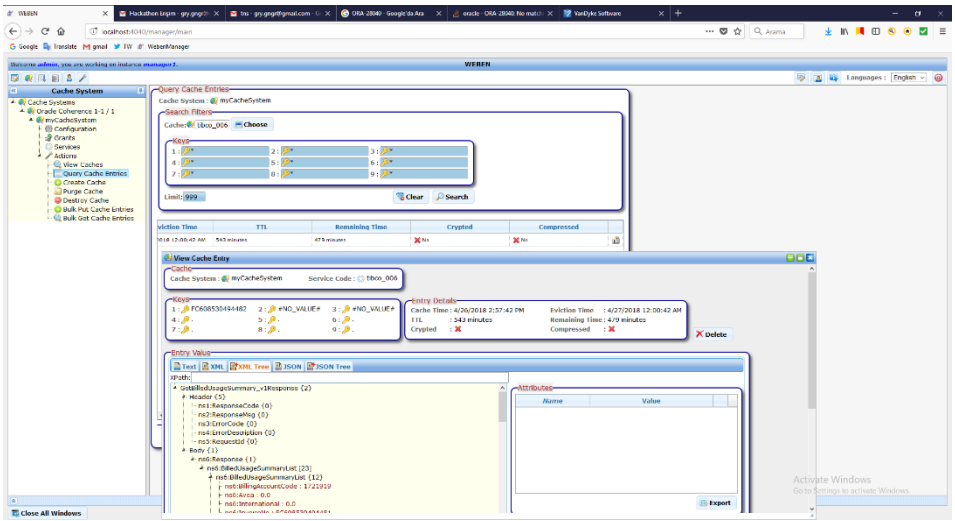
WEBEN Application

Definition:

Weben is an enterprise application which aims to decrease the response time and increase the performance of the all kinds of http based query service (SOAP, RESTFul, for example) exposed by service provider systems by storing the service data in a cache system and returning the response from the cache when the same query is made again and decrease the load of service provider system by reducing the consumption of resources such as CPU, Memory, I/O.

Needs met by Weben:

The basic approach applied with Weben is keeping query data of client applications in a cache with its input criteria. In this way when the same query is received from the same service with the same input criterion, this request will be sent directly to the client application via the cache without calling back-end system. Hereby this will result in a significant improvement in response time for both client application responses (performance improvement) as well as a reduction in the overhead of system resources (CPU, Memory, Disc, Network) or back-end systems. Performance improvement and decreasing load is the measure of success that Weben application aims. This measure is directly proportional to the cache hit rate (Cache Hit Rate) of the queried data as it is in all cache systems.

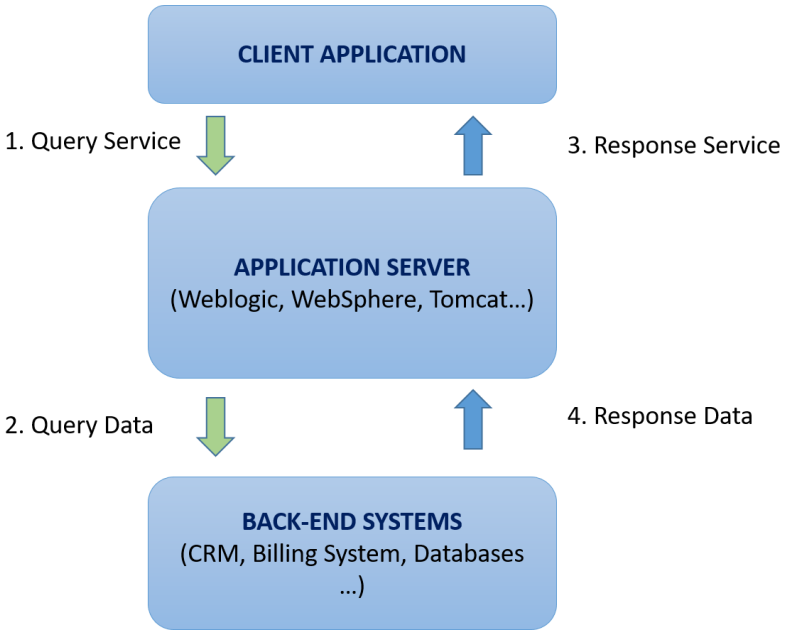


One of the issues that needs to be solved with Weben is eviction. Removing data from cache because of the lifetime is over is called eviction. The data placed on the cache has a lifetime and should be deleted from the cache when its timeliness and validity are lost. Otherwise, both data lost its timeliness and validity is returned to the querying client application and also causes unnecessary data stacking on the cache.

In order to solve this problem, there are two approaches to the eviction process within the scope of Weben. TTL Eviction (Time to Live) and Command Base Eviction approaches. Both of these approaches can be applied to the same service. Time to Live and Command Base Eviction approaches can be applied to a service (for example "Query Invoice") at the same time. In the Time to Live Eviction approach is to determine how long afterwards the cache will be invalid and will be automatically deleted. Command Base Eviction is sending an eviction command to Weben by that the client system (for example "Billing System") knows that the data on the cache is no longer valid. In this case, when this command is sent to the Weben itself, data will be cleared from the cache without waiting for the Time to Live for that data to be expired. In addition, the Time to Live duration can be configured as "infinite". All configuration operations can be done separately for each service.

Typical Web Service Architecture:

A typical service provider architecture is as follows.



Definitions :

Client Applications: The client applications that call the service from the service provider application. For example Web User Interfaces, Mobile Apps etc.

Application Server: The service application and platform on which it is deployed. For example web services developed to provide service, and WebLogic Application Server where it is deployed.

Backend Systems: Systems that the service provider is accessing to respond to incoming service requests. For example databases, CRM etc.

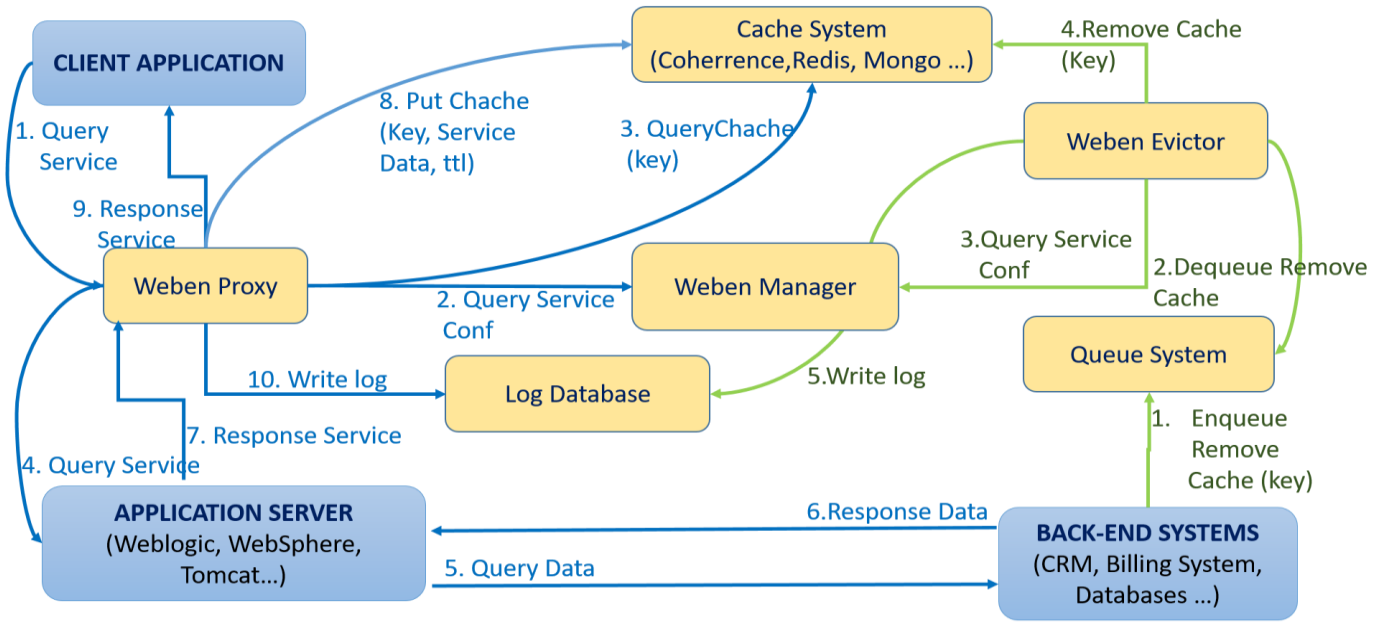
Flow:

1) QueryService: Client Application queries a service. For example query Invoice.
2) QueryData: The service provider obtains data from the Backend System where the incoming response of request is stored. For example, connects to the billing database and execute queries.
3) RespondData: Service data is obtained from backend system.
4) RespondService: Client application is responded for the service query. This 4-step flow is operated synchronously. When the client application repeats the same query (For example queryInvoice) with the same input (for example customerId: 123456), this 4-step flow is run again.

Weben architecture comes into play at this point. When the same query is repeated the data will now be requested from the cache system very quickly and the load on both the service provider and the backend system will be reduced.

Weben-Integrated Web Service Architecture:

Weben-Integrated Web Service Architecture is as follows.



Blue colorized modules are already exist in the current architecture.

Yellow colorized modules comes with Integration of Weben Application..

Blue colorized line flow belongs to the service query.

Green colorized line flow belongs to the data removing request from cache (eviction).

Definitions:

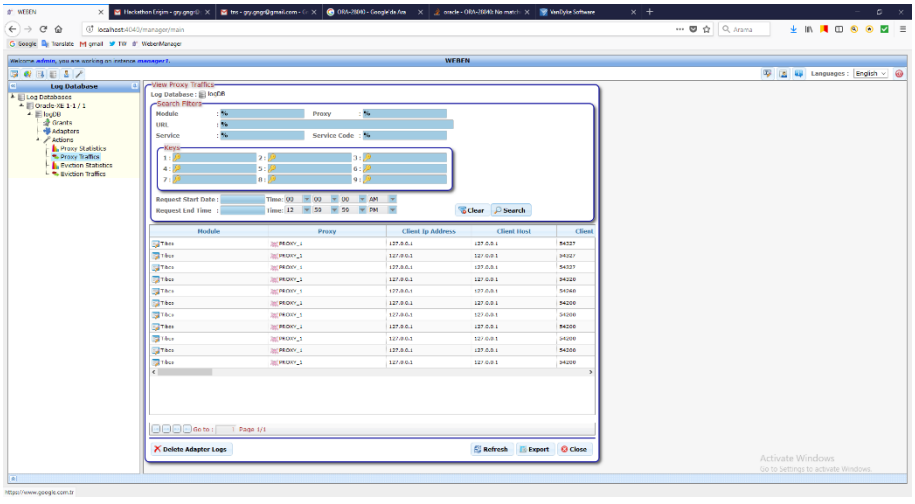
Client Applications: The client applications that call the service from the service provider application. For example Web User Interfaces, Mobile Apps vs.

Application Server: The service application and platform on which it is deployed. For example web services developed to provide service, and WebLogic Application Server where it is deployed.

Backend Systems: Systems that the service provider is accessing to respond to incoming service requests. For example databases, CRM for example

Weben Manager: It is a module that allows all Weben Adapters (Weben Proxy and Weben Evictor Adapters) to be configured and managed from a single point. The configuration and management is performed by web frontends provided by the Weben Manager application. Any kind of configuration changes made here are automatically transmitted to the respective adapters without any need for a restart operation. The Weben Manager application stores the configuration data in the Oracle database.

This module is unique so deployed only once and the entire Weben topology is configured and managed through this central application. Cache and Eviction



Systems, Modules and Modules connected to that Proxies, Evictors, and Services are defined here. The most detailed definition is made in the service subject. For

example "queryInvoice" service's Time to Live period, which Cache system will keep the data, which formulas (XPath, JSONPath for example) will be used that makes unique (for example "Customer_Id") service data in the http requests (request) of this service, which part of the response from back-end will be written to Cache as billing data are defined in Weben Manager module.

This application can be deployed to any existing application server as a .war file or run as stand-alone.

Weben Proxy: It executes "Service Execution Logic" between Client Application and Service Provider Application. The Client and Service Provider implement proxy functionality and execute the Service Execution Logic without making any changes to the application's data traffic format, without briefly making sense of its presence. This logic is more briefly described below as analyse the request and return the response from the cache to the client application, to send back the query to the backend system (service provider) in case the cache is not available and to return the response to the cache and return to the client application. Since the data is put in the cache, when the same query comes up again, the response will be given directly from the cache and this will save a significant decrease in the system load as well as a significant improvement in web service speed.

This module is ideally deployed to one-to-one for each web service cluster in the customer system. In short there is a Proxy assigned to each web service in front of it. Essentially it makes http "tunnelling" between the client and the real back-end web services. But unnlike a standard "Tunnelling" application, the Service Execution Logic for Weben is performed entirely according to the configurations defined in the Weben Manager Module. This sequential logic;

a) Define which service is called from the incoming http request. (for example “queyInvoice”)
b) Determine whether the server sends a unique key (may be a composite key) from the incoming http request (for example “CustomerId”).
c) Query the Cache system with the service and key information you get.
d) If the "Billing Information" belongs to the key exists in Cache return that to the client application without calling backend system.
e) If there is no "Billing Information" belonging to the key in Cache, redirect the incoming call to the backend system and store "Billing Information" of the "queryInvoice" service with the relevant key in the Cache.
f) When another client that requests the same "Billing Information" it is sent from Cache to the Client Application.

As a result of this sequential logic being performed, the data held will be stored in the Cache used by the clients until eviction.

Weben Evictor: Evictor removes the data from the cache by regularly pulling the data deletion messages from queue system which are pushed by backend system. In this way expired data will be removed from the cache.

This module can be set up and scaled as many times as needed to meet incoming Eviction commands as quickly as possible and to be able to process them. In this point communication between the client systems (For example “Billing System”) and the evictor is asynchronous and made via queues. In this regard, potential, temporary, technical and performance problems that may occur in the Weben system will not be reflected in the critical customer system. For this purpose communication between evictor modules and customer systems is done asynchronously via queues. Queue systems are called "Eviction Queue Systems". Evictors are modules that pulls data pushed by client systems regularly from queue and remove the data from the relevant cache. In addition the systems which stores caches are called "Cache Systems".

Cache System: It is an application which caches services data. It may be one of these technologies, In-Memory Oracle Data Grid (Coherence, Hazelcast, Redis, for example) or No-Sql Database (Mongo, Cassandra, for example) or Sql Database (Oracle, MySql, etc.). Weben Application performs data caching process integrated with these systems. Cache systems must be installed in stand-alone externally, and the cached data will be accessible and usable from any point. This means that a Weben Proxy can access and use the data that is put by another proxy.

Log Database: It is the database where traffic and statistical information of adapters are stored.

Queue System: These are the queue technologies for storing eviction messages which comes from the backend systems to process.(Oracle Advanced Queue, JMS, for example)

Flow – 1 (Query Service – without DataCache):

1) QueryService: Client Application queries a service. For example queryInvoice.
2) QueryServiceConf: Proxy retrieves all configuration information defined for service via Manager. Essentially this information is stored in memory in Proxy, so this query is not repeated in every flow, the service configuration information that is already in memory is used. If the service configuration information changes, the Manager automatically transmits this information to all Proxy Adapters and updates this information stored in Proxy's memories.

3) QueryCache: Proxy Cache system queries the service data and cannot find in the cache.
4) QueryService: Proxy redirects the service inquiry to the backend system.
5) QueryData: The service provider obtains data from the Backend System where the incoming response of request is stored. For example, connects to the billing database and execute queries.
6) RespondData: Service data is obtained from backend system.
7) RespondService: Proxy receives the response of the service inquiry from the backend system.
8) PutCache: Proxy puts the service data to the cache which receives from the backend system. At this point t decides how long it will wait in the cache by looking at the service configuration data and sets the time to TTL (Time to Live) and processes it into the cache system. At the end of the TTL period the data is automatically expired from the cache system.
9) RespondService: Client application is responded for the service query.
10) WriteLog: All operations on the Proxy are stored in the log database.

When a service question is processed for the first time for an input the above Flow-1 steps are performed. At the end of this flow the data is put into the cache in the Cache System. When the service with the same input for this service is performed the following Flow-2 steps will follow.

Flow – 2 (Query Service – with Data Cache):

1) QueryService: Client Application queries a service. For example queryInvoice.
2) QueryServiceConf: Proxy retrieves all configuration information defined for service via Manager. Essentially this information is stored in memory in Proxy, so this query is not repeated in every flow, the service configuration information that is already in memory is used. If the service configuration information changes, the Manager automatically transmits this information to all Proxy Adapters and updates this information stored in Proxy's memories.
3) QueryCache: The proxy queries and retrieves the service data from the Cache System.
4) RespondService: Client application is responded for the service query.
5) WriteLog: All operations on the Proxy are stored in the log database.

It is explained above because of loss of the validity how to delete the TTL-based data that needs to be deleted from the cache. TTL-based deletion is the determination of the life time while putting data (putCache) to the cache. However if this alone is not enough, Command Base Eviction from cache can be performed with or without TTL. Command Base Eviction technique may be preferred when the life time is not known while putting the data in the cache.

Command Based Eviction from the cache is the backend system recognizes the time when the data must be removed and sends it to the Weben system as a command (enqueueRemoveCache). This flow is explained in Flow - 3 steps. This communication between backend systems and Weben Evictor is asynchronous. In this manner the eviction process for the backend system has been reduced to the simplicity of sending only one message to a queue.

Flow – 3 (Eviction) :

1) EnqueueRemoveCache: The backend system sends a message to the Queue system for the data wants to delete from the cache.
2) DequeueRemoveCache: Evictor application dequeues pull the eviction message from the queue.
3) QueryServiceConf: Evictor retrieves all configuration information defined for service via Manager. Essentially this information is stored in memory in Evictor, so this query is not repeated in every flow, the service configuration information that is already in memory is used. If the service configuration information changes, the Manager automatically transmits this information to all Evictor Adapters and updates this information stored in Evictor's memories.
4) RemoveCache: Evictor removes data from the cache.
5) WriteLog: All operations on the Evictor are stored in the log database.

Success Story

1 week of statistics on a customer site:

- Total number of web service requests/responses: 40.626.981
- Number of responses from directly cache: 11.492.271
- Number of responses from backend web service: 29.134.710
- Cache Hit Ratio: %28
- Average web service response time without WEBEN: 1065 ms. (> 1 s.)
- Average web service response time with WEBEN cache: 1 ms.
- Average web service response time with WEBEN: 764 ms. (with %28 cache hit ratio)
- Reduce of CPU usage: %41