkt zum Ursprungsserver geht, anstatt auf eine zwischengespeicherte Version zuzugreifen. Mit anderen Worten: Die Anfrage „umgeht“ den Cache und verursacht eine direkte Abfrage an den Server, was die Vorteile des Cachings (wie reduzierte Ladezeiten und geringere Serverlast) zunichte macht. Es wird also die Webseite komplett neu geladen.

Was verursacht einen Cache-Bypass?

Ein Cache-Bypass kann auf verschiedene Weise ausgelöst werden:

- Fehlende oder unpassende Cache-Header: Wenn der Server oder die Anwendung keine Cache-Header (wie `Cache-Control` oder `ETag`) korrekt setzt, kann der Browser oder ein Content Delivery Network (CDN) entscheiden, die Anfrage direkt an den Server zu senden, da er nicht weiß, ob die zwischengespeicherte Version noch gültig ist.
- Nutzung von `Cache-Control`-Headern wie `no-cache` oder `no-store`: Diese Header geben an, dass der Cache nicht verwendet werden sollte. Bei `no-cache` muss der Browser jedes Mal beim Server nachfragen, ob die zwischengespeicherte Ressource noch aktuell ist. `no-store` weist den Browser an, die Ressource gar nicht erst zu speichern, sodass sie immer vom Server abgerufen wird.
- Manuelle Benutzeraktionen: Ein Benutzer kann den Cache manuell umgehen, indem er eine Seite in seinem Browser mit speziellen Tastenkombinationen wie `Strg+F5` (auf Windows) oder `Cmd+Shift+R` (auf Mac) neu lädt. Dies weist den Browser an, die Seite direkt vom Server zu laden, ohne den Cache zu berücksichtigen.
- Unterschiedliche URL-Parameter oder Query-Strings: Selbst kleine Änderungen in der URL (z. B. Query-Strings wie `?v=2.1`) können dazu führen, dass die Anfrage nicht mit einer im Cache gespeicherten Version übereinstimmt, was den Browser oder CDN dazu veranlasst, die Ressource erneut vom Server abzurufen.

Warum Cache-Bypasses problematisch sein können

Unnötige Cache-Bypasses können zu mehreren Problemen führen:

- Erhöhte Ladezeiten: Wenn der Cache umgangen wird, muss die Ressource jedes Mal vom Ursprungsserver abgerufen werden, was zu längeren Ladezeiten führt, besonders wenn die Server geografisch weit entfernt sind oder das Netzwerk langsam ist.
- Erhöhte Serverlast: Bei vielen Anfragen, die den Cache umgehen, wird die Serverlast erhöht, was die

Leistung des Servers beeinträchtigen kann, besonders bei stark frequentierten Websites.

- Höhere Bandbreitennutzung: Durch häufigere direkte Anfragen an den Server wird mehr Bandbreite genutzt, was zu höheren Kosten und möglicherweise zu Engpässen führt.

Wie kann man unnötige Cache-Bypasses vermeiden?

Um unnötige Cache-Bypasses zu vermeiden, sollten Entwickler:

- Caching-Header korrekt konfigurieren: Setze die richtigen Cache-Header (`Cache-Control`, `ETag`, `Expires`) basierend auf dem Typ der Ressource (statisch oder dynamisch) und der erwarteten Aktualisierungsfrequenz.
- Längere Cache-Zeiten für statische Ressourcen: Verwende längere `max-age` Werte für statische Ressourcen (wie Bilder, CSS, JS-Dateien), die sich selten ändern, damit sie länger im Cache bleiben und der Cache-Bypass vermieden wird.
- Konsistente URLs für identische Ressourcen sicherstellen: Vermeide unnötige Änderungen in URLs (wie unnötige Query-Strings oder Parameter), die zu Cache-Bypassen führen können.
- Cache-Invalidierung effizient planen: Nutze Cache-Busting-Techniken (wie Versionsnummern oder Hashes in Dateinamen), um gezielt Caches zu leeren, ohne unnötige Bypasses zu erzeugen. Dies stellt sicher, dass nur veraltete Ressourcen neu abgerufen werden.

Durch die Vermeidung unnötiger Cache-Bypasses kann die Performance einer Webanwendung deutlich verbessert und die Serverlast sowie die Bandbreitennutzung minimiert werden.

Fragen zum Browser Cache

Wie wirkt sich die automatische Aktualisierung von Browser-inhalten auf die Cache-Verwaltungs-Wahrscheinlichkeit aus?

Die automatische Aktualisierung von Browser-Inhalten kann die Cache-Verwaltungs-Wahrscheinlichkeit beeinflussen. Einige Aspekte sind zu beachten:

- Zeitintervalle: Wenn Sie zB ein Addon in den Suchergebnissen einsetzen, können Sie einstellen, wie oft der Browser-Inhalt automatisch aktualisiert wird (z.B. alle 5 Sekunden, alle 5 Minuten). Je kürzer das Zeitintervall, desto wahrscheinlicher wird es, dass der Cache veraltet wird, da der Browser-Inhalt häufiger neu geladen wird.
- Cache-Entsorgung: Einige Browser ermöglichen die manuelle Entsorgung des Cache-Speichers. Wenn der Browser-Inhalt automatisch aktualisiert wird, kann dies zu einer vermehrten Cache-Entsorgung führen, da der Browser-Inhalt häufiger neu geladen wird und ältere Versionen des Inhalts entfernt werden.
- Browser-Cache-Verhalten: Der Browser-Cache speichert Inhalte, um schnellen Zugriff zu ermöglichen. Wenn der Browser-Inhalt automatisch aktualisiert wird, kann dies dazu führen, dass der Cache ältere Versionen des Inhalts enthält, bevor diese aktualisiert werden. Dies kann die Cache-Veraltungs-Wahrscheinlichkeit erhöhen.
- Proxy-Einstellungen: Wenn ein Proxy-Server eingesetzt wird, kann dies die Cache-Veraltungs-Wahrscheinlichkeit beeinflussen. Einige Proxy-Server können den Browser-Cache blockieren oder ändern, was die Automatisierung des Inhaltsaktualisierens beeinträchtigen kann.

Insgesamt kann die automatische Aktualisierung von Browser-Inhalten die Cache-Veraltungs-Wahrscheinlichkeit erhöhen, insbesondere wenn kurze Zeitintervalle gewählt werden oder wenn der Browser-Cache nicht ordnungsgemäß geleert wird. Es ist jedoch wichtig, die spezifischen Einstellungen und Verhaltensweisen des verwendeten Browsers und Proxy-Server zu berücksichtigen, um die Cache-Veraltungs-Wahrscheinlichkeit zu minimieren.

Welchen Einfluss haben Browser-Einstellungen auf die Wahrscheinlichkeit, dass der Cache veraltet ist?

Die Wahrscheinlichkeit, dass der Cache veraltet ist, wird von Browser-Einstellungen beeinflusst. Hier sind einige wichtige Faktoren:

- Cache-Löschen: Wenn du den Cache manuell leeren kannst, wie in den Einstellungen deines Browsers, reduzierst du die Wahrscheinlichkeit, dass veraltete Inhalte angezeigt werden.
- Cookie-Einstellungen: Blockiere Cookies von Drittanbietern, um unnötige Cache-Elemente zu entfernen und die Wahrscheinlichkeit von Veraltungen zu reduzieren.
- Cache-Verfallsdatum: Wenn dein Browser ein Verfallsdatum für Cache-Inhalte setzt, wird er automatisch alte Inhalte entfernen, wenn sie überholt sind. Dies kann die Wahrscheinlichkeit von Veraltungen reduzieren.
- Cache-Control: Einige Browser ermöglichen es, Cache-Control-Header zu setzen, die angeben, ob der Cache für bestimmte Ressourcen verwendet werden soll oder nicht. Ein Wert wie “no-cache” kann die Wahrscheinlichkeit von Veraltungen erhöhen, da der Browser immer den Server anfragt, bevor er eine Ressource aus dem Cache lädt.

Wie beeinflussen Cache-Strukturen die Geschwindigkeit und Zuverlässigkeit von Web-Applikationen?

Cache-Strukturen beeinflussen die Geschwindigkeit und Zuverlässigkeit von Web-Applikationen signifikant. Ein Cache speichert häufig genutzte Daten zwischen, um den Zugriff auf diese Daten zu beschleunigen. Dieser Mechanismus bietet folgende Vorteile:

- Geschwindigkeit: Durch die Suche in einem Cache vor dem Aufruf an den Ursprungsserver kann die Ausführungszeit von Anfragen reduziert werden. Eine Beschleunigung um einen Faktor von einhundert ist nicht ungewöhnlich, insbesondere bei Anwendungen mit vielen wiederkehrenden Datenanfragen.
- Lastverteilung: Caches können Lastverteilung innerhalb eines Clustern von Servern oder zwischen verschiedenen Servern ermöglichen. Dadurch kann die Belastung einzelner Server reduziert und die Verfügbarkeit der Anwendung verbessert werden.
- Reduzierung von Netzwerkübertragungen: Durch die lokale Speicherung von Daten im Cache reduziert sich der Bedarf an Netzwerkübertragungen, was zu einer Verbesserung der Verfügbarkeit und Zuverlässigkeit der Anwendung beiträgt.

Es gibt verschiedene Arten von Caches, wie z.B.:

- Browser-Cache: Der Browser speichert lokale Kopien von Ressourcen, wie Bildern oder JavaScript-Dateien, um die Wiederholung von Anfragen zu reduzieren.
- Webserver-Cache: Der Webserver speichert Antworten auf Anfragen in einem Cache, um die Auslieferung von Inhalten zu beschleunigen.
- Datenbank-Cache: Eine Datenbank speichert häufig genutzte Daten in einem Cache, um die Ausführungszeit von Anfragen zu verbessern.

Es ist jedoch wichtig zu beachten, dass Caches auch einige Nachteile haben können, wie z.B.:

- Invalidation: Wenn Daten im Cache veraltet sind, kann dies zu inkonsistenten Ergebnissen führen.
- Speicherplatz: Caches benötigen Speicherplatz, der jedoch oft begrenzt ist.

Insgesamt spielen Cache-Strukturen eine wichtige Rolle bei der Optimierung der Geschwindigkeit und Zuverlässigkeit von Web-Applikationen. Es ist jedoch entscheidend, die Implementierung eines Caches sorgfältig zu planen und zu konfigurieren, um die Vorteile zu maximieren und die Nachteile zu minimieren.

Kann ein zu hoher Cache-Anteil zu einer verminderten Datenschutz-Sicherheit führen?

Der Cache kann indirekt zu einer verminderten Datenschutz-Sicherheit beitragen, wenn er zu groß wird oder nicht ordnungsgemäß verwaltet wird. Hier sind einige Aspekte zu beachten:

- Cookies und personenbezogene Daten: Der Cache kann Cookies von Drittanbietern speichern, die personenbezogene Daten enthalten. Wenn dieser Cache nicht regelmäßig geleert wird, können diese Daten langfristig gespeichert bleiben und somit die Datenschutz-Sicherheit gefährden.
- Unnötige Daten: Ein zu großer Cache kann auch unnötige Daten speichern, wie z.B. Eingaben von Formularen oder Suchbegriffe. Diese Daten können potenziell verwendet werden, um die Privatsphäre von Benutzern zu verletzen.
- Sicherheitslücken: Ein nicht ordnungsgemäß konfiguriertes Cache-System kann auch Sicherheitslücken aufweisen, die es Angreifern ermöglichen, auf gespeicherte Daten zuzugreifen.

Um den Datenschutz-Sicherheitsrisiken durch einen zu großen Cache-Anteil entgegenzuwirken, sollten Sie:

- Regelmäßig den Cache leeren: Stellen Sie sicher, dass Sie den Cache regelmäßig leeren, um unnötige Daten zu entfernen und die Datenschutz-Sicherheit zu verbessern.
- Cookies von Drittanbietern kontrollieren: Überprüfen Sie die Cookie-Einstellungen Ihres Browsers und blockieren Sie Cookies von Drittanbietern, die personenbezogene Daten enthalten.
- Cache-System ordnungsgemäß konfigurieren: Stellen Sie sicher, dass das Cache-System ordnungsgemäß konfiguriert ist und Sicherheitslücken minimiert werden.

Welche Rolle spielen HTTP-Cache-Control-Headers bei der Festlegung von Cache-Lifespans und -Verfallsdaten?

Die HTTP-Cache-Control-Headers spielen eine entscheidende Rolle bei der Festlegung von Cache-Lifespans und -Verfallsdaten. Sie ermöglichen es Clients, eine gecachte Ressource zu speichern und zu verwenden, solange sie noch gültig ist.

Die wichtigsten Cache-Control-Headers für die Festlegung von Cache-Lifespans sind:

- max-age: Spezifiziert die maximale Zeit in Sekunden, für die die Ressource im Cache gespeichert werden darf. Zum Beispiel Cache-Control: max-age=600 bedeutet, dass die Ressource 10 Minuten lang im Cache gültig bleibt.
- Expires: Angeben eines bestimmten Datums und Uhrzeiten, ab dem die Ressource nicht mehr gültig ist. Es wird jedoch nicht mehr empfohlen, Expires zu verwenden, da es von Clients ignoriert werden kann, wenn sie die max-age-Einstellung vorziehen.

Weitere Cache-Control-Headers, die bei bestimmten Anwendungen eingesetzt werden können, sind:

- no-cache: Verhindert das Caching der Ressource und fordert den Client, jede Anfrage an den Server zu senden, um die neueste Version zu erhalten.
- must-revalidate und proxy-revalidate: Sichern die Gültigkeit der Ressource durch eine Zwischenspeicherung an einem Proxy-Server oder direkt am Client. Wenn die Ressource abgelaufen ist, muss sie vom Server neu generiert werden.

Insgesamt ermöglichen die HTTP-Cache-Control-Headers eine feine Einstellung der Cache-Lifespans und -Verfallsdaten, um die Leistung und Zuverlässigkeit einer Anwendung zu verbessern.

Wie beeinflussen verschiedene Browser die Cache-Veraltungs-Wahrscheinlichkeit?

Die Cache-Veraltungs-Wahrscheinlichkeit wird von verschiedenen Browser-Typen wie Chromium und Firefox beeinflusst. Obwohl die grundlegenden Mechanismen ähnlich sind, gibt es Unterschiede in den Einstellungen und Funktionen, die die Wahrscheinlichkeit von Cache-Veraltungen beeinflussen.

Chromium-basierte Browser (z.B. Chrome, Brave)

- Diese Browser verwenden einen gemeinsamen Cache-Manager, der die Cache-Veraltungs-Wahrscheinlichkeit beeinflusst.
- Die Cache-Löschen-Funktion kann über die Entwicklertools oder Chrome-Erweiterungen aktiviert werden.
- Chromium-basierte Browser ignorieren den Server, wenn der Cache-Inhalt nicht veraltet ist und den angeforderten Inhalt enthält.

Firefox

- Firefox verwendet einen eigenen Cache-Manager und bietet eine umfangreichere Einstellungsvariablität.
- Die Cache-Löschen-Funktion kann über das Menü “Zu löschende Zeitspanne” im Popup-Fenster erreicht werden, das durch die Tastenkombination CTRL + SHIFT + DEL aufgerufen wird.
- Firefox kann automatisch den Cache leeren, wenn der Browser geschlossen wird, wenn das Kontrollkästchen “Cookies und Websitedaten löschen, wenn Firefox geschlossen wird” aktiviert ist.

Gemeinsame Aspekte

- Beide Browser-Typen verwenden Cache-Verfallsdaten, um alte Inhalte zu entfernen, wenn sie veraltet sind.
- Die Cache-Veraltungs-Wahrscheinlichkeit kann reduziert werden, indem Cookies von Drittanbietern blockiert oder bestimmte URLs hinzugefügt werden, um sie entweder zu blockieren oder zuzulassen.

Insgesamt beeinflussen die Einstellungen und Funktionen von Chromium-basierten Browsern wie Chrome und Brave sowie Firefox die Cache-Veraltungs-Wahrscheinlichkeit. Durch sorgfältige Einstellungen kann die Wahrscheinlichkeit von Veraltungen reduziert werden und sicherstellen, dass du die neuesten Inhalte siehst.