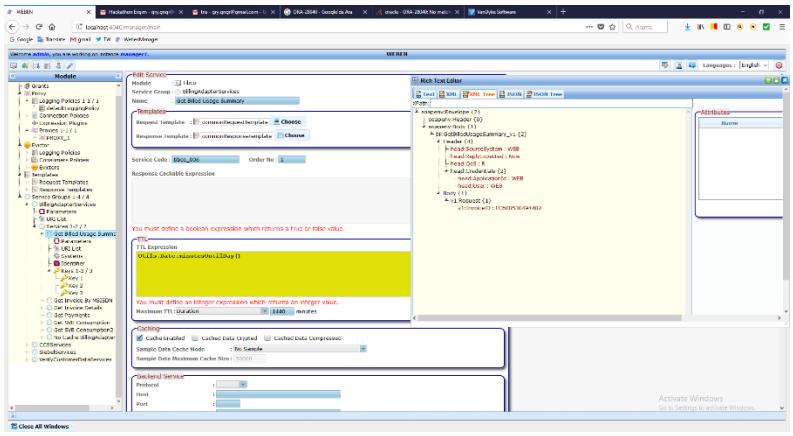
WEBEN-ANWENDUNG

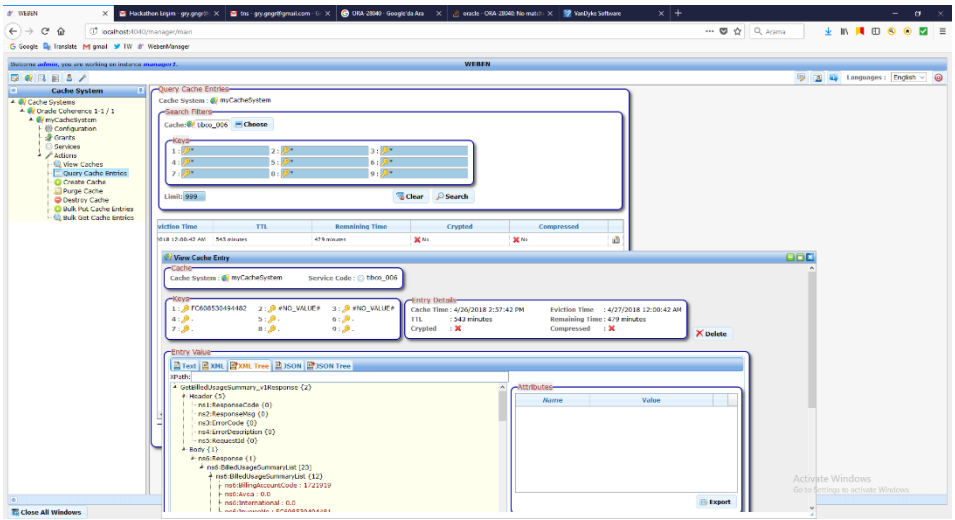
Definition

Weben ist eine Unternehmensanwendung, die darauf abzielt, die Antwortzeit zu verringern und die Leistung aller Arten von http-basierten Abfragediensten (z. B. SOAP, RESTFul) zu erhöhen, die von Diensteanbietersystemen bereitgestellt werden, indem die Dienstdaten in einem Cachesystem gespeichert und zurückgegeben werden Antwort vom Cache, wenn die gleiche Abfrage erneut ausgeführt wird, und Verringerung der Belastung des Diensteanbietersystems durch Verringerung des Ressourcenverbrauchs wie CPU, Speicher, E / A.

Bedürfnisse die von Weben erfüllt werden

Der bei Weben implementierte Ansatz besteht im Wesentlichen darin, die Daten, die der Client abfragt, zusammen mit den Eingabekriterien in dem Cache zu halten. In diesem Fall wird diese Anfrage, wenn dieselbe Anfrage mit demselben Eingabekriterium empfangen wird, direkt an den Back-End-Systemen über den Cache gesendet und an die Client-Anwendung übergeben. Dies führt zu einer signifikanten Verbesserung der Antwortzeit (Leistungsverbesserung) sowie einer Reduzierung der Back-End-Systemressourcen (CPU, Speicher, Festplatte, Netzwerk) für beide Client-Anwendungsantworten. Das Erhöhen der Leistung und das Verringern der Last ist das Maß für den Erfolg, auf den die Weben-Anwendung abzielt. Dieses Maß ist wie bei allen Cache-Systemen direkt





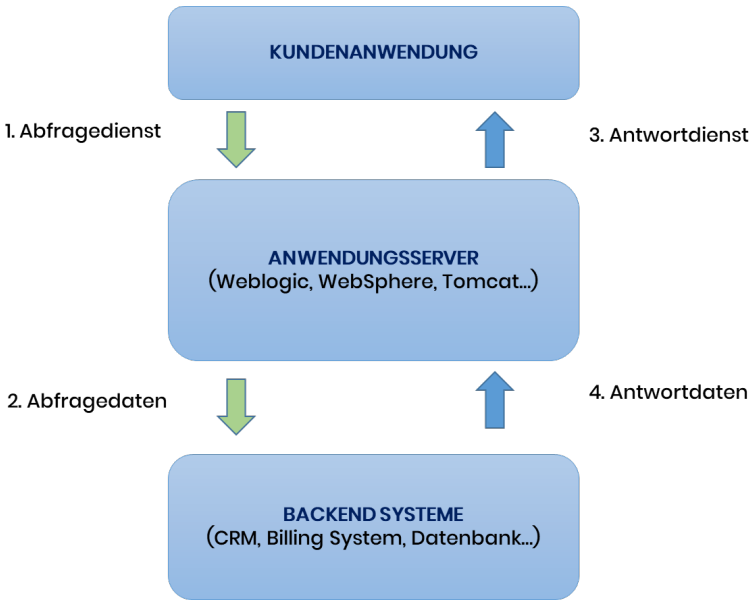
proportional zur Cache-Trefferrate.

Eines der Probleme, die mit Weben gelöst werden müssen, ist das Räumungskonzept (eviction). Die Definition der Räumung erfolgt durch Löschen der Daten im Cache, die ablaufen. Die im Cache platzierten Daten haben eine Lebensdauer und sollten daher aus dem Cache gelöscht werden. Daten gehen verloren und werden aktualisiert. Andernfalls kehren sowohl der abfragende Client mit seiner Korrektheit und verlorenen Informationen zur Anwendung zurück, was dazu führt, dass unnötige Daten im Cache gesammelt werden.

Um dieses Problem zu lösen, gibt es zwei Ansätze zur Räumung im Rahmen von Weben. TTL Eviction (Time to Live) und Command Base Eviction Ansätze. Beide Ansätze können auf den gleichen Dienst angewendet werden. Time-to-Live- und Command Base Eviction-Ansätze können gleichzeitig für einen Service angewendet werden (Beispiel: „Rechnungsabfrage"). Die Basis des Time to Live Eviction-Ansatzes besteht darin, zu bestimmen, wie lange nach dem Speichern der Daten der Cache ungültig und automatisch gelöscht wird. Command Base Eviction ist das Client-System, das weiß welche Daten auf dem Cache nicht mehr aktualisiert werden (z.B "Rechnungssystem") und dies durch Senden eines Befehls an Weben realisiert. In diesem Fall, wenn Weben diesen Befehl sendet, wird es über den Cache gelöscht, ohne auf die Time to Live Dauer zu warten. Alle Konfigurationsvorgänge können für jeden Service getrennt durchgeführt werden.

Typische Web-Service-Architektur:

Eine typische Service-Provider-Architektur ist wie folgt.



Definition

Kunden(Client)-Anwendungen: Client- Anwendungen, die den Dienst von der Anwendung des Dienstanbieters aufrufen. Zum Beispiel Web Frontends, MobileAndwendungen usw.

Anwendungsserver: Die Plattform, die den Dienst und die Anwendung bereitstellt. Zum Beispiel Webdienste, die entwickelt wurden, um Dienste bereitzustellen und die WebLogic Application Server auf denen sie installiert sind.

Backend-Systeme: Systeme, die der Dienstanbieter durchläuft, um auf eingehende Dienstanforderungen zu reagieren. Beispielsweise Datenbanken, CRM-System usw.

Ablauf

1) QueryService (Abfragedienst): Die Client-Anwendung (Client) führt eine Serviceanfrage durch. Zum Beispiel queryInvoice (Rechnungsabfrage).

2) QueryData (Abfragedatei): Der Dienstanbieter erhält die Daten vom Back-End-System, in dem die eingehende Anfrageantwort gespeichert ist. Beispielsweise stellt die Fakturierung eine Verbindung zur Datenbank her und führt Abfragen durch.

3) RespondData(Antwortdatei): Der Service wird vom Backend-System bezogen.

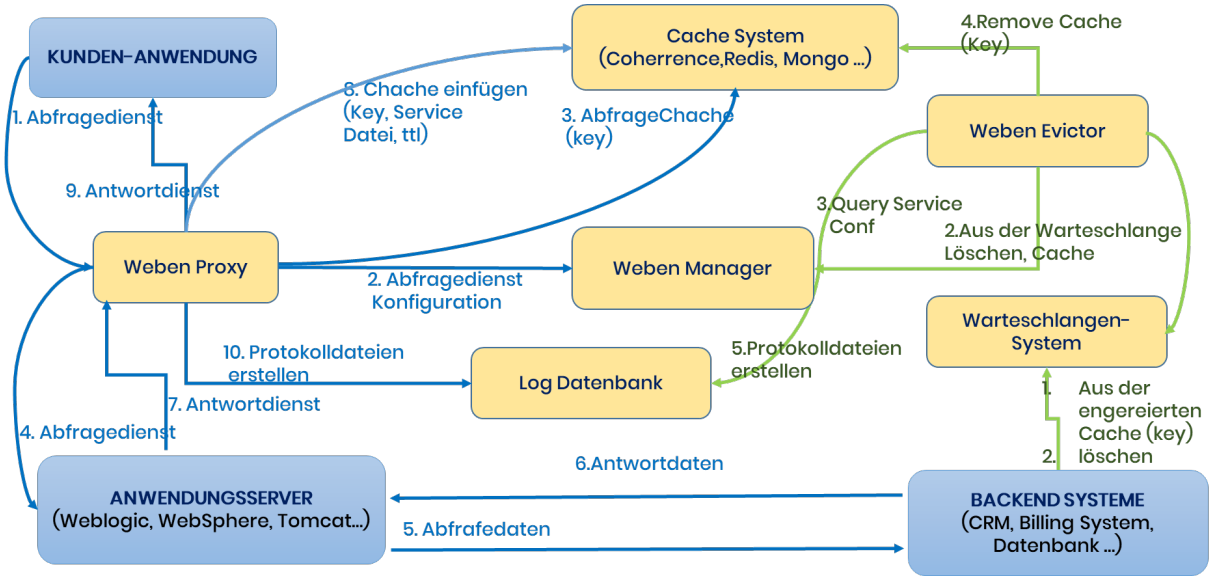
4) RespondService (Antwortdienst): Der Client antwortet der Anwendung bezüglich der Serviceanfrage.

Dieser 4-stufige Ablauf wird synchron betrieben. Wenn die Client-Anwendung dieselbe Abfrage (Beispiel queryInvoice) erneut mit der gleichen Eingabe ausführt (z.B. Beispiel customerId: 123456), wird dieser 4-Stufen-Ablauf erneut ausgeführt.

Die Weben-Architektur kommt an dieser Stelle ins Spiel. Wenn dieselbe Abfrage wiederholt wird, werden die Daten nun sehr schnell vom Cachesystem angefordert, wodurch sowohl der Service Provider als auch das Backend-System entlastet werden.

Web-integrierte Web-Service-Architektur:

Das Ergebnis der Weben-Integration ist die folgende Service-Provider-Architektur.



Die gelb dargestellten Module sind die Module, die mit der Weben Application, dem Ergebnis der Integration, geliefert werden.

Blau dargestellte Module sind Module die in der bestehenden Struktur bereits vorhanden sind.

Die blau dargestellten Flusslinien sind die Schritte der Service-Frage.

Die grün dargestellten Flusslinien sind die Schritte, die zu der Datenräumung des Caches gehören.

Definition

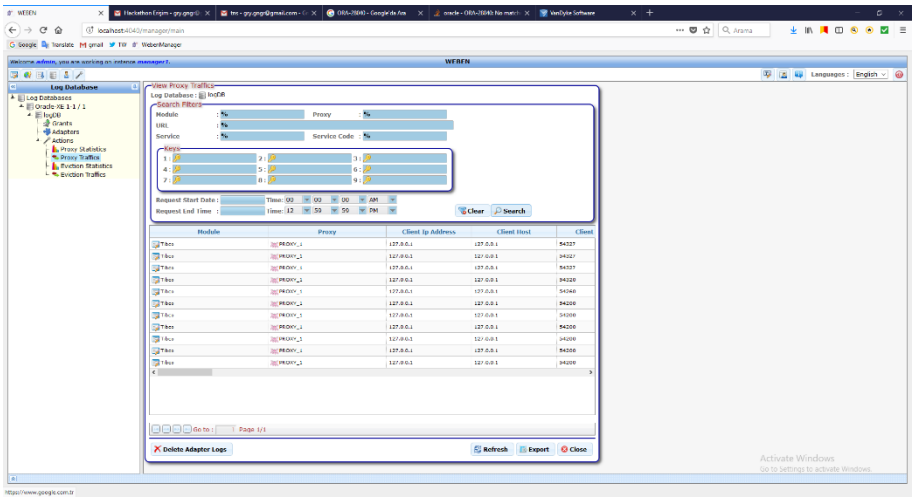
Client-Anwendungen: Client-Anwendungen, die den Dienst von der Anwendung des Dienstanbieters aufrufen. Zum Beispiel Web Frontends, mobile Anwendungen usw.

Anwendungsserver: Die Plattform, die den Dienst und die Anwendung bereitstellt. Zum Beispiel Webdienste, die entwickelt wurden, um Dienste bereitzustellen und die WebLogic Application Server auf denen sie installiert sind.

Backend-Systeme: Systeme, die der Dienstanbieter durchläuft, um auf eingehende Dienstanforderungen zu reagieren. Beispielsweise Datenbanken, CRM-System usw.

Weben Manager

Mit diesem Modul können alle Weben-Adapter (Weben-Proxy-Adapter und Weben-Evictor-Adapter) von einem einzigen Punkt aus konfiguriert und verwaltet werden. Die Konfiguration und Administration erfolgt über Web-Frontends, die von der Anwendung Weben-Manager bereitgestellt werden. Alle hier vorgenommenen Konfigurationsänderungen werden automatisch an die entsprechenden Adapter übertragen, ohne dass ein Neustart erforderlich ist. Die Weben-Manager-Anwendung speichert die Konfigurationsdaten in der Oracle-Datenbank.



Dieses Modul wird deinstalliert, d.h. wird eine installiert und die gesamte Weben-Topologie wird über diese zentrale Anwendung konfiguriert und verwaltet. Cache und Evicton Systeme, Module, Evictors, Services und Proxy die an diese Module angeschlossen sind, sind hier definiert. Die ausführlichste Beschreibung erfolgt in dem Servicebereich. Zum Beispiel bestimmt die Time to Live-Zeit des "Invoice Inquiry" - Dienstes, in welchem Cache-System die Daten gespeichert werden, welche Formeln (XPath, JSONPath, etc.) verwendet werden, um die Dienstinformationen (z. B. "Customer_Id") aus dem Anruf in den http-Anrufen dieses Dienstes abzurufen (Anfrage), welcher Teil der Antwort vom Back-End in den Cache als Rechnungsinformationen geschrieben wird, all dies wir vom Weben Managermodul definiert.

Diese Anwendung kann als .war Datei auf jedem vorhandenen Anwendungsserver bereitgestellt oder auch eigenständig ausgeführt werden.

Weben-Proxy

Der Client führt die "Service Execution Logic" zwischen der Anwendung und der Service Provider-Anwendung aus. Der Client und der Dienstanbieter führen die Proxy-Funktionalität und die Dienstausführungslogik aus, ohne Änderungen am Datenverkehr und am Format der Anwendung vorzunehmen, kurz gesagt ohne eine Anwesenheit zu zeigen. Diese Logik besteht darin, die eingehende Abfrage kurz zu analysieren und die Antwort an die Clientanwendung zurückzugeben, zum Backendsystem (Dienstanbieter) zurückzukehren, wenn der Cache nicht verfügbar

ist, und die Abfrage zurück an den Cache zu senden und zur Clientanwendung zurückzukehren, weil die Daten in den Cache gestellt werden und die gleiche Abfrage erneut angezeigt wird, erfolgt die Antwort direkt aus dem Cache, wodurch die Geschwindigkeit des Webdienstes erheblich verbessert und die Systemlast verringert wird.

Dieses Modul ist idealerweise so dimensioniert, dass es für jeden Web-Service-Cluster im System des Kunden eins zu eins passt. Kurz gesagt, jedem Web-Service wird vorher ein Proxy zugewiesen. Im Wesentlichen macht es http "tunneling" (tunneling) zwischen dem Client und den tatsächlichen Back-End-Webdiensten. Aber im Gegensatz zu einem Standard "Tunneling"-Anwendung wird hier die Service Execution Logicfür Weben gemäß den im Weben-Manager-Modul definierten Konfigurationen ausgeführt. Diese sequentielle Logik;

a) Definiert, welcher Dienst vom eingehenden http-Aufruf angerufen wird (zB "Rechnungsprüfung")

b) von den Servicedaten vom eingehenden http-Aufruf einen einzelnen Schlüssel (Key) (kann ein Composite Key sein) ein (z. B. "customerId") definieren.

c) fragt das Cache-System mit den erhaltenen Service- und Schlüsselinformationen ab.

d) Wenn die "Abrechnungsinformation" zu dem Schlüssel im Cache gefunden wird, lautet der Befehl "Gehe nicht zum Back-End-System, Rückgabe der Daten zum Client vom Cache“.

e) Wenn zum Schlüssel im Cache keine "Rechnungsinformationen" gefunden wird, wird der Befehl "Übergeben des eingehenden Anrufs an das Backend-System" und "Rechnungsinformationen" im Cache mit dem entsprechenden Schlüssel "Rechnungsanfrage" übergeben“ gegeben.

f) Wenn ein anderer Client die gleichen "Rechnungsinformationen" anfordert, angefordert, wird der Befehl "Daten von Cache an Client senden" ausgegeben.

Als Ergebnis dieser sequentiellen Logik werden die im Cache gespeicherten Daten in dem Zustand gespeichert, der von den Clients bis zur Räumung verwendet wird.

Weben Evictor

Das Backend-System löscht die Daten aus dem Cache, indem es regelmäßig die Datenlöschmeldungen (enqueueRemoveCache) aus dem Cache in das Queue-

System zieht. Aufgrund dessen wird der Cache nicht mehr aktualisiert, da er nicht mehr aktuell ist.

Dieses Modul kann beliebig oft eingerichtet und skaliert werden, um eingehende Eviction-Befehle möglichst schnell zu erfüllen und verarbeiten zu können. An dieser Stelle erfolgt die Kommunikation zwischen den Kundensystemen (z.B. "Billing System" des Kunden) und dem Evictor asynchronisch, sie werden über Warteschlangen (queue) ausgeführt. In diesem Zusammenhang werden mögliche technische Probleme und Leistungsprobleme, die im Weben-System auftreten können, im kritischen Kundensystem nicht auftreten. Zu diesem Zweck erfolgt die Kommunikation zwischen Evictor-Modulen und Kundensystemen asynchron über queue. Queue-Systeme heißen "Eviction-Queue-Systeme". Evictor-Module sind Module, die von den Kundensystemen regelmäßig in die Warteschlange gezogen und aus den entsprechenden Cache verworfen werden. Außerdem werden Systeme in denen Cache festgehalten wird als "Cache-Systeme" bezeichnet.

Cache-System

Eine Anwendung, in der Service-Daten zwischengespeichert werden. Es kann eines von In-Memory Oracle Data Grid (Coherence, Hazelcast, Redis, vs..) oder No-Sql Database (Mongo, Cassandra, vs..) oder Sql Database (Oracle, MySql, vs.) Technologien sein. Die Weben-Anwendung führt einen Datencache-Prozess durch, der in diese Systeme integriert ist. Cachesysteme müssen extern als Stand-Alone installiert werden, und die zwischengespeicherte Version ist von jedem beliebigen Punkt aus zugänglich und nutzbar. Mit anderen Worten, ein anderer Proxy kann auf den Cache zugreifen der vom Weben-Proxy in den Cache gelegt wurde.

Protokolldatenbank

Es ist die Datenbank, in der Verkehrs- und statistische Informationen von Adaptern gespeichert sind.

Queue-System

Die Queue-Technologie, die Räumungsnachrichten von den Backend-Systemen zur Verarbeitung speichert. (Oracle Advanced Queue, JMS, etc ..)

Stream - 1 (Serviceabfrage - nicht im Datencache gefunden):

1) queryService (Abfragedienst): Die Client-Anwendung führt eine Serviceanfrage durch. Zum Beispiel queryInvoice (Rechnungsabfrage).

2) queryServiceConf (AbfragedienstKonfiguration): Der Proxy ruft alle über den Manager für den Dienst definierten Konfigurationsinformationen ab. Da

diese Informationen im Speicher des Proxyservers gespeichert sind, wird diese Abfrage nicht in jedem Fluss wiederholt und die bereits im Speicher befindlichen Dienstkonfigurationsinformationen werden verwendet. Informationen zur Dienstkonfiguration Änderungen, leitet der Manager diese Informationen automatisch an alle Proxy-Adapter weiter und aktualisiert diese Informationen, die die Proxys in ihrem Speicher behalten.

3) queryCache (Abfrage Cache): Der Proxy fragt die Servicedaten vom Cache-System ab und kann die Daten im Cache nicht finden.

4) queryService (Abfragedienst): Proxy **leitet die** Dienstabfrage an das Backend-System weiter.

5) queryData (Abfragedatei): Der Dienstanbieter erhält die Daten vom Back-End-System, in dem die eingehende Anfrageantwort gespeichert ist. Beispielsweise stellt die Rechnungsschreibung eine Verbindung zur Datenbank her und führt Abfragen durch.

6) respondData (Antwortdatei): Der Service wird vom Backend-System bezogen.

7) respondService (Antwortdienst): Proxy empfängt die Antwort der Dienstabfrage vom Back-End-System.

8) putCache (Cache einfügen): Der Proxy stellt die vom Backend-System erhaltenen Servicedaten in den Cache. An diesem Punkt entscheidet er, wie lange er sich im Cache befindet, indem er sich die Servicekonfigurationsdaten ansieht und diese als TTL (Time to Live) festlegt und verarbeitet das Cache-System. Am Ende der TTL-Periode werden die Daten automatisch durch Timeout aus dem Cache-System gelöscht.

9) respondService (Antwortdienst): Antwort auf die Serviceanfrage der Client Anwendung.

10) writeLog (Erstellen von Portokolldatein): Alle Vorgänge auf dem Proxy werden in der Protokolldatenbank verarbeitet.

Wenn eine Service-Frage zum ersten Mal für eine Eingabe bearbeitet wird, werden die obigen Ablauf-1-Schritte ausgeführt. Am Ende dieses Ablaufs werden die Daten im Cache-System zwischengespeichert. Wenn der Service mit der gleichen Eingabe einen Service ausführt, folgen die folgenden Ablauf-2-Schritte.

Stream - 2 (Serviceabfrage - Daten im Cache gefunden):

1) queryService (Abfragedienst): Clientanwendungsdienstanforderung. Zum Beispiel queryInvoice .

2) queryServiceConf (Abfragedienst Konfiguration): Der Proxy ruft alle über den Manager für den Dienst definierten Konfigurationsinformationen ab. Da diese Informationen im Speicher des Proxyservers gespeichert sind, wird diese Abfrage nicht in jedem Fluss wiederholt und die bereits im Speicher befindlichen Dienstkonfigurationsinformationen werden verwendet. Wenn sich die Informationen zur Dienstkonfiguration ändern, leitet der Manager diese Informationen automatisch an alle Proxy-Adapter weiter und aktualisiert diese Informationen, die die Proxys in ihrem Speicher behalten.

3) queryCache (Abfragedienst): Proxy fragt Service-Daten aus dem Cache-System ab und **ruft sie** ab.

4) respondService (Antwortdienst): Client (Client) Antwort auf die Serviceanfrage der Anwendung.

5) writeLog (Erstellen von Portokolldatein): Alle Vorgänge auf dem Proxy werden in der Protokolldatenbank verarbeitet.

Es wird erklärt, wie die TTL-basierte Löschung der Daten, die aus dem Cache gelöscht werden müssen, aufgrund der obigen Aktualisierung verloren geht. Die TTL-basierte Löschung ist die Bestimmung der Lebensdauer während des Einbringens der Daten in den Cache (putCache). Wenn dies jedoch alleine nicht ausreicht, kann eine auf Eviction basierende Löschung aus dem Cache mit oder ohne TTL durchgeführt werden. Eine auf der Evidenz basierende Löschungstechnik kann bevorzugt sein, wenn die Lebensdauer nicht bekannt ist, während die Daten in den Cache gelegt werden.

Eviction-basiertes Löschen der Daten aus dem Cache ist die Methode, wie das Backend- System den Moment erkennt, in dem die Daten gelöscht werden müssen und es dies als Befehl (Nachricht) an das Weben-System sendet. (enqueueremovecach A). Dieser Ablauf wird in den Schritten Ablauf - 3 erläutert. Diese Kommunikation zwischen Backend-Systemen und Weben Evictor ist asynchron. In diesem Fall reduziert sich der Räumungsprozess für das Backend-System auf nur eine Sendung einer Nachricht an eine queue.

Ablauf - 3 (Eviction):

1) enqueueRemoveCache (Aus der engereierten Cache löschen): Das Backend-System sendet eine Nachricht an das Queue-System für die Daten, die es aus dem Cache löschen möchte.

2) dequeueRemoveCache (Aus der Warteschlage löschen): Die **Evictor-**Anwendung **ruft die** Räumungsnachricht aus der Warteschlange ab.

3) queryServiceConf (Abfragedienst Konfiguration): Der Proxy ruft alle über den Manager für den Dienst definierten Konfigurationsinformationen ab. Da diese Informationen im Speicher des Proxyservers gespeichert sind, wird diese Abfrage nicht in jedem Ablauf wiederholt und die bereits im Speicher befindlichen Dienstkonfigurationsinformationen werden verwendet. Wenn sich die Informationen zur Dienstkonfiguration ändern, leitet der Manager diese Informationen automatisch an alle Proxy-Adapter weiter und aktualisiert diese Informationen, die die Proxys in ihrem Speicher behalten.

4) removeCache (Cache Löschen): Löscht die Daten aus dem Evictor-Cache.

5) writeLog (Erstellen von Portokolldateien): Alle Vorgänge auf dem Proxy werden in der Protokolldatenbank verarbeitet.

Ein wöchige Statistik im Kundensystem:

- **Gesamtzahl der Web-Service-Abfragen / -Antworten: 40.626.981**
- **Anzahl der Antworten direkt aus dem Cache: 11.492.271**
- **Anzahl der Antworten direkt aus dem Backend-System: 29.134.710**
- **Cache-Aufsuchrate: 28%**
- **Durchschnittliche Antwortzeit für Web-Service ohne WEBEN: 1065 ms. (> 1 s.)**
- **Durchschnittliche Antwortzeit für Web-Service vom WEBEN-Cache: 1 ms.**
- **Durchschnittliche Antwortzeit für Web-Service auf WEBEN: 764 ms. (Mit einer Caching-Rate von 28%)**
- **Verringerung der CPU-Auslastung: 41%**