Web caching

Corso di ***Applicazioni Telematiche***

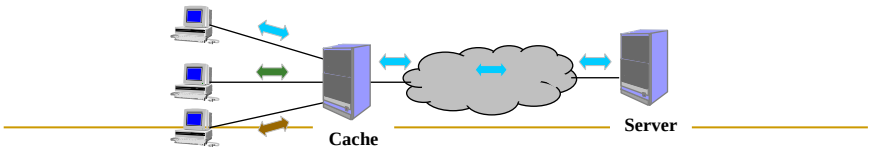
A.A. 2005-06 – Lezione n.3 - parte I

Prof. Roberto Canonico

Università degli Studi di Napoli Federico II
Facoltà di Ingegneria

Web caching

- Si parla genericamente di Web caching quando le richieste di un determinato client non raggiungono il Web Server ma vengono intercettate da una proxy cache
- Tipicamente, un certo numero di client di una stessa rete condividono una stessa cache web, posta nelle loro prossimità (es. nella stessa LAN)
- Se l'oggetto richiesto non è presente nella cache, questa lo richiede *in vece* del client conservandone una copia per eventuali richieste successive
- Richieste successive alla prima sono servite più rapidamente
- Due tipi di interazione HTTP: client-cache e cache-server



Caching: vantaggi

- Possibilità di limitare il traffico entrante e/o uscente da una specifica rete servita da un opportuno sistema di caching
- Diminuzione, avvertibile dal client, della latenza relativa al recupero degli oggetti Web
- In alcune configurazioni può aiutare a realizzare una forma di *load balancing*

Problemi connessi al caching

- Violazione del modello *end-to-end*
  - sono più difficili , per il Content Server, le forme di autenticazione e la personalizzazione dei contenuti
- Gestione della coerenza
  - gli oggetti disponibili nelle cache devono essere allineati con gli originali
  - una possibile soluzione:
    il content server associa ad ogni oggetto un'informazione sulla validità temporale del documento; le cache utilizzano questa informazione per calcolare un "Time-To-Live"
- Non tutti gli oggetti sono "*cacheable*": alcune tipologie di contenuti, per la loro dinamicità non si prestano ad essere conservati localmente

Caching (1)

- Può essere client-side, server-side o intermedia (su un proxy)
- La cache server-side riduce i tempi di computazione di una risposta, ma non ha effetti sul carico di rete
- Le altre riducono il carico di rete
- HTTP 1.0 si basava su tre header:
  - **Expires**: il server specifica la data di scadenza di una risorsa
  - **If-Modified-Since**: il client richiede la risorsa solo se modificata dopo il giorno X. Richiede una gestione del tempo comune tra client e server
  - **Pragma: no-cache**: Fornita dal server, istruisce il client di non fare cache della risorsa in ogni caso
- HTTP 1.1 introduce due tipi di cache control:
  - Server-specified expiration
  - Heuristic expiration

Caching (2)

- Server-specified expiration
  - Il server stabilisce una data di scadenza della risorsa, con l'header Expires o con la direttiva max-age nell'header Cache-Control
  - Se la data di scadenza è già passata, la richiesta deve essere rivalidata. Se la richiesta accetta anche risposte scadute, o se l'origin server non può essere raggiunto, la cache può rispondere con la risorsa scaduta ma con il codice 110 (Response is stale)
  - Se Cache-Control specifica la direttiva must-revalidate, la risposta scaduta non può mai essere rispedita. In questo caso la cache deve riprendere la risorsa dall'origin server. Se questo non risponde, la cache manderà un codice 504 (Gateway time-out)
  - Se Cache-Control specifica la direttiva no-cache, la richiesta deve essere fatta sempre all'origin server

Caching (3)

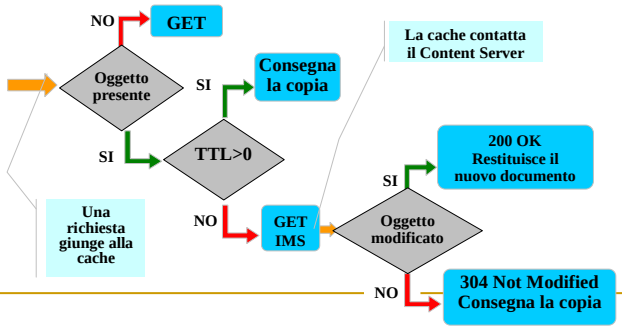
- Heuristic expiration
  - Poiché molte pagine non conterranno valori espliciti di scadenza, la cache stabilisce valori euristici di durata delle risorse, dopo le quali assume che sia scaduta
  - Queste assunzioni possono a volte essere ottimistiche, e risultare in risposte scorrette

Caching (4)

- Validazione della risorsa in cache
  - Anche dopo la scadenza, nella maggior parte dei casi, una risorsa sarà ancora non modificata, e quindi la risorsa in cache valida
  - Un modo semplice per fare validazione è usare HEAD: il client fa la richiesta, e verifica la data di ultima modifica. Ma questo richiede una richiesta in più sempre
  - Un modo più corretto è fare una richiesta condizionale: se la risorsa è stata modificata, viene fornita la nuova risorsa normalmente, altrimenti viene fornita la risposta 304 (not modified) senza body della risposta. Questo riduce il numero di richieste

Gestione della coerenza

- Problema: cosa succede se l'oggetto presente nel server è aggiornato ?
- La copia in cache deve essere aggiornata per mantenersi uguale all'originale
- HTTP fornisce due meccanismi per la gestione della coerenze:
  - TTL (Time To Live) : il server quando fornisce un oggetto dice anche quando quell'oggetto "scade" (header *Expires*)
    - Quando TTL diventa < 0, non è detto in realtà che l'oggetto sia stato realmente modificato
  - Il client può fare un ulteriore controllo mediante una GET condizionale (*If-Modified-Since*)



Parametri per valutare le prestazioni di una cache

- *Page Hit Rate*
  - percentuale di oggetti richiesti forniti senza interpellare il Server originario
- *Byte Hit Rate*
  - rapporto tra byte forniti localmente e byte richiesti al Content Server

Politiche per la gestione degli oggetti conservati nella cache

- Al riempirsi della cache, possono essere utilizzate tecniche diverse per il *replacement* dei documenti:
  - Least Recently Used (LRU)
  - Least Frequently Used (LFU)
  - Least Frequently Used with Dinamic Aging (LFUDA)
  - Greedy Dual Size e sue varianti

Organizzazione e comunicazione tra cache

- In presenza di più cache, queste possono essere organizzate in strutture gerarchiche o distribuite
- L'appartenenza di una cache ad una generica struttura può essere statica o dinamica
- Opportuni protocolli di gestione e comunicazione facilitano l'organizzazione e lo scambio di dati tra differenti cache
  - ICP (IETF)
  - WCCP (CISCO)
  - Cache-Digest (IETF)

Domande?